Output-based Mesh Adaptation for High Order Navier-Stokes Simulations on Deformable Domains

Steven M. Kast*, Krzysztof J. Fidkowski

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

Abstract

We present an output-based mesh adaptation strategy for Navier-Stokes simulations on deforming domains. The equations are solved with an arbitrary Lagrangian-Eulerian (ALE) approach, using a discontinuous Galerkin finite-element discretization in both space and time. Discrete unsteady adjoint solutions, derived for both the state and the geometric conservation law, provide output error estimates and drive adaptation of the space-time mesh. Spatial adaptation consists of dynamic order increment or decrement on a fixed tessellation of the domain, while a combination of coarsening and refinement is used to provide an efficient time step distribution. Results from compressible Navier-Stokes simulations in both two and three dimensions demonstrate the accuracy and efficiency of the proposed approach. In particular, the method is shown to outperform other common adaptation strategies, which, while sometimes adequate for static problems, struggle in the presence of mesh motion.

Keywords: Output error estimation, Space-time adaptation, Unsteady adjoint, Arbitrary Lagrangian-Eulerian, Discontinuous Galerkin, Geometric conservation law

1. Introduction

In a typical CFD simulation, most of the data goes to waste. The user is often interested in computing a certain output such as lift or drag, and has little use for auxiliary information. But if this is the case – if the sole objective is to obtain an accurate output – resolving all regions of the flow with equal precision is both unnecessary and inefficient. Instead, a better strategy is to (i) estimate the amount of error in the output, (ii) determine *where* this error originates from, and (iii) drive that error down by targeting the regions of the mesh responsible for it. This strategy of output error estimation and mesh adaptation is especially important for unsteady problems. Here, we present these output-based techniques for Navier-Stokes simulations with moving meshes, such as a wing in flapping flight, and show the efficiency gains possible for these cases.

In large part, the success of an output-based method hinges on the quality of the output error estimate. Accurate *a posteriori* error estimates can be obtained by solving an adjoint problem for the outputs of interest, and this is the strategy we adopt here. An adjoint provides the sensitivity of an output to perturbations in the residuals of the governing equations, which in the context of error estimation can identify regions of the domain contributing most to the output error. These regions can then be adapted to reduce the error and to obtain a more accurate output. This concept is not new, and has received considerable attention for steady problems [1, 2, 3, 4, 5, 6, 7]. However, focus has only recently shifted toward use of these methods for unsteady problems.

Computing an unsteady adjoint can be expensive, due in part to significant storage requirements for nonlinear problems, and techniques for reducing these costs are the subject of ongoing research [8, 9, 10, 11, 12]. Fortunately, however, the large costs often come with an even larger payoff. These payoffs have been observed in optimization problems [13, 14, 15] where a premium is placed on obtaining accurate sensitivities with respect to a large number of inputs. This same reasoning extends to unsteady CFD simulations, where small errors made in remote regions of the

*Corresponding author

Email address: kastsm@umich.edu (Steven M. Kast)

space-time domain can coalesce into large errors in the output of interest. Identifying these errors with an unsteady adjoint and eliminating them through mesh adaptation can lead to significant computational savings.

Some work in this area has already been done. In a finite-element context, output error estimation for scalar parabolic problems was studied in [16] and [17], with a high order reconstructed adjoint used to drive dynamic space-time mesh adaptation. Recently, spatial-only [18] and combined space-time [19] adaptation have been performed for two-dimensional Navier-Stokes simulations on static domains. Within a finite volume framework, temporal-only adaptation has been shown for the Euler equations on deforming domains [20, 21], while on static domains a spatial-only [22] and preliminary space-time adaptation [23] have been demonstrated. Finally, in recent work [24, 25, 26], Fidkowski *et al* presented combined space-time adaptation strategies for Euler and Navier-Stokes simulations on static domains. In each of the above works, improvements in output convergence were obtained through the use of unsteady adjoint-based adaptation.

In this work, we extend output-based adaptation techniques to high order Navier-Stokes simulations on deforming domains, in both two and three dimensions. Such simulations have far-reaching applications, from bio-inspired flight to aircraft maneuver and flutter analysis. The runs are generally computationally intensive and the resulting solutions are often rich in features. Building on previous work [26, 27], we employ combined space-time adaptation of the computational domain, with dynamic-in-time order refinement used to resolve the critical flow features identified by the adjoint. On deformable domains, satisfaction of the so-called geometric conservation law (GCL) requires the adjoint system to change. In this work, we present the required modifications to the adjoint system for an ALE discontinuous Galerkin (DG) method [28], and derive a new discrete adjoint for the GCL itself. We then incorporate this adjoint into the error estimation and adaptation strategy, and evaluate its performance on a series of 2D and 3D test cases. These cases verify the validity of the output-based strategy and show the computational savings that can be achieved.

The remainder of the paper is organized as follows: in Section 2 we introduce the mapping for deformable domains; in Section 3 we discuss the GCL; in Section 4 we describe the primal discretization; in Section 5 we present the error estimation and adjoint discretization; and in Section 6 we discuss the mesh adaptation. Finally, results for several compressible Navier-Stokes simulations are presented in Section 7.

2. Arbitrary Lagrangian-Eulerian Mapping

The Navier-Stokes equations can be written in conservation form as

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \qquad \vec{\mathbf{F}} = \vec{\mathbf{F}}^i(\mathbf{u}) - \vec{\mathbf{F}}^v(\mathbf{u}, \nabla \mathbf{u}), \tag{1}$$

where $\mathbf{u}(\vec{x}, t) \in \mathbb{R}^s$ is the state vector, $\vec{x} \in \mathbb{R}^d$ is the spatial coordinate, $t \in \mathbb{R}$ is time, and $\vec{\mathbf{F}}^i$ and $\vec{\mathbf{F}}^v$ are the inviscid and viscous fluxes, respectively [29]. For the cases considered in this work, the physical domain in which Eqn. 1 holds is deforming in time, and a direct solution would be difficult to obtain. Instead, we can map the problem to a fixed reference domain and solve using an arbitrary Lagrangian-Eulerian (ALE) approach. Since the reference domain remains fixed for all time, standard numerical methods for static problems can then be employed to obtain the solution. A simple and effective ALE method for DG was recently introduced by Persson *et al* [28], and we follow their approach here. In this method, the physical equations are transformed to equivalent reference-domain equations, and the deformation of the mesh is encapsulated within the mapping between these domains.

2.1. ALE mapping

The transformation between reference and physical domains is summarized graphically in Figure 1, and definitions of relevant variables are given in Table 1. Each point $\vec{X}$ in the static reference domain is mapped to a corresponding point $\vec{x}(\vec{X}, t)$ in the physical domain, based on some desired deformation of the mesh. With this mapping, we can then relate each point in the deformed physical domain *back* to a fixed point in the reference domain. The strategy is then to solve for new state variables $\mathbf{u}_X = g\mathbf{u}$ in the reference domain, and later apply the inverse transformation to obtain the physical states we are actually interested in. The variable g is the Jacobian of the mapping, and its deviation from unity indicates whether a region in the reference domain is contracted or dilated in the physical domain.

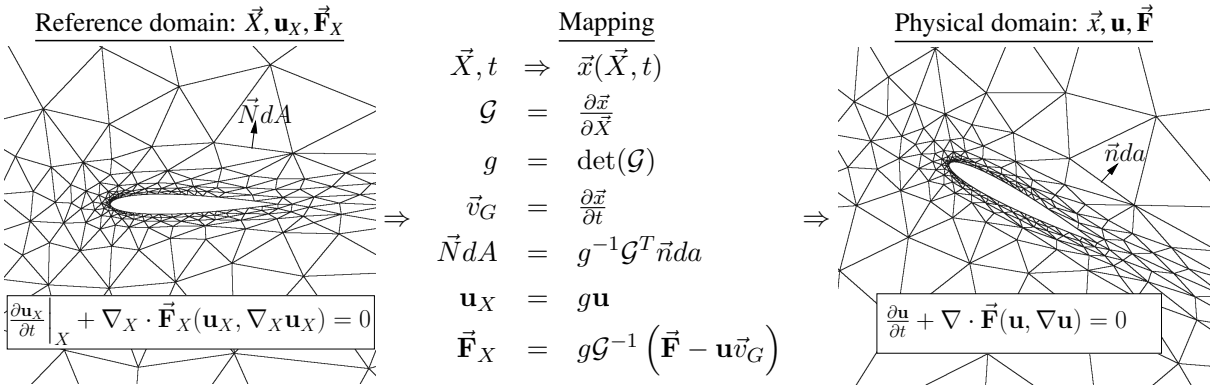


Figure 1: Summary of the mapping between reference and physical domains. The equations are solved on the reference domain, which remains fixed for all time. When denoting reference-domain quantities, we use a subscript X rather than $\vec{X}$ for visual clarity.

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

$\vec{X}$	=	reference-domain coordinates	$\vec{x}$	=	physical-domain coordinates
$\mathbf{u}_X$	=	state on reference domain	$\mathbf{u}$	=	physical state
$\vec{\mathbf{F}}_X$	=	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
$\mathcal{G}$	=	mapping Jacobian matrix	$\vec{v}_G$	=	grid velocity, $\partial\vec{x}/\partial t$
g	=	determinant of Jacobian matrix, $det(\mathcal{G})$			

If we define the reference-domain flux, $\vec{\mathbf{F}}_X$, to be

$$\vec{\mathbf{F}}_X = \vec{\mathbf{F}}^i_X - \vec{\mathbf{F}}^v_X, \qquad \vec{\mathbf{F}}^i_X = g\mathcal{G}^{-1}\left(\vec{\mathbf{F}}^i(\mathbf{u}) - \mathbf{u}\vec{v}_G\right), \qquad \vec{\mathbf{F}}^v_X = g\mathcal{G}^{-1}\vec{\mathbf{F}}^v(\mathbf{u}, \nabla\mathbf{u}),$$

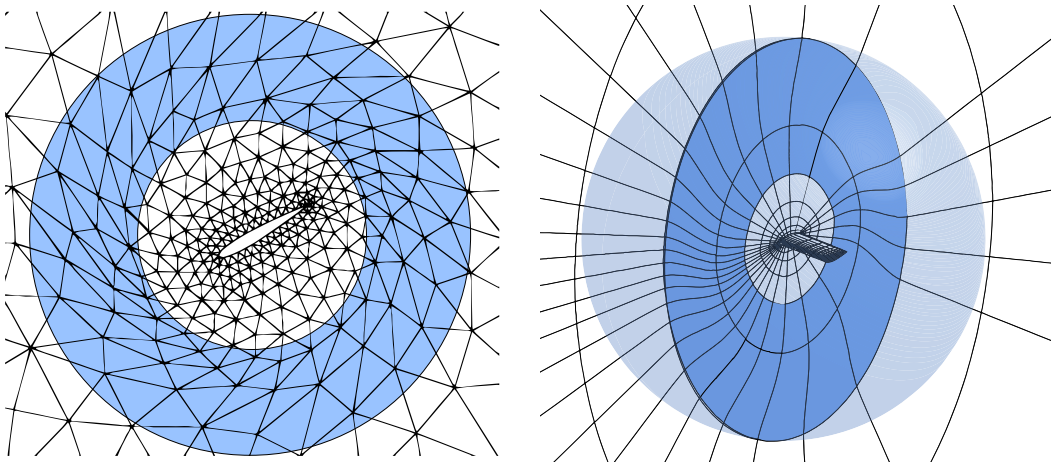
where $\vec{\mathbf{F}}^i_X$ and $\vec{\mathbf{F}}^v_X$ are the inviscid and viscous components, respectively, then the new equations to be satisfied are simply

$$\left.\frac{\partial \mathbf{u}_X}{\partial t}\right|_X + \nabla_X \cdot \vec{\mathbf{F}}_X(\mathbf{u}_X, \nabla_X \mathbf{u}_X) = \mathbf{0}. \tag{2}$$

Implementation of these equations involves the introduction of grid velocity terms into the boundary conditions and the inviscid flux function, as well as modifications to the viscous discretization. The viscous modifications entail the inclusion of a $\partial g/\partial X$ term in the flux, which arises when the product rule is applied to $\nabla\mathbf{u}$. However, once these steps are taken, Eqn. 2 represents a standard conservation law on a fixed domain, and the solution procedure is the same as if the mesh were not deforming at all.

2.2. Analytical Mesh Motion

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 [28], a polynomial blending function is a simple way 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 2 shows an example of this blending for an airfoil and a wing.



(a) (b)

Figure 2: Airfoil (a) and wing (b) 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.

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 Eqn. 2 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 [28]:

$$\frac{\partial \bar{g}}{\partial t} - \nabla_X \cdot (g\mathcal{G}^{-1}\vec{v}_G) = 0. \tag{3}$$

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

$$\left.\frac{\partial \mathbf{u}_{\bar{X}}}{\partial t}\right|_X + \nabla_X \cdot \vec{\mathbf{F}}_{\bar{X}}(\mathbf{u}_{\bar{X}}, \nabla_X \mathbf{u}_{\bar{X}}, \bar{g}) \quad = \quad 0, \tag{4}$$

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

Figure 3 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 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

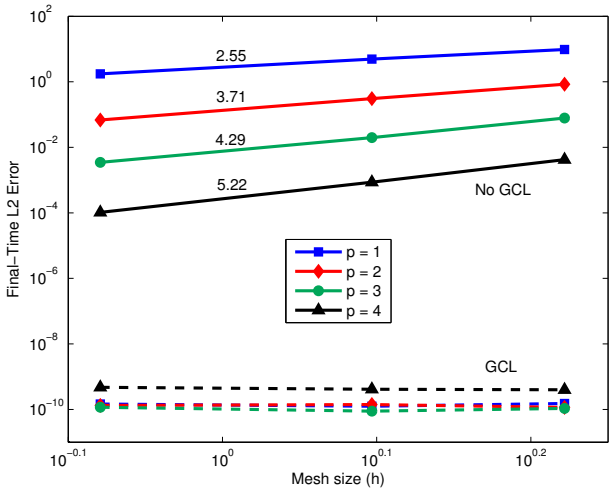


Figure 3: Free-stream errors with and without the GCL. A large number of time steps is used so that the spatial error dominates the temporal error in each case.

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 Eqn. 3 has introduced an additional numerical quantity, $\bar{g}$, which is subject to discretization errors just like the state quantities. This will be relevant for error estimation later on.

4. Primal Discretization

To discretize the state and GCL equations (Eqns. 4 and 3), we use a discontinuous Galerkin finite element method in both space and time. Each element has a spatial approximation order p, which can differ between elements and vary dynamically in time. Temporally, the discretization consists of time slabs on which the 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.

4.1. Approximation

Each space-time element is identified by two indices, (e, k), where e is the spatial element index and k is the time slab index. On each element, the state and GCL variable are approximated as

$$\mathbf{u}_{\bar{X},H}(\vec{X},t)\Big|_{e,k} = \underbrace{\mathbf{u}^{k,n}_{\bar{X},H,e,j}}_{\in\mathbb{R}^s}\ \underbrace{\phi^k_{H,e,j}(\vec{X})}_{\text{order } p_{e,k}}\ \underbrace{\varphi^n_H(t)}_{\text{order r}}, \tag{5}$$

$$\bar{g}_H(\vec{X},t)\Big|_{e,k} = \underbrace{\bar{g}^{k,n}_{H,e,j}}_{\in\mathbb{R}^1}\ \underbrace{\phi^k_{H,e,j}(\vec{X})}_{\text{order } p_{e,k}}\ \underbrace{\varphi^n_H(t)}_{\text{order r}}, \tag{6}$$

where s is the number of governing equations, $1 \le j \le \text{dof}(p_{e,k})$ is the spatial degree-of-freedom index on element (e,k), and $1 \le n \le r+1$ is the temporal degree-of-freedom index on slab k. The number of spatial degrees of freedom depends on both the approximation basis and dimension; for example, for full-order approximation in two dimensions, $\text{dof}(p_{e,k}) = (p_{e,k}+1)(p_{e,k}+2)/2$, where $p_{e,k}$ is the interpolation order. In this work, we use Lagrange bases in both space and time, with the spatial basis functions $\phi^k_{H,e,j}(\vec{X})$ specific to a given element and time slab, and the temporal basis functions $\varphi^n_H(t)$ the same for each time slab. Finally, the subscript H on the above terms just indicates that these quantities are defined on our current ("coarse") space-time mesh. When discussing error estimation later on, we will be dealing with a fine mesh as well, which will be denoted with a lowercase h.

For compactness of notation, we lump all spatial degrees of freedom associated with time node n on slab k into one vector, for both the state and the GCL variable,

$$\mathbf{U}^{k,n}_H = \left\{ \mathbf{u}^{k,n}_{\bar{X},H,e,j} \right\}_{\forall e,j} \in \mathbb{R}^{sN^k_H} \qquad \text{and} \qquad \mathbf{G}^{k,n}_H = \left\{ \bar{g}^{k,n}_{H,e,j} \right\}_{\forall e,j} \in \mathbb{R}^{N^k_H}, \tag{7}$$

where $N^k_H = \sum_e \text{dof}(p_{e,k})$ is the total number of spatial degrees of freedom on time slab k. As a shorthand, we will denote by $\mathbf{U}^k_H$ and $\mathbf{G}^k_H$ 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 $\mathbf{U}_H$ and $\mathbf{G}_H$.

4.2. Residuals

A nonlinear system of equations on each time slab is obtained by substituting the approximations from Eqns. 5 and 6 into Eqns. 4 and 3, 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. Thus, we take the equations

$$\left.\begin{aligned} \int_{T_k}\int_{\Omega_{e,k}} \varphi^n_H(t)\,\phi_{H,j}(\vec{X}) \left[\left.\frac{\partial \mathbf{u}_{\bar{X}}}{\partial t}\right|_X + \nabla_X \cdot \vec{\mathbf{F}}_{\bar{X}}\left(\mathbf{u}_{\bar{X}}, \nabla_X \mathbf{u}_{\bar{X}}, \bar{g}\right) \right] d\Omega\, dt &= 0 \\ \int_{T_k}\int_{\Omega_{e,k}} \varphi^n_H(t)\,\phi_{H,j}(\vec{X}) \left[\frac{\partial \bar{g}}{\partial t} - \nabla_X \cdot (g\mathcal{G}^{-1}\vec{v}_G) \right] d\Omega\, dt &= 0 \end{aligned}\right\} \quad \begin{array}{l} 1 \le j \le \text{dof}(p_{e,k}) \\ 1 \le n \le r+1 \end{array} \tag{8}$$

(where $\Omega_{e,k}$ and T_k represent a given spatial element and time slab, respectively) and express them in terms of the following discrete $r+1$ residual vectors on each time slab k:

$$\begin{aligned} \textit{State residual:} \qquad \overline{\mathbf{R}}^{k,m}_{\mathbf{U},H} &\equiv a^{m,n}\,\mathbf{M}^{k,k}_H \mathbf{U}^{k,n}_H - \varphi^m_H(t_{k-1})\,\mathbf{M}^{k,k-1}_H \mathbf{U}^{k-1,r+1}_H + \int_{t_{k-1}}^{t_k} \varphi^m_H(t)\,\mathbf{R}_{\mathbf{U},H}\left(\mathbf{U}^k_H(t), \mathbf{G}^k_H(t)\right) dt = \mathbf{0}, \\ \textit{GCL residual:} \qquad \overline{\mathbf{R}}^{k,m}_{\mathbf{G},H} &\equiv a^{m,n}\,\mathbf{M}^{k,k}_H \mathbf{G}^{k,n}_H - \varphi^m_H(t_{k-1})\,\mathbf{M}^{k,k-1}_H \mathbf{G}^{k-1,r+1}_H + \int_{t_{k-1}}^{t_k} \varphi^m_H(t)\,\mathbf{R}_{\mathbf{G},H}(t)\, dt = \mathbf{0}, \end{aligned} \tag{9}$$

where $1 \le m \le r+1$, and the $a^{m,n}$ are constant coefficients specific to the time scheme (see Appendix A). The temporal approximations of the state and GCL variable on time slab k are given by $\mathbf{U}^k_H(t) = \sum_n \mathbf{U}^{k,n}_H \varphi^n_H(t)$ and $\mathbf{G}^k_H(t) = \sum_n \mathbf{G}^{k,n}_H \varphi^n_H(t)$, respectively, and the spatial state and GCL residuals lie in $\mathbf{R}_{\mathbf{U},H} \in \mathbb{R}^{sN^k_H}$ and $\mathbf{R}_{\mathbf{G},H} \in \mathbb{R}^{N^k_H}$. Note that the GCL residual is independent of the state, while conversely, the state residual depends on both the state and GCL variable. This coupling will be reflected in the adjoint system derived later.

The flux functions used in the spatial discretization are Roe's inviscid flux [30] and the second form of Bassi and Rebay (BR2) viscous flux [29]. Finally, the $\mathbf{M}^{k,l}_H$ terms in Eqn. 9 refer to the mass matrices formed from the spatial basis functions on neighboring time slabs k and l, which will generally differ due to dynamic order refinement. We slightly abuse matrix-vector product notation in Eqn. 9 since $\mathbf{U}^{k,n}_H$ and $\mathbf{G}^{k,n}_H$ are of different size – for discretized systems the mass matrix is understood to act on each equation separately.

4.3. Implementation

Since the state residual depends on the GCL variable, the GCL must be advanced in time before advancing the state. This is a small cost due to the simplicity of the GCL residual and the fact that $\bar{g}_H$ is a scalar field. In addition, because of the nonlinear nature of the ALE mapping, we increase the default numerical quadrature order used to evaluate the spatial integrals in $\mathbf{R}_{\mathbf{U},H}$ and $\mathbf{R}_{\mathbf{G},H}$. Specifically, we use an order of $2p+5$ rather than the standard order of $2p+1$. Finally, we note that the unsteady solver used in this work is an iterative technique based on an inexact linearization of the unsteady residuals, so that the residual Jacobian costs do not exceed those of a steady solve [31, 24].

5. Output Error Estimation and Adjoint Formulation

Above, we described the technique for simulating problems on deforming domains. Our main purpose is then to answer: for an output like lift or drag at a given time, (i) what is the error in this output computed with our current space-time mesh, and (ii) how can we adapt the mesh to efficiently reduce this error?

Fundamentally, errors in an output arise due to discretization errors, which are a consequence of solving the governing equations in a finite-dimensional space. Typically, outputs of interest such as lift or drag are functions of the physical state, $\mathbf{u}$, which means errors in these outputs are directly related to errors in the physical state. However, from the transformations defined in Section 2, the physical state is a function of both the reference state $\mathbf{u}_{\bar{X}}$ and the GCL variable $\bar{g}$. Therefore, for simulations satisfying the GCL, the error in a given output will be related to errors in *both* $\mathbf{u}_{\bar{X}}$ and $\bar{g}$.

To estimate the error in an output J_H computed on our current mesh (denoted by subscript H), we consider the value of the output on a finer space-time mesh, denoted by subscript h. This fine-space output J_h, were it known, could be expanded about the coarse-space solution with a truncated Taylor series as follows:

$$J_h(\mathbf{U}_h, \mathbf{G}_h) \approx \underbrace{J_h(\mathbf{U}_h^H, \mathbf{G}_h^H)}_{\approx J_H} + \left.\frac{\partial J_h}{\partial \mathbf{U}_h}\right|_{(\mathbf{U}_h^H, \mathbf{G}_h^H)} \delta\mathbf{U} + \left.\frac{\partial J_h}{\partial \mathbf{G}_h}\right|_{(\mathbf{U}_h^H, \mathbf{G}_h^H)} \delta\mathbf{G}\,, \tag{10}$$

$$\Rightarrow \quad J_h - J_H \approx \left.\frac{\partial J_h}{\partial \mathbf{U}_h}\right|_{(\mathbf{U}_h^H, \mathbf{G}_h^H)} \delta\mathbf{U} + \left.\frac{\partial J_h}{\partial \mathbf{G}_h}\right|_{(\mathbf{U}_h^H, \mathbf{G}_h^H)} \delta\mathbf{G}\,. \tag{11}$$

Equation 11 gives an estimate of the output error $J_h - J_H$ between fine and coarse spaces. Here, $\mathbf{U}_h^H$ and $\mathbf{G}_h^H$ are just injections of the coarse solution $(\mathbf{U}_H,\ \mathbf{G}_H)$ into the fine space, while the perturbations $\delta\mathbf{U} = \mathbf{U}_h - \mathbf{U}_h^H$ and $\delta\mathbf{G} = \mathbf{G}_h - \mathbf{G}_h^H$ are the differences between fine and coarse solutions, which are at this point unknown.

Next, if we assume the fine-space equations are satisfied, our set of state and GCL residuals over the whole space-time domain, $\overline{\mathbf{R}}_{\mathbf{U},h}$ and $\overline{\mathbf{R}}_{\mathbf{G},h}$, must be zero. Just as we did with J, we can expand these residuals about the coarse-space solution to get

$$\overline{\mathbf{R}}_{\mathbf{U},h}(\mathbf{U}_h, \mathbf{G}_h) = 0 \approx \overline{\mathbf{R}}_{\mathbf{U},h}(\mathbf{U}_h^H, \mathbf{G}_h^H) + \left.\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{U}_h}\right|_{(\mathbf{U}_h^H, \mathbf{G}_h^H)} \delta\mathbf{U} + \left.\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{G}_h}\right|_{(\mathbf{U}_h^H, \mathbf{G}_h^H)} \delta\mathbf{G}\,, \tag{12}$$

$$\overline{\mathbf{R}}_{\mathbf{G},h}(\mathbf{G}_h) = 0 \approx \overline{\mathbf{R}}_{\mathbf{G},h}(\mathbf{G}_h^H) + \left.\frac{\partial \overline{\mathbf{R}}_{\mathbf{G},h}}{\partial \mathbf{G}_h}\right|_{\mathbf{G}_h^H} \delta\mathbf{G}\,. \tag{13}$$

Due to discretization errors, the solution on the coarse mesh will generally not satisfy the fine-space equations, and hence $\overline{\mathbf{R}}_{\mathbf{U},h}(\mathbf{U}_h^H, \mathbf{G}_h^H)$ and $\overline{\mathbf{R}}_{\mathbf{G},h}(\mathbf{G}_h^H)$ will be nonzero.

From Eqn. 13 we can obtain an expression for $\delta\mathbf{G}$, which we can then use in Eqn. 12 to obtain $\delta\mathbf{U}$. Doing so gives

$$\delta\mathbf{G} \approx -\left[\frac{\partial \overline{\mathbf{R}}_{\mathbf{G},h}}{\partial \mathbf{G}_h}\right]^{-1} \overline{\mathbf{R}}_{\mathbf{G},h}(\mathbf{G}_h^H)\,, \tag{14}$$

$$\delta\mathbf{U} \approx -\left[\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{U}_h}\right]^{-1} \left(\overline{\mathbf{R}}_{\mathbf{U},h}(\mathbf{U}_h^H, \mathbf{G}_h^H) - \frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{G}_h}\left[\frac{\partial \overline{\mathbf{R}}_{\mathbf{G},h}}{\partial \mathbf{G}_h}\right]^{-1} \overline{\mathbf{R}}_{\mathbf{G},h}(\mathbf{G}_h^H)\right)\,, \tag{15}$$

where all Jacobian matrices and their inverses are evaluated using the $(\mathbf{U}_h^H,\ \mathbf{G}_h^H)$ states injected from the coarse mesh. Since Eqns. 14 and 15 are computable from the coarse solution alone, we could in theory calculate $\delta\mathbf{U}$ and $\delta\mathbf{G}$ directly, then insert them into Eqn. 11 to obtain the output error estimate. However, there is an issue with this. While it would provide us with a total output error estimate, it would not tell us *where* in the mesh that output error originated from. Since we are interested in adapting the mesh to reduce the error, this is a problem, since it would effectively leave us blind.

The solution to this problem can be found by inserting Eqns. 14 and 15 into Eqn. 11 and grouping all terms multiplying the residual perturbations:

$$J_h - J_H \approx \underbrace{-\frac{\partial J_h}{\partial \mathbf{U}_h}\left[\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{U}_h}\right]^{-1}}_{\boldsymbol{\Psi}_{\mathbf{U},h}^T} \overline{\mathbf{R}}_{\mathbf{U},h}(\mathbf{U}_h^H, \mathbf{G}_h^H) \; + \; \underbrace{\left\{\frac{\partial J_h}{\partial \mathbf{U}_h}\left[\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{U}_h}\right]^{-1}\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{G}_h} - \frac{\partial J_h}{\partial \mathbf{G}_h}\right\}\left[\frac{\partial \overline{\mathbf{R}}_{\mathbf{G},h}}{\partial \mathbf{G}_h}\right]^{-1}}_{\boldsymbol{\Psi}_{\mathbf{G},h}^T} \overline{\mathbf{R}}_{\mathbf{G},h}(\mathbf{G}_h^H). \tag{16}$$

We can now define the quantity multiplying the state residual perturbation as the state adjoint $\boldsymbol{\Psi}_{\mathbf{U},h}^T$, and the quantity multiplying the GCL residual perturbation as the GCL adjoint $\boldsymbol{\Psi}_{\mathbf{G},h}^T$. These adjoint vectors represent the sensitivity of the output J to perturbations in the state and GCL residuals, respectively. By rearranging the terms in the definition of the adjoints, we see that the following equations must hold:

$$\text{State Adjoint Equation:} \qquad \left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{U}_h}\right)^T \boldsymbol{\Psi}_{\mathbf{U},h} + \left(\frac{\partial J_h}{\partial \mathbf{U}_h}\right)^T = 0, \tag{17}$$

$$\text{GCL Adjoint Equation:} \qquad \left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{G},h}}{\partial \mathbf{G}_h}\right)^T \boldsymbol{\Psi}_{\mathbf{G},h} + \left(\frac{\partial J_h}{\partial \mathbf{G}_h}\right)^T = -\left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}}{\partial \mathbf{G}_h}\right)^T \boldsymbol{\Psi}_{\mathbf{U},h}. \tag{18}$$

Notice that we have two linear systems for $\boldsymbol{\Psi}_{\mathbf{U},h}$ and $\boldsymbol{\Psi}_{\mathbf{G},h}$, which can be solved via the same iterative method used for the primal problem. Furthermore, once obtained, $\boldsymbol{\Psi}_{\mathbf{U},h}$ and $\boldsymbol{\Psi}_{\mathbf{G},h}$ provide the sensitivity of J to residual perturbations at specific locations in the mesh, which is essential for driving adaptation. Finally, the output error estimate, defined as $\delta J_{est} = J_H - J_h$, is readily computable from the adjoints as

$$\delta J_{est} \approx -\boldsymbol{\Psi}_{\mathbf{U},h}^T \overline{\mathbf{R}}_{\mathbf{U},h}(\mathbf{U}_h^H, \mathbf{G}_h^H) - \boldsymbol{\Psi}_{\mathbf{G},h}^T \overline{\mathbf{R}}_{\mathbf{G},h}(\mathbf{G}_h^H). \tag{19}$$

Thus, by solving for the state and GCL adjoints on a finer mesh, we can obtain both an adaptive indicator and an estimate for the output error using only our original solution on the coarse mesh.

5.1. Some notes on the GCL adjoint

For static simulations (or those not employing a GCL), the state adjoint would provide all relevant sensitivity information, and the GCL adjoint would not exist. For deforming domain problems satisfying the GCL, the GCL adjoint relates errors made in the motion itself to errors in the output of interest. For example, the analytical $g(t)$ that describes the desired motion could have a sinusoidal variation in space and time, but $\bar{g}(t)$ will only approximate this variation with a polynomial. Therefore, on a given mesh, we may be sufficiently resolving the states $\mathbf{U}$, but could be getting an accurate answer for the wrong motion! Injecting the coarse $\mathbf{G}_H$ into the fine-space $\overline{\mathbf{R}}_{\mathbf{G},h}$ and weighting by $\boldsymbol{\Psi}_{\mathbf{G},h}$ tells us where the mesh should be refined to more accurately represent the true motion.

The form of the GCL adjoint equation (18) is also worth noting. We see that the state adjoint $\boldsymbol{\Psi}_{\mathbf{U},h}$ appears as a source term on the right hand side, while conversely, the state adjoint is independent of $\boldsymbol{\Psi}_{\mathbf{G},h}$ in Eqn. 17. This is due to the fact that the state residual depends on the GCL variable, but the GCL residual does not depend on the state. From an implementation standpoint, it means that on a given time slab the state adjoint must be computed before the GCL adjoint can be obtained.

5.2. Unsteady Adjoint Equations

Equations 17 and 18 represent the adjoint equations at a high level, with all space-time residuals and adjoints lumped into the $\overline{\mathbf{R}}$ and $\boldsymbol{\Psi}$ terms. To actually solve them, we need to consider the details of the space-time Jacobian. The sparsity pattern of the full space-time Jacobian for a DG-in-time discretization is shown in Figure 4. This pattern reveals the temporal dependencies of the residuals on the states, which enables us to write the following form of the adjoint equations:

$$\left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}^{k,m}}{\partial \mathbf{U}_h^{k,n}}\right)^T \boldsymbol{\Psi}_{\mathbf{U},h}^{k,m} + \left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}^{k+1,m}}{\partial \mathbf{U}_h^{k,n}}\right)^T \boldsymbol{\Psi}_{\mathbf{U},h}^{k+1,m} + \left(\frac{\partial J_h}{\partial \mathbf{U}_h^{k,n}}\right)^T = 0, \tag{20}$$

$$\left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{U},h}^{k,m}}{\partial \mathbf{G}_h^{k,n}}\right)^T \boldsymbol{\Psi}_{\mathbf{U},h}^{k,m} + \left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{G},h}^{k,m}}{\partial \mathbf{G}_h^{k,n}}\right)^T \boldsymbol{\Psi}_{\mathbf{G},h}^{k,m} + \left(\frac{\partial \overline{\mathbf{R}}_{\mathbf{G},h}^{k+1,m}}{\partial \mathbf{G}_h^{k,n}}\right)^T \boldsymbol{\Psi}_{\mathbf{G},h}^{k+1,m} + \left(\frac{\partial J_h}{\partial \mathbf{G}_h^{k,n}}\right)^T = 0, \tag{21}$$

Jacobian matrix

	$\mathbf{U}^1$	$\mathbf{G}^1$	$\mathbf{U}^2$	$\mathbf{G}^2$	$\mathbf{U}^3$	$\mathbf{G}^3$
$\mathbf{R}^1_{\mathbf{U}}$	•	•				
$\mathbf{R}^1_{\mathbf{G}}$		•				
$\mathbf{R}^2_{\mathbf{U}}$	•		•	•		
$\mathbf{R}^2_{\mathbf{G}}$		•		•		
$\mathbf{R}^3_{\mathbf{U}}$			•		•	•
$\mathbf{R}^3_{\mathbf{G}}$				•		•

Jacobian matrix transpose

	$\mathbf{R}^1_{\mathbf{U}}$	$\mathbf{R}^1_{\mathbf{G}}$	$\mathbf{R}^2_{\mathbf{U}}$	$\mathbf{R}^2_{\mathbf{G}}$	$\mathbf{R}^3_{\mathbf{U}}$	$\mathbf{R}^3_{\mathbf{G}}$
$\mathbf{U}^1$	•		•			
$\mathbf{G}^1$	•	•		•		
$\mathbf{U}^2$			•		•	
$\mathbf{G}^2$			•	•		•
$\mathbf{U}^3$					•	
$\mathbf{G}^3$					•	•

Figure 4: Sparsity patterns for the coupled state/GCL system in a DG-in-time discretization, shown for the first three time slabs. Each entry corresponds to a $\frac{\partial \overline{\mathbf{R}}}{\partial \mathbf{U}}$ or $\frac{\partial \overline{\mathbf{R}}}{\partial \mathbf{G}}$ matrix over the spatial domain.

where k represents the time slab index, and m and n index the temporal nodes within each slab. These equations are just a more explicit form of Eqns. 17 and 18, and are marched backward in time to obtain $\mathbf{\Psi}_{\mathbf{U},h}$ and $\mathbf{\Psi}_{\mathbf{G},h}$.

5.3. Adjoint Implementation

The adjoint equations above require several derivative terms, including residual Jacobians and output linearizations. In the current work, we perform all differentiation analytically, with the exception of $\partial \overline{\mathbf{R}}^{k,m}_{\mathbf{U},h} / \partial \mathbf{G}^{k,n}_h$, which we evaluate using finite differences for ease of implementation.

In solving the GCL adjoint equation (Eqn. 21), the derivatives of $\overline{\mathbf{R}}_{\mathbf{G},h}$ with respect to $\mathbf{G}$ are straightforward to obtain, as the result is just a mass matrix. Obtaining the derivative of the output J with respect to $\mathbf{G}^{k,n}_h$ is done by simply applying the chain rule on derivatives with respect to the physical state, which are already used in the state adjoint equation. In addition, when solving the adjoint equations, we use the entire time history of the primal state and GCL, 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 [8] or local-in-time adjoint solvers [12] may be considered.

5.4. Error Estimate Implementation

The error estimate in Eqn. 19 requires an evaluation of the fine-space unsteady residuals associated with the coarse solution, as well as the fine-space adjoints $\mathbf{\Psi}_{\mathbf{U},h}$ and $\mathbf{\Psi}_{\mathbf{G},h}$. In this work, when computationally feasible, we solve the fine-space adjoint equations to machine precision to minimize additional sources of error in our estimates. For complex problems, we smooth or reconstruct the coarse-space adjoints in order to minimize computational cost [32].

When Galerkin orthogonality holds, coarse-space approximations of the adjoints can be subtracted from the fine-space adjoints appearing in Eqn. 19. Theoretically this has no effect on δJ_{est}, but in practice it minimizes errors due to converging residuals only to a finite tolerance. We note however that care must be taken when using the BR2 viscous discretization, which does not exhibit Galerkin orthogonality for coarse-space solutions injected into an order-enriched fine space. This is due to an order-dependence of the BR2 stabilization terms. We employ a simple remedy [33], which consists of using the coarse-space orders to approximate the stabilization terms when evaluating the fine-space residuals. In addition, as quadrature effects tend to be more pronounced for simulations on deformable domains, we use the coarse-space quadrature rules in these fine-space residuals.

5.5. Error Localization

Since our aim is to adapt the mesh to reduce the output error, a global error estimate is not enough – we need to localize the error contributions to individual space-time elements in the mesh. This is done by noting that the output error estimate in Eqn. 19 can be written as a sum over all space-time elements,

$$\delta J_{\text{est}} = \sum_k \sum_e \varepsilon_{e,k}, \tag{22}$$

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

$$\varepsilon_{e,k} = \left(-\mathbf{\Psi}^m_{\mathbf{U},h}\right)^T \overline{\mathbf{R}}^m_{\mathbf{U},h}\left(\mathbf{U}^H_h\right)\Big|_{e,k} + \left(-\mathbf{\Psi}^m_{\mathbf{G},h}\right)^T \overline{\mathbf{R}}^m_{\mathbf{G},h}(\mathbf{G}^H_h)\Big|_{e,k}. \tag{23}$$

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 (implied by the repeated index above). The error indicator is then taken as the absolute value of this elemental contribution to the output error,

$$\text{error indicator} = \epsilon_{e,k} = \left|\varepsilon_{e,k}\right|.$$

This indicator identifies the space-time elements that contribute most to the output error. However, for adaptation purposes we require still more information – specifically, is the error on a given space-time element due primarily to the *spatial* or *temporal* discretization?

This information is obtained from a space-time anisotropy measure, which we calculate in the same manner as presented in [32, 26]. Specifically, we calculate the error anisotropy using separate projections of the fine-space adjoint onto semi-coarsened spatial and temporal spaces. The spatial and temporal error estimates for space-time element (e, k) are obtained by using these projected adjoints in Eqn. 23, resulting in separate $\varepsilon^{\text{space}}_{e,k}$ and $\varepsilon^{\text{time}}_{e,k}$ estimates. We then use the ratio of these values to estimate the fractions (β) of spatial and temporal error on element $(e,\ k)$ as

$$\beta^{\text{space}}_{e,k} = \frac{|\varepsilon^{\text{space}}_{e,k}|}{|\varepsilon^{\text{space}}_{e,k}| + |\varepsilon^{\text{time}}_{e,k}|}, \qquad \beta^{\text{time}}_{e,k} = 1 - \beta^{\text{space}}_{e,k}. \tag{24}$$

6. Spatial and Temporal Adaptation

With the above estimates of spatial and temporal error, we have the fundamental information needed for adaptation. The next step is to decide *what* to adapt. In this work, to address spatial errors, we will adapt the spatial order p on each element dynamically in time (see Figure 5 for a schematic). To address temporal errors, we will adapt the temporal grid by selectively reducing or increasing the time slab widths (while potentially adding or removing time slabs as necessary). So the "objects" to be adapted are both (i) individual space-time elements and (ii) time slabs. Adaptive

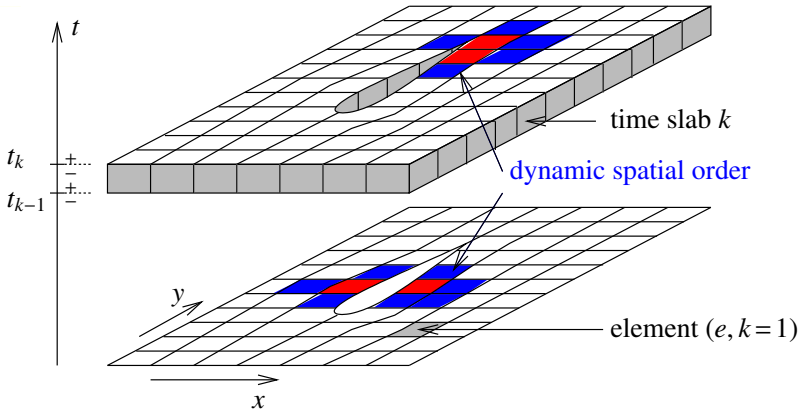


Figure 5: An illustration of dynamic p-refinement, in which spatial interpolation orders change in time to track relevant flow features.

indicators identifying the spatial error on each space-time element and the temporal error on each slab are given by

$$\text{spatial indicator on space-time element } e, k = \quad \epsilon^{\text{space}}_{e,k} = \epsilon_{e,k}\beta^{\text{space}}_{e,k}, \tag{25}$$

$$\text{temporal indicator on time slab } k = \quad \epsilon^{\text{time}}_{k} = \sum_{e} \epsilon_{e,k}\beta^{\text{time}}_{e,k}, \tag{26}$$

where the sum indexed by e is taken over all spatial elements on a given time slab. With these indicators, we can then conceivably lump all time slabs and space-time elements into the same "bin," rank them according to highest and lowest error indicators, and determine which to adapt based on their relative positions in that ranking.

However, from a computational perspective, this isn't quite what we want to do. The issue is that, if (for example) a time slab and a space-time element had the same error, they would be equal candidates for refinement. But refining an entire time slab generally results in a much larger increase in degrees of freedom than refining a single 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. 25 and 26) *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.

The above 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. As mentioned, the figure of merit is the amount of output error addressed, Eqns. 25 and 26, 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**
   (a) Set coarsening degree-of-freedom tally to zero.
   (b) Choose an unmarked space-time element or time slab with the lowest merit function.
   (c) 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.
   (d) 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.
   (e) If the tally meets or exceeds the coarsening budget, B^{coarsen}, stop. Otherwise return to step 2b.

3. **Refine**
   (a) Set refinement degree-of-freedom tally to zero.
   (b) Choose an unmarked space-time element or time slab with the highest merit function.
   (c) If a time slab was chosen, mark it for a factor of 0.5 refinement and add dof_k to the refinement tally.
   (d) 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.
   (e) If the refinement budget, B^{refine}, is met or exceeded, stop. Else, return to step 3b.

When we say "factor of 2 coarsening" (in 2c) and "factor of 0.5 refinement" (in 3c), 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. In fact, if *only* refinement occurs, this is exactly the case – 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. To perform this shuffling, time slabs are redistributed using one-dimensional metric-based meshing, the details of which are given below (with an additional schematic provided in Figure 6).

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.

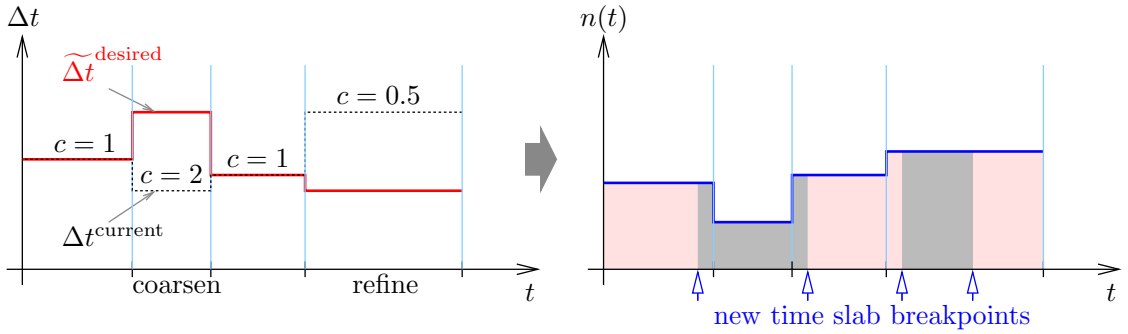


Figure 6: Demonstration of the one-dimensional remeshing algorithm used to define new time slab breakpoints.

The above procedure describes how output-based adaptation of the current space-time mesh is performed. For a given number of degrees of freedom, this adaptation represents our best effort at eliminating errors in the output of interest. During the overall adaptive process, several of these adaptations are performed, and the unsteady problem is solved on successively refined space-time meshes. 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.

6.1. Alternative Adaptive Methods

In the results section below, we compare our output-based adaptation to uniform h- and p-refinements in space (combined with bisection in time), as well as to a cheaper indicator based on the unweighted state residuals. The residual indicator is given by a form similar to the output error (Eqn. 23), but without the adjoint and with absolute values on the individual residual components:

$$\epsilon_{e,k}^{\text{res}} = \sum_m \left| \overline{\mathbf{R}}_{\mathbf{U},h}^m (\mathbf{U}_h^H) \right| \Big|_{(e,k)}.$$

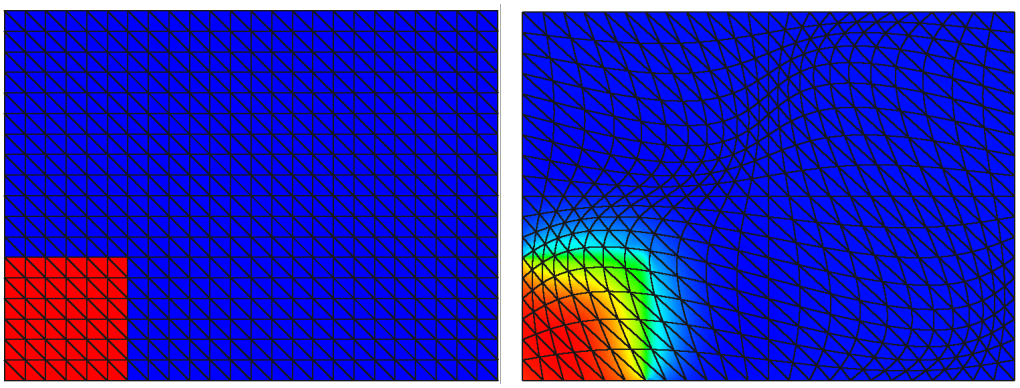
This indicator targets areas of the space-time domain where the governing equations are not well-satisfied, and is commonly used in practice as an adaptive metric. A jump-based space-time anisotropy measure [24] is used to determine $\beta_{e,k}^{\text{space}}$ and $\beta_{e,k}^{\text{time}}$ for this metric.

7. Results

In this section, we present the results for several test cases. The first is a verification of the adjoint and error estimation procedures described above. The following cases then employ this error indicator to drive unsteady mesh adaptation for problems of engineering interest. These problems consist of airfoils and wings pitching and plunging at low Reynolds number, and the convergence of the lift on these bodies is compared for the adaptive methods described above.

7.1. Error Estimate Verification

Here, we verify the proposed error estimation strategy with a simple Navier-Stokes problem. The problem consists of a density perturbation in a stagnant fluid, situated in the corner of a rectangular basin bounded by no-slip walls (Figure 7). The density perturbation is allowed to diffuse for two time units, and the final vertical force on the left wall is taken as our output of interest. The force is kept dimensional while using the following convenient units: density = 1, density perturbation = 0.25, total energy = 2.675, gas constant = 1, laminar viscosity = 1. One time unit then corresponds to $t = 1$ in the physical evolution of the equations.



(a) Density, $t = 0$ (b) Density, $t = 2$

Figure 7: Initial and final meshes and densities. The initial density perturbation is 25% above the nominal value.

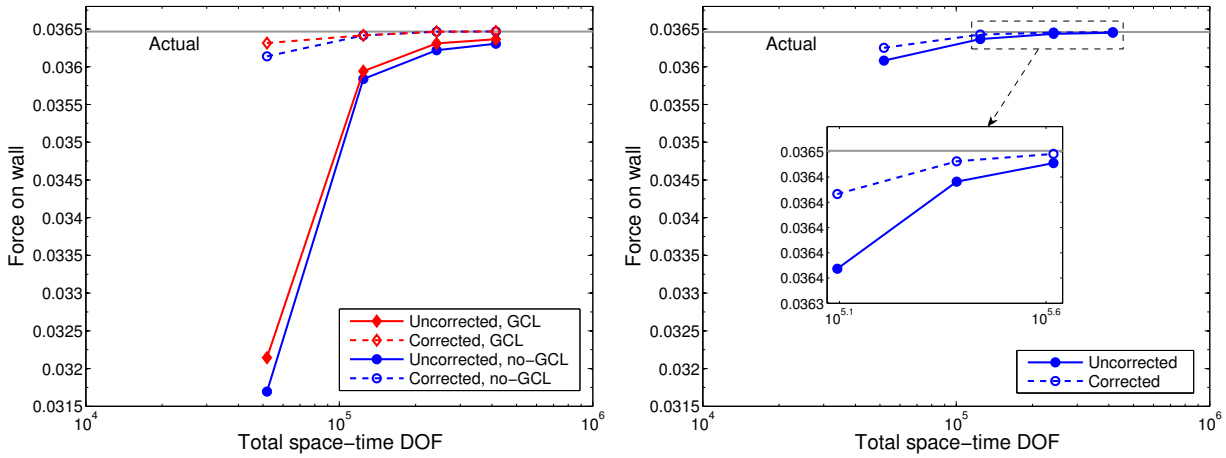
Since no boundaries are moving here, mesh motion is not required and the problem could be solved with a fixed grid. However, to test the error estimation with mesh motion, we instead choose to wave the mesh in the background, using the following formula [28] to map the domain from reference (X_1, X_2) to physical (x_1, x_2) space:

$$\begin{aligned} x_1 &= X_1 + 2.0 \sin\left(\frac{2\pi X_1}{20}\right) \sin\left(\frac{\pi X_2}{7.5}\right) \sin\left(\frac{2\pi t}{3}\right), \\ x_2 &= X_2 + 1.5 \sin\left(\frac{2\pi X_1}{20}\right) \sin\left(\frac{\pi X_2}{7.5}\right) \sin\left(\frac{4\pi t}{3}\right). \end{aligned} \tag{27}$$

Since this mapping is relatively violent, with large variations in g in both space and time, it should introduce motion-related errors and provide a good test of our error estimation procedure. Note that the movement of the mesh should in theory have no impact on the physics of the problem, and the solution should converge upon refinement to that with no mesh motion.

To verify the error estimation, three separate cases were run: (i) a no-motion case; (ii) a case with motion but no GCL; and (iii) a case with both motion and GCL. These cases were run at interpolation orders ranging from $p = 1$ to 4, with a DG1 time scheme and 10, 12, 14, and 16 time steps, respectively. For each run, the output J_H on the current mesh is first recorded. Next, error estimation based on a $(p + 1, r + 1)$ fine space is performed, and the predicted change in the output relative to the fine space (δJ_{est}) is computed. Finally, the primal problem is solved directly on the $(p + 1, r + 1)$ space, and the output J_h is determined. The difference $\delta J_{\text{act}} = J_H - J_h$ between coarse and fine outputs can then be obtained and compared to δJ_{est}. If the error estimation is working properly, these values should correspond closely. Of course, since the problem is nonlinear, the correspondence will generally not be exact.

Figure 8 shows both the output and corrected output ($J - \delta J_{est}$) convergence for all cases. Though the errors for the motion cases are significantly larger than those without motion, the error estimate brings the corrected output very close to the true value. Table 2 gives a more quantitative comparison between the actual and predicted errors. From the table, we see that the error estimates with mesh motion display 93-99% accuracy relative to the actual output errors. The no-motion error estimates, despite being relatively less accurate initially, also achieve greater than 95% accuracy



(a) With motion (b) Without motion

Figure 8: Output convergence for both motion and no-motion cases, shown with the same scale. Despite larger-magnitude errors, the corrected output converges rapidly to the true value for motion cases.

as the mesh is refined. The reason for the initial inaccuracy is not completely known, but the smaller magnitude of the no-motion errors means low-level noise due to (e.g.) numerical quadrature has a larger effect on the percent accuracy.

For the GCL runs specifically, the above results verify that the GCL adjoint and error estimation procedures were implemented correctly. The breakdown of errors due to the state and GCL individually are given in Table 3 (which includes an additional $p = 0$ run), and we see that the GCL-related errors constitute from 8-42% of the total error estimate.

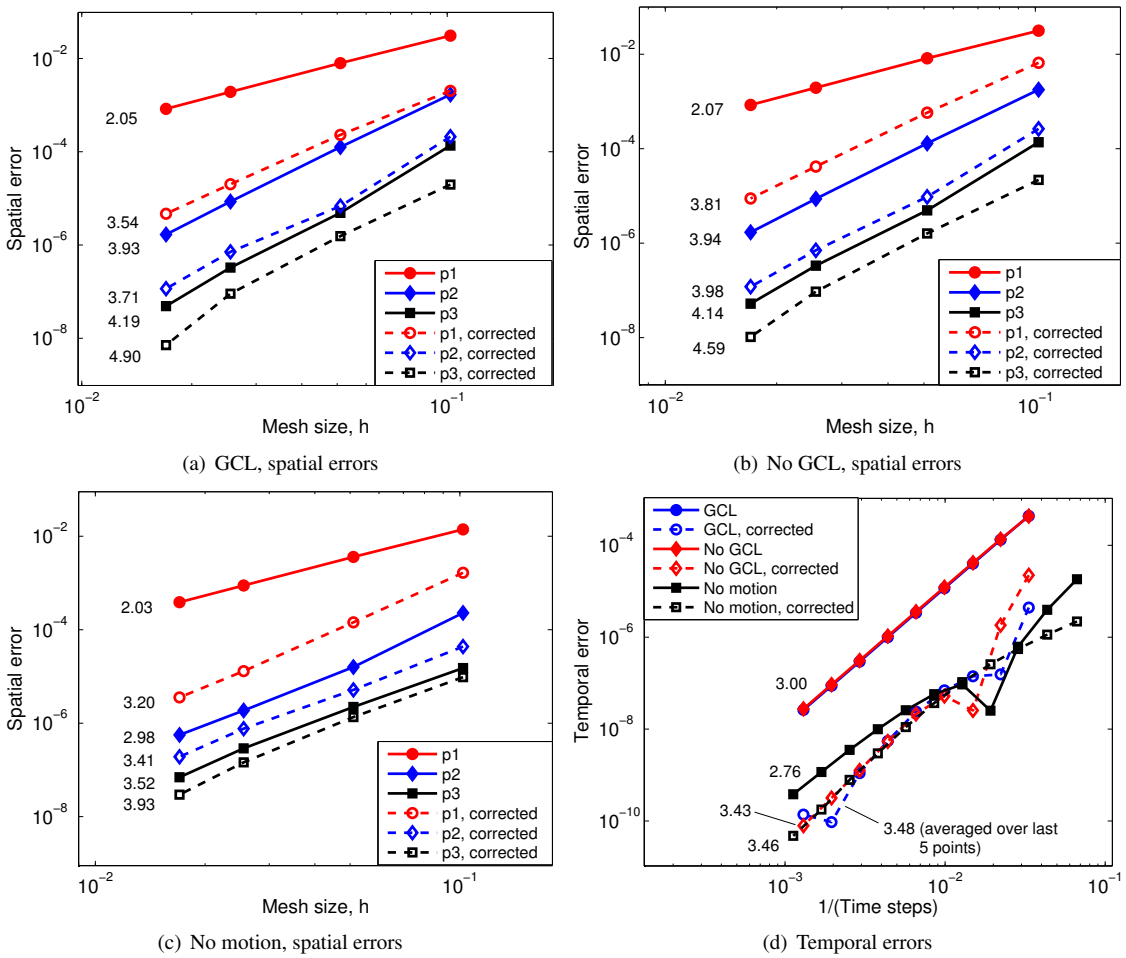
	GCL			**No GCL**			**No Motion**		
Run	δJ_{est}	δJ_{act}	% Error	δJ_{est}	δJ_{act}	% Error	δJ_{est}	δJ_{act}	% Error
$p = 1$	-4.170e-3	-4.020e-3	**3.73**	-4.443e-3	-4.489e-3	**1.03**	-1.687e-4	-2.882e-4	**41.44**
$p = 2$	-4.746e-4	-5.080e-4	**6.58**	-5.782e-4	-6.115e-4	**5.46**	-5.922e-5	-7.026e-5	**15.70**
$p = 3$	-1.551e-4	-1.543e-4	**0.49**	-2.448e-4	-2.438e-4	**0.40**	-1.610e-5	-1.555e-5	**3.51**
$p = 4$	-1.022e-4	-1.008e-4	**1.40**	-1.626e-4	-1.608e-4	**1.12**	-7.308e-6	-7.015e-6	**4.18**

Table 2: Relative accuracy of error estimates for motion and no-motion cases at different orders p. "% Error" denotes the error in δJ_{est} relative to δJ_{act}.

	$p = 0$	$p = 1$	$p = 2$	$p = 3$	$p = 4$
State Errors	3.593e-02	-4.607e-03	-5.930e-04	-2.526e-04	-1.675e-04
GCL Errors	2.633e-02	4.376e-04	1.184e-04	9.743e-05	6.525e-05
GCL % of Total	**42.29**	**8.67**	**16.64**	**27.84**	**28.03**

Table 3: Contribution of GCL and state errors to total error estimate.

While these results demonstrate the qualitative and quantitative accuracy of the error estimates, it is also worthwhile to look at formal convergence rates. To ensure that the singularities in the corners of the basin do not inhibit these rates, we change the problem slightly. The basin walls are replaced by periodic boundaries, and the perturbed region (which now consists of both density and velocity variations) is shifted so that it no longer touches the boundary. Since the walls have vanished, the output is taken to be the final-time integral of pressure over the domain.



(a) GCL, spatial errors

(b) No GCL, spatial errors

(c) No motion, spatial errors

(d) Temporal errors

Figure 9: Spatial and temporal convergence histories for the corrected and uncorrected final-time output. Average convergence rates are shown next to each curve, and expected rates are achieved for all runs. Note that different truth values are used for the GCL, no-GCL, and no-motion cases, so a direct comparison of error magnitudes should not be made.

We are interested in how the uncorrected and corrected outputs ($J - \delta J_{est}$) converge in both space and time. We first fix the number of time steps and perform error estimation in space only, relative to a $p+1$ fine space, for various values of p and several h-refinements of the spatial mesh. From these refinements, we can plot the spatial convergence rates for GCL, no-GCL, and no-motion runs (shown in Figure 9). For nearly all runs, we see the expected $p+1$ convergence rates for the uncorrected outputs and the expected $p+2$ rates for the corrected outputs (except for the $p=2$ motion runs, which show slightly higher rates). A rate of $p+2$ for the corrected output is anticipated because, ideally, the error estimate should correct the coarse-space output to the fine-space value, and the fine-space value itself converges at a rate of $p+2$.

Next, we fix the spatial grid and interpolation order (at $p=2$), and perform both error estimation and refinement in time, using a DG time scheme with order $r=1$. The results of this temporal convergence study are shown in the final plot of Figure 9. Since the output is computed at the final time, it lies on a downwind time node and is expected to superconverge [34] at a rate of $2r+1=3$. This rate was obtained, and is evident in the plot. Error estimation for these runs is performed relative to an $r+1$ space, and a convergence rate of $2(r+1)=4$ is expected for the corrected outputs (see Appendix B). The actual corrected outputs achieve a rate of approximately 3.5 before numerical precision causes a bottoming out.

7.2. Dynamic Mesh Adaptation for Pitching and Plunging Airfoils

With the performance of the error estimates confirmed, we next use them to drive mesh adaptation for a case of engineering interest – two airfoils pitching and plunging in series at low Reynolds number. 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 10. 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).

To compute this output, the adaptive methods described in Section 6 (output-based, residual, and uniform h- and p-refinement) were considered. 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.

7.2.1. Results

The output convergence for each adaptive method as a function of total space-time degrees of freedom is shown in Figure 11. 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 12 and 13, 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.

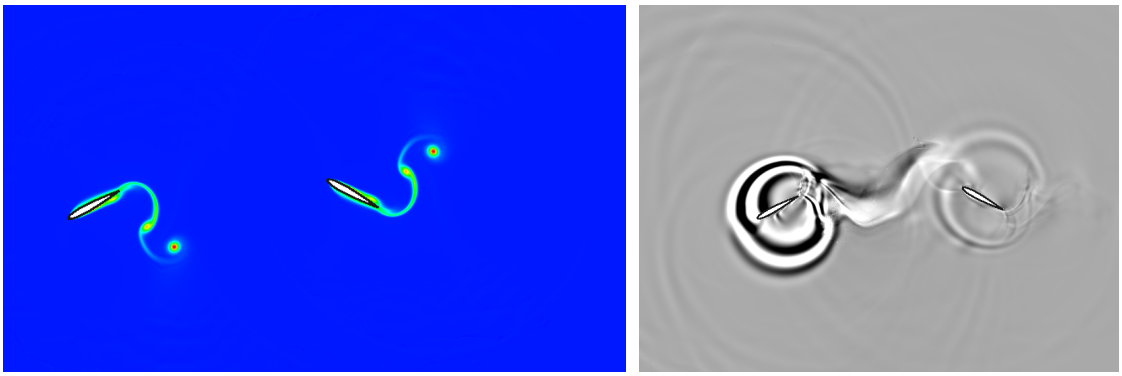
To highlight one of the factors driving the adaptation, contours of the GCL adjoint are shown alongside the entropy contours in Figure 10. The time $t = 0.70T$ is the instant before the initial vortex is shed, and the large sensitivity of the output to this event can be seen in the adjoint contours. 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.

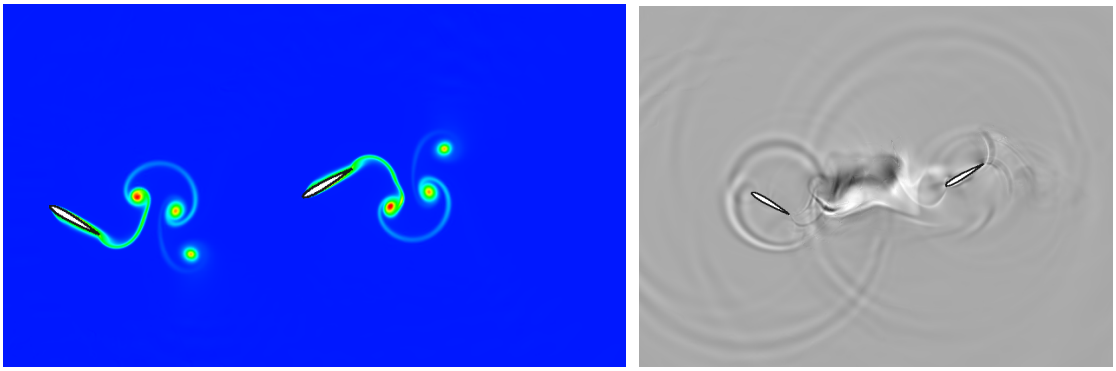
Finally, since the GCL adjoint and error estimates are of particular interest in this paper, a breakdown of the output error into its state and GCL components is provided in Table 4. From the table, we see that the GCL errors initially make up only a small percentage of the total error, but as the adaptations proceed, their contribution increases to over half of the total error estimate. This implies that *if* the GCL is used, a corresponding GCL adjoint is necessary to obtain an accurate error estimate.

7.2.2. CPU Time Comparison

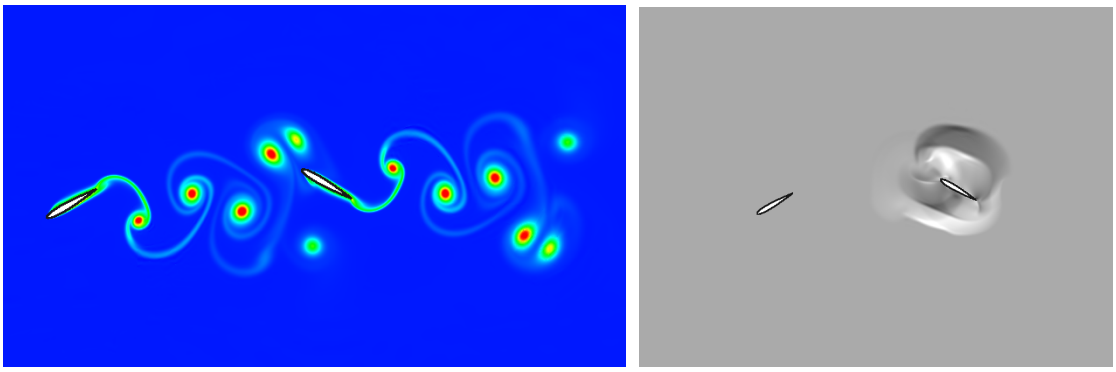
Above, we solved the fine-space adjoint to machine precision to ensure accurate error estimates and efficient allocation of mesh degrees of freedom. In practice, CPU time is another important factor, and solving the adjoint to



(a) $t = 0.70T$



(b) $t = 1.25T$



(c) $t = 2.75T$

Figure 10: 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.

machine precision is generally not the most efficient strategy. Previous work [32] has shown that a few smoothing iterations on the fine-space can provide acceptable error estimates at reduced computational cost. In Figure 14, 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

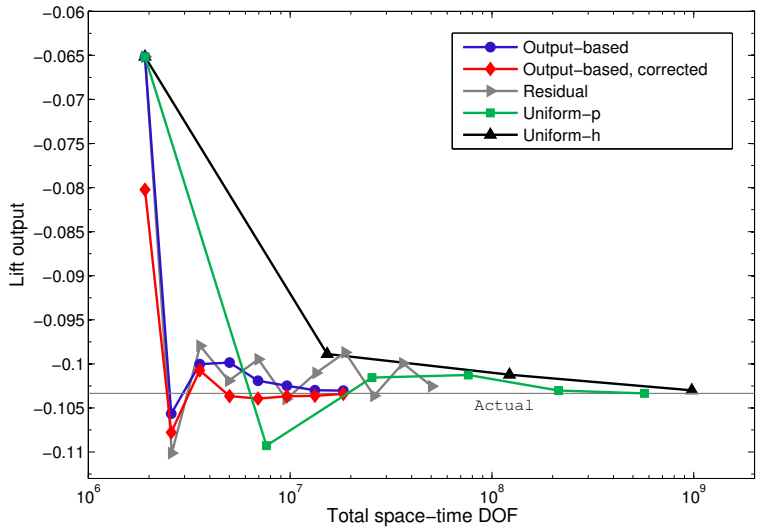


Figure 11: Two-airfoil case: Output convergence for various adaptive methods. The output-based method performs the best. Note that for all plots, the "actual" value is taken from the final uniform p-refinement.

	Adpt. 1	Adpt. 2	Adpt. 3	Adpt. 4	Adpt. 5	Adpt. 6	Adpt. 7	Adpt. 8
State Errors	1.52e-02	2.21e-03	7.45e-04	3.84e-03	2.37e-03	1.03e-03	3.62e-04	-5.79e-04
GCL Errors	-1.54e-04	-9.30e-05	-3.75e-05	-3.87e-05	-3.49e-04	1.68e-04	2.77e-04	9.53e-04
GCL % of Total	**1.00**	**4.04**	**4.79**	**1.00**	**12.84**	**13.99**	**43.36**	**62.19**

Table 4: Two-airfoil case: Contribution of GCL and state errors to the total error estimate for all output-based adaptations.

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.

7.2.3. Single Pitching Airfoil

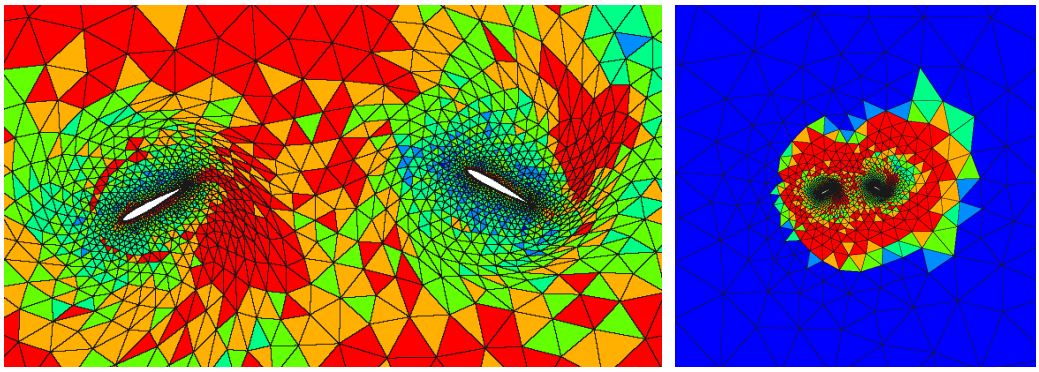
The two-airfoil case above was chosen in part to demonstrate how the output-based adaptation can track both acoustic and convective modes of error propagation. The reader may wonder whether the conclusions drawn above apply to a simpler, perhaps more realistic, case.

To answer this question, we remove one airfoil, and consider a single airfoil pitching back and forth. The airfoil undergoes three periods of motion, with a Strouhal number of 1.0, a Reynolds number of 400, and a free-stream Mach number of 0.2. We then use the same adaptive methods considered above, with the output taken to be the lift integrated over the final 2.5% of the simulation time. Entropy contours and the adapted spatial mesh near the end of the simulation are shown in Figure 15.

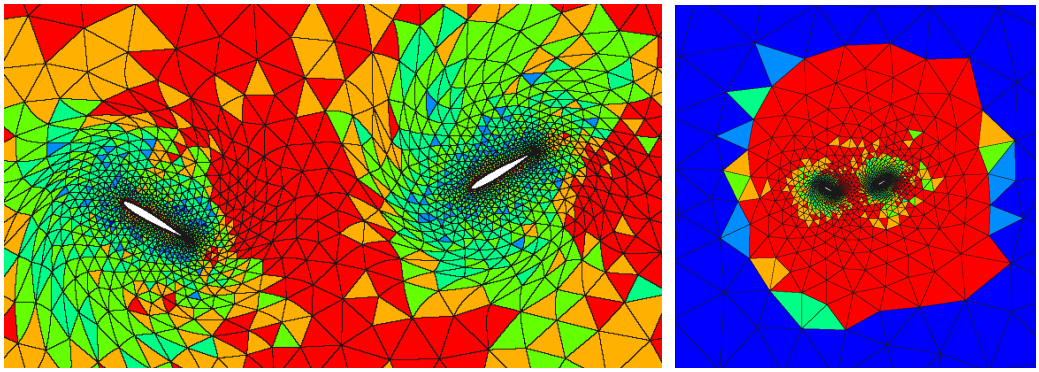
Output convergence is shown as a function of both degrees of freedom and wall time in Figure 16. Again, the output-based adaptation significantly outperforms all other methods in terms of degrees of freedom, and demonstrates savings in total run time as well. Just as before, the residual adaptation is oscillatory and fails to converge, being distracted by acoustic waves and errors made in the far field. Finally, a breakdown of the output error into its state and GCL components is shown in Table 5, and we see that the GCL makes up a significant percentage of the total – again underscoring the importance of including the GCL adjoint when the GCL is used.

7.3. Mesh Adaptation for a Three-Dimensional Flapping Wing

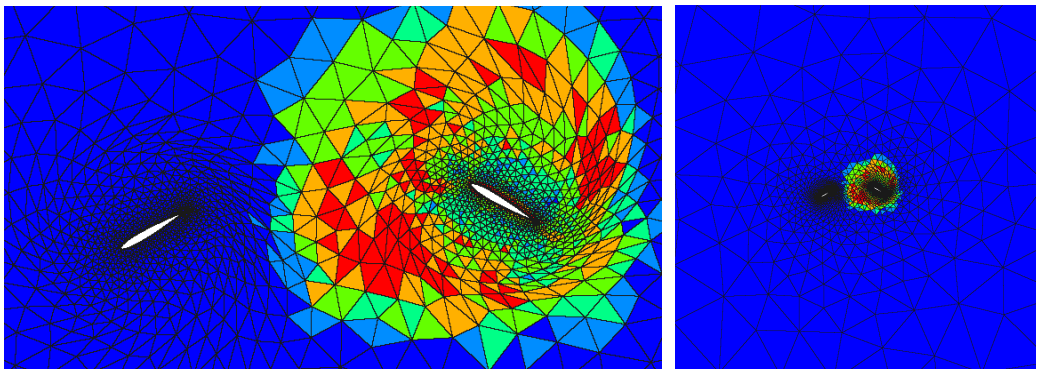
The above cases show the effectiveness of our output-based strategy in two dimensions. In three dimensions, the underlying ideas for mesh motion and adaptation remain the same, but the increased algorithmic complexity and



(a) $t = 0.70T$



(b) $t = 1.25T$



(c) $t = 2.75T$

Figure 12: Two-airfoil case: Output-adapted meshes at various stages of the motion. Blue is $p = 0$, red is $p = 5$.

	Adpt. 1	Adpt. 2	Adpt. 3	Adpt. 4	Adpt. 5	Adpt. 6	Adpt. 7	Adpt. 8
State Errors	9.87e-03	1.09e-02	3.82e-03	1.93e-03	6.62e-04	4.00e-04	4.49e-04	3.14e-04
GCL Errors	-1.26e-04	-2.75e-04	-4.54e-04	-4.87e-04	1.27e-04	-8.90e-05	-3.49e-04	-3.36e-04
GCL % of Total	**1.26**	**2.47**	**10.61**	**20.17**	**16.12**	**18.19**	**43.69**	**51.76**

Table 5: Single airfoil: Contribution of GCL and state errors to the total error estimate for all output-based adaptations.

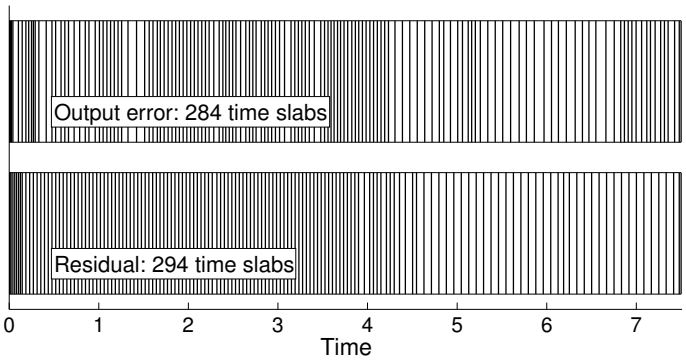


Figure 13: Two-airfoil case: Temporal grids from the seventh adaptation of both output-based and residual runs. For clarity, only every other time slab is plotted.

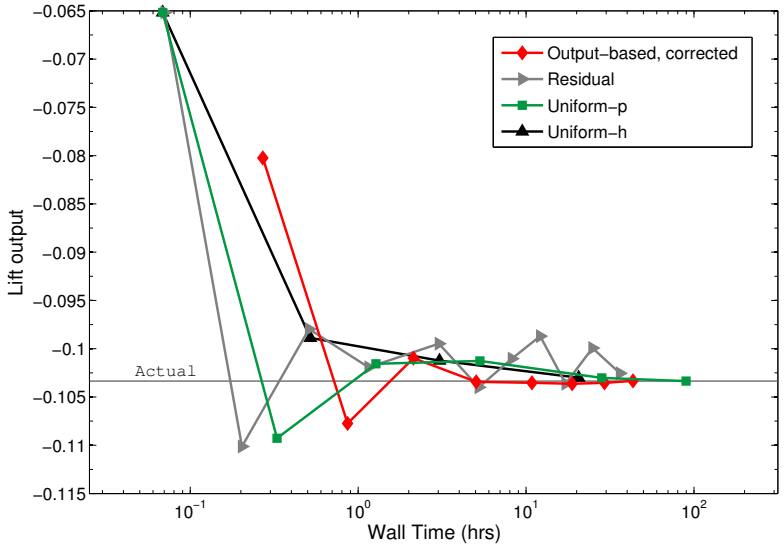
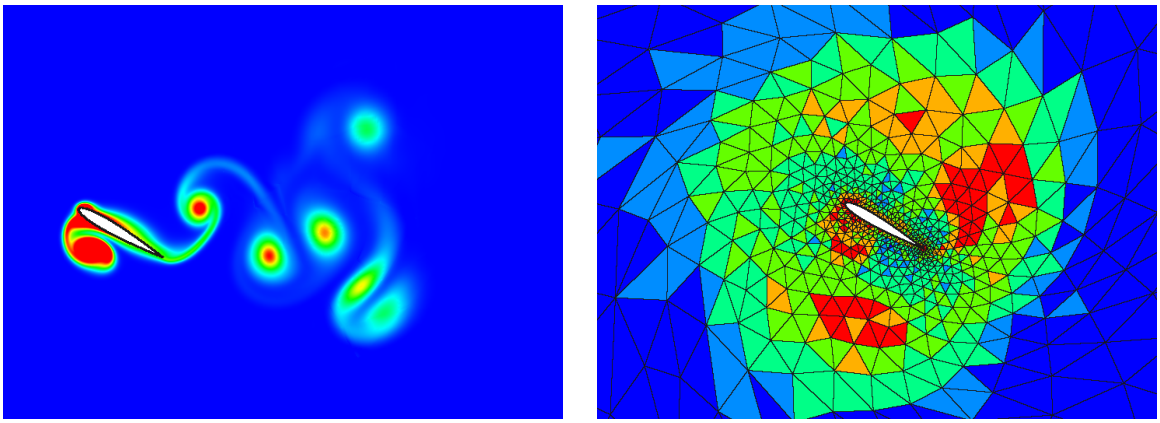


Figure 14: Two-airfoil case: Wall time comparison for the adaptive methods. The output-based method converges the fastest.

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.

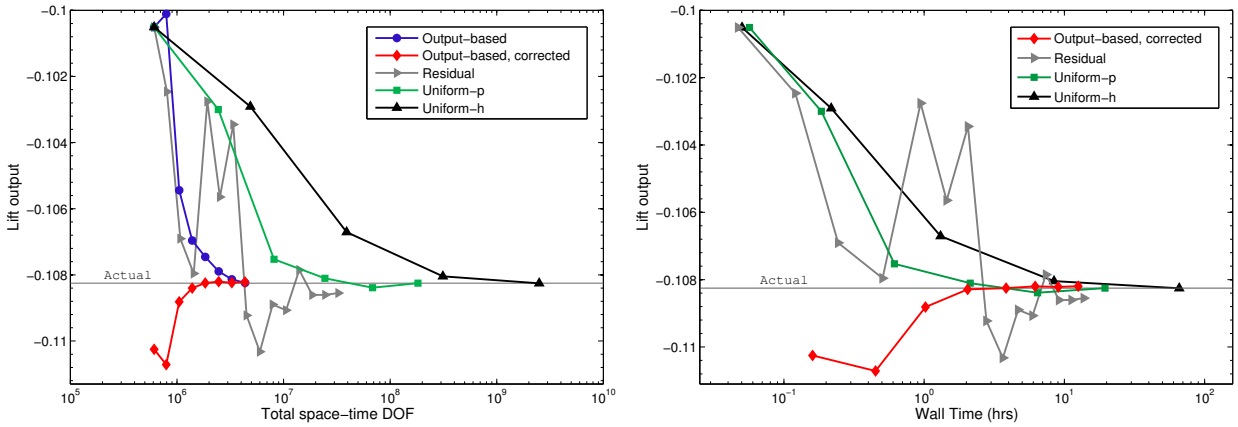
To address this issue, we discard the idea of computing the fine-space adjoint directly. Instead, we compute the adjoint in the same space as the primal problem, and then reconstruct it in both space and time to obtain an approximation to the fine-space adjoint. This reconstruction is performed locally, 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]. See Figure 17 for a depiction of these procedures.

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 18, 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.



(a) Entropy (b) Adapted spatial mesh

Figure 15: Single airfoil: Entropy contours and the corresponding adapted mesh at time $t = 2.75T$.



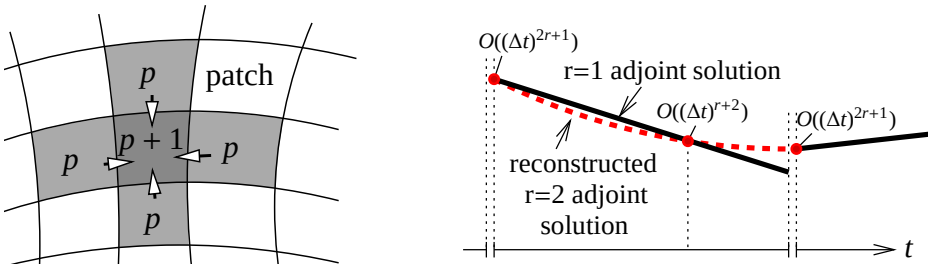
(a) (b)

Figure 16: Single airfoil: Output convergence as a function of (a) degrees of freedom and (b) total wall time.

Three periods of motion were simulated, and the solution at various times is shown in Figure 19. 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, with a 30% growth factor and 5% coarsening factor used for the output- and residual-based methods. As before, the spatial order p was constrained to lie between 0 and 5, and a DG1 scheme was used in time.

7.3.1. Results

The convergence for each adaptive method is shown in Figure 20, 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 20, and we see that the method also performs well in this context.



(a) Spatial reconstruction (b) Temporal reconstruction

Figure 17: Illustration of spatial and temporal adjoint reconstructions. (a) shows a patch of nearest-neighbor elements used for spatial high order reconstruction via least-squares interpolation. (b) shows 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 $r = 1$.

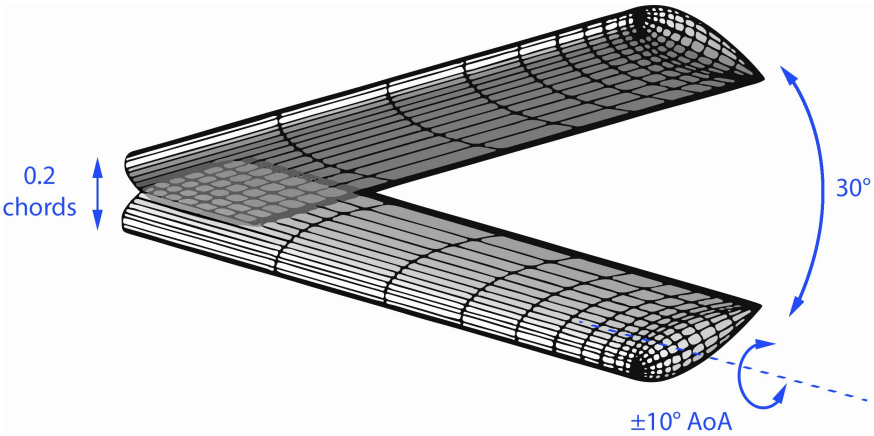
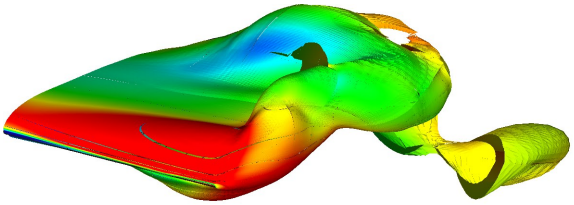


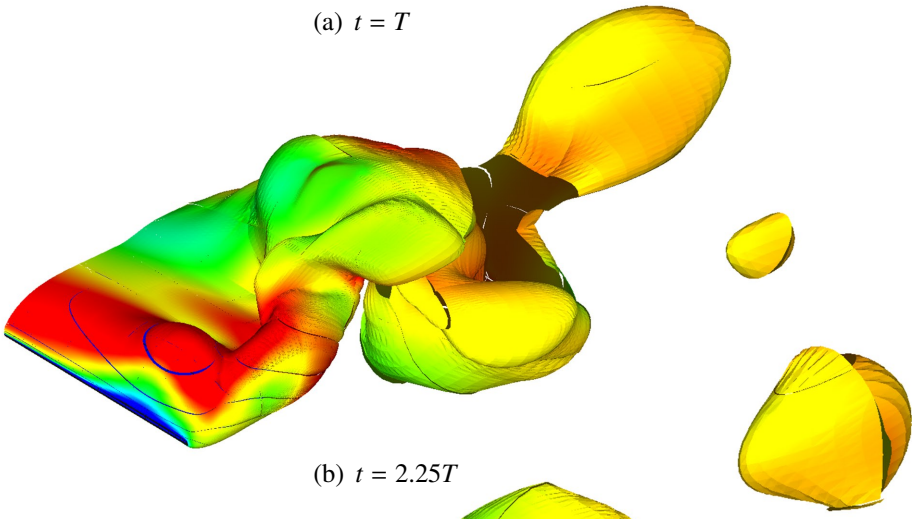
Figure 18: 3D wing: Schematic of the flapping motion. The flow regime is approximately that of a housefly.

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. In the 2D cases, the residual convergence was oscillatory, but at least hovered near the true output value. Here, the residual adaptation fails outright, and the output value does not converge at all from its initial value. The reason for this behavior is evident in Figure 21, 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 22, while adapted temporal grids are shown in Figure 21. 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.

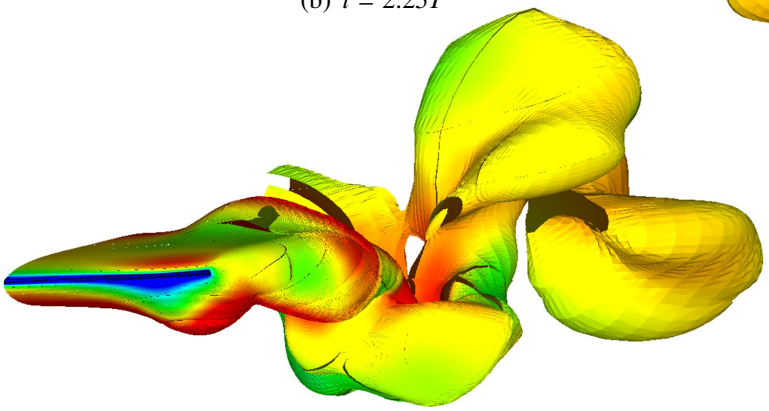
Finally, the state vs. GCL error breakdown for the output-based runs is shown in Table 6. As in 2D, the GCL errors make up a relatively large percentage of the total error estimate, again indicating the importance of the GCL adjoint. Overall, we thus find that both the performance of the adaptive algorithm and the conclusions drawn about the GCL extend from two to three dimensions.



(a) $t = T$



(b) $t = 2.25T$



(c) $t = 2.8T$

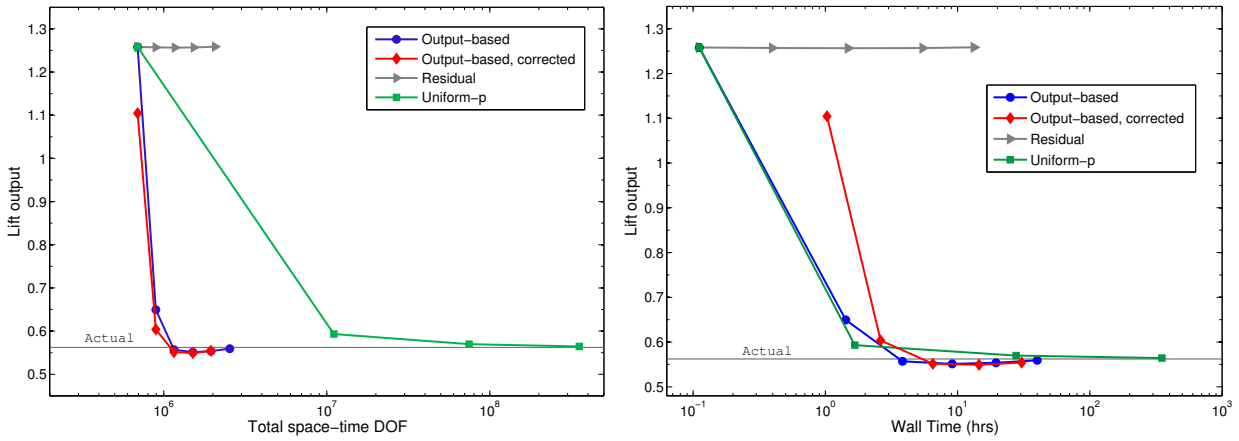
Figure 19: 3D wing: Mach contours projected onto entropy isosurfaces, shown for several stages of the flapping motion. The maximum Mach number (in red) is approximately 0.5. The images are taken from the final ($p=3$) uniform refinement.

	Adpt. 1	Adpt. 2	Adpt. 3	Adpt. 4	Adpt. 5
State Errors	1.68e-01	5.56e-02	1.98e-02	1.77e-02	1.83e-02
GCL Errors	-1.42e-02	-9.91e-03	-1.43e-02	-1.57e-02	-1.89e-02
GCL % of Total	**7.80**	**15.13**	**41.84**	**47.03**	**50.74**

Table 6: 3D wing: Contribution of state and GCL errors to the total error estimate for all output-based adaptations.

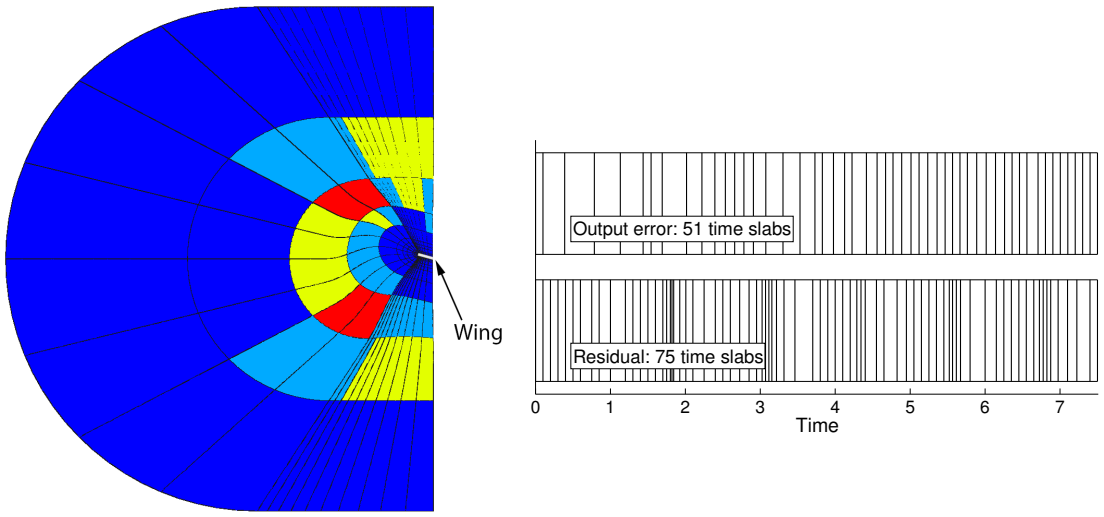
8. Relevance of the GCL

In the cases presented, we have shown that output-based error estimation and mesh adaptation lead to efficient output convergence in terms of both mesh size and computational time. For these cases, the GCL was used to ensure



(a) (b)

Figure 20: 3D wing: Output convergence as a function of (a) degrees of freedom and (b) total wall time.

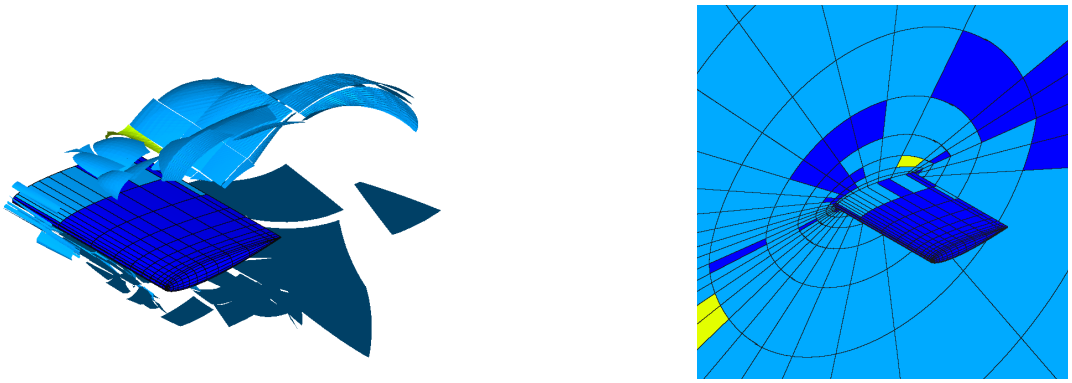


(a) Residual-adapted orders (b) Temporal grids

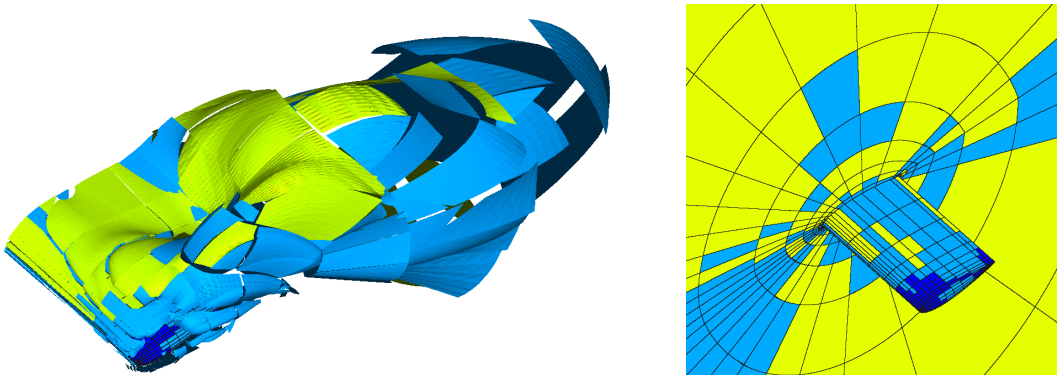
Figure 21: 3D wing: (a) Spatial orders from the residual adaptation midway through the simulation (blue is $p = 0$, red is $p = 3$). The adaptation targets only elements far from the wing, leading to poor output convergence. (b) Temporal grids from the final output-based and residual adaptations. The residual adaptation refines periodically with the motion, while the output-based adaptation increases the resolution near the final times.

conservation, and we saw that accurate error estimates were obtained only if a corresponding GCL adjoint was used. However, an important question is: what if the GCL were ignored from the start? If we accept the possibility of conservation errors, we can neglect the GCL equation and simply solve the original form of the governing equations (Eqn. 2). If this is done, no separate GCL adjoint is required, and the state adjoints provide the information necessary for error estimation and adaptation.

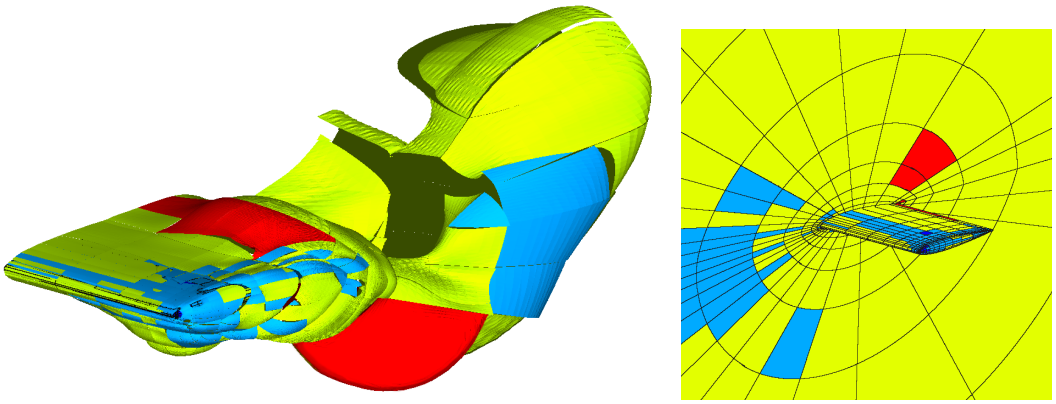
Ignoring the GCL for the airfoil cases presented above gives the output convergence histories shown in Figure 23. We see that use of the GCL for these cases does provide a slight improvement in output convergence; however, if error estimation is performed, both GCL and no-GCL runs show nearly identical corrected outputs. This suggests that



(a) $t = T$



(b) $t = 2.25T$

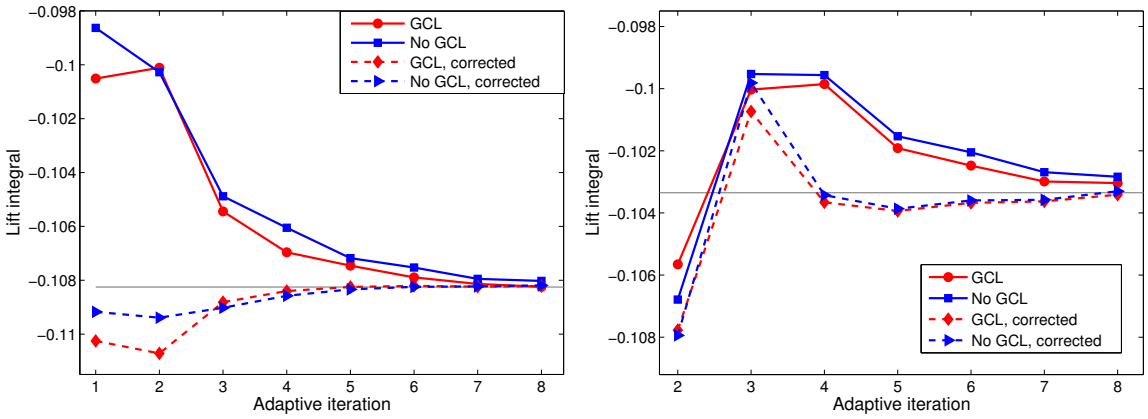


(c) $t = 2.8T$

Figure 22: 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.

satisfying the GCL may not be critical to obtaining accurate outputs, especially if those outputs are corrected by an *a posteriori* error estimate.

Granted, the cases presented were at relatively low Mach number, and we might expect the difference between GCL and no-GCL runs to increase as compressibility becomes more important. In particular, for cases with shocks, it is known that incorrect shock speeds can be obtained if a numerical method is not strictly conservative [35, 36].



(a) Single airfoil case

(b) Two-airfoil case

Figure 23: Output convergence for the single and two-airfoil runs with and without the GCL.

However, we have found that even for cases with relatively strong shocks, correct solutions are obtained with no-GCL runs provided the order and/or mesh resolution is high enough (see Appendix C).

Overall, for cases where the GCL can be safely ignored, the primary savings would be in implementation rather than computation time. The GCL represents a small (~3%) addition to the CPU time of the primal problem, while the GCL adjoint adds roughly 6% to the adjoint solve time. This means that the total reduction in CPU time obtained by neglecting the GCL would be about 5%.

9. Conclusions

In this paper, we derived unsteady adjoints for both the governing equations and the geometric conservation law, and showed that using these adjoints to drive adaptation provides orders of magnitude savings in the mesh size required for output convergence. A significant reduction in computational time was also achieved when the adjoints were reconstructed or smoothed to an appropriate tolerance. More common adaptation methods, such as uniform or residual-based refinement, were shown to be either inefficient or erratic for cases with mesh motion. These results were demonstrated in both two and three dimensions.

The overall impact of the GCL on error estimation was also assessed. While the contribution of the GCL to the output error is generally smaller than the state contribution, it can become significant during later stages of adaptation. Hence, if the GCL is enforced, a corresponding GCL adjoint is required to ensure accurate error estimates. However, preliminary results indicate that for certain cases, accurate output values may be obtained while ignoring the GCL altogether, particularly if the output is corrected by an *a posteriori* error estimate.

Overall, the proposed output-based adaptation represents an unquestionable improvement over more heuristic strategies when considering total mesh size. While reductions in CPU time were also obtained, further gains in this area are possible. For parallel computations, an algorithm for dynamically repartitioning the mesh could prove beneficial, while changes to the adaptive strategy itself may provide additional improvement. These issues are the subject of ongoing research.

Acknowledgments

The authors acknowledge support given by the University of Michigan and by the Air Force Office of Scientific Research under grant FA9550-11-1-0081. Further support was provided by the Department of Defense through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program.

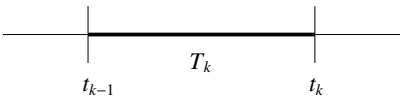
Appendix A. Derivation of the DG Space-time Residuals

Equations 3 and 4, once discretized spatially using the standard DG formulation, can be written as the following system of discrete ordinary differential equations:

$$\mathbf{M}\frac{d\mathbf{U}}{dt} + \mathbf{R}_{\mathbf{U}}(\mathbf{U}, \mathbf{G}) = 0 \tag{A.1}$$

$$\mathbf{M}\frac{d\mathbf{G}}{dt} + \mathbf{R}_{\mathbf{G}}(t) = 0\,, \tag{A.2}$$

where $\mathbf{M}$ is the standard DG mass matrix spanning the spatial domain, $\mathbf{R}_{\mathbf{U}}$ and $\mathbf{R}_{\mathbf{G}}$ are the spatial state and GCL residuals (respectively), and we drop the subscript H for clarity. To integrate these equations in time, we divide the temporal domain into slabs, with a typical time slab k (denoted by $T_k = [t_{k-1}, t_k]$) shown below.



For an order r temporal discretization, we represent the state and GCL on each time slab k as a sum of weighted Lagrange basis functions, $\varphi^n(t)$:

$$\mathbf{U}^k(t) = \sum_{n=1}^{r+1} \mathbf{U}^{k,n}\varphi^n(t), \qquad \mathbf{G}^k(t) = \sum_{n=1}^{r+1} \mathbf{G}^{k,n}\varphi^n(t), \qquad \text{for} \quad \mathrm{t} \in T_k\,. \tag{A.3}$$

These representations are discontinuous between time slabs. Multiplying (A.1) by test functions $\varphi^m(t)$ and integrating over the time slab yields

$$-\int_{T_k}\left(\frac{d\varphi^m(t)}{dt}\mathbf{M}\mathbf{U}^k(t)\right)dt + \left[\varphi^m(t)\mathbf{M}\mathbf{U}(t)\right]_{t_{k-1}}^{t_k} + \int_{T_k}\varphi^m(t)\mathbf{R}_{\mathbf{U}}\left(\mathbf{U}^k(t), \mathbf{G}^k(t)\right)dt = 0, \tag{A.4}$$

where the time derivative term was integrated by parts. Next, we use an "upwind" flux for $\mathbf{U}(t_{k-1})$; that is, we replace $\mathbf{U}(t_{k-1})$ in the above equation with the right-most state on the previous time slab, $\mathbf{U}^{k-1,r+1}$. Making this substitution, and replacing $\mathbf{U}^k(t)$ in the first term with its expansion in (A.3), we then obtain

$$-\int_{T_k}\left(\frac{d\varphi^m(t)}{dt}\mathbf{M}\sum_n \mathbf{U}^{k,n}\varphi^n(t)\right)dt + \varphi^m(t_k)\mathbf{M}\mathbf{U}^{k,r+1} - \varphi^m(t_{k-1})\mathbf{M}\mathbf{U}^{k-1,r+1} + \int_{T_k}\varphi^m(t)\mathbf{R}_{\mathbf{U}}\left(\mathbf{U}^k(t), \mathbf{G}^k(t)\right)dt = 0. \tag{A.5}$$

In the first term, only $\varphi^m(t)$ and $\varphi^n(t)$ depend on time, so we can rewrite the above equation as

$$-\left(\int_{T_k}\varphi^n(t)\frac{d\varphi^m(t)}{dt}dt\right)\mathbf{M}\mathbf{U}^{k,n} + \varphi^m(t_k)\mathbf{M}\mathbf{U}^{k,r+1} - \varphi^m(t_{k-1})\mathbf{M}\mathbf{U}^{k-1,r+1} + \int_{T_k}\varphi^m(t)\mathbf{R}_{\mathbf{U}}\left(\mathbf{U}^k(t), \mathbf{G}^k(t)\right)dt = 0, \tag{A.6}$$

with implied summation over the n index. To further simplify, we can combine the first and second terms and write:

$$a^{m,n}\,\mathbf{M}\mathbf{U}^{k,n} - \varphi^m(t_{k-1})\mathbf{M}\mathbf{U}^{k-1,r+1} + \int_{T_k}\varphi^m(t)\mathbf{R}_{\mathbf{U}}\left(\mathbf{U}^k(t), \mathbf{G}^k(t)\right)dt = 0, \tag{A.7}$$

where the $a^{m,n}$ are constant coefficients given by

$$a^{m,n} = \varphi^n(t_k)\,\varphi^m(t_k) - \int_{t_{k-1}}^{t_k}\varphi^n(t)\frac{d\varphi^m(t)}{dt}\,dt. \tag{A.8}$$

By the properties of Lagrange bases, the first term in $a^{m,n}$ is nonzero only when $n = m = r + 1$, and we recover the second term in Equation A.6, as desired. Finally, for dynamic-p simulations, Equation A.7 must be altered slightly,

since the details of the spatial discretization (and hence $\mathbf{M}$) change in time. We thus write the final form of the state residuals as follows:

$$\overline{\mathbf{R}}_{\mathbf{U}}^{k,m} \equiv a^{m,n}\,\mathbf{M}^{k,k}\mathbf{U}^{k,n} - \varphi^m(t_{k-1})\,\mathbf{M}^{k,k-1}\mathbf{U}^{k-1,r+1} + \int_{T_k} \varphi^m(t)\,\mathbf{R}_{\mathbf{U}}\left(\mathbf{U}^k(t), \mathbf{G}^k(t)\right) dt = 0, \tag{A.9}$$

where $M^{k,l}$ denotes the mass matrix formed from the spatial basis functions on slabs k and l, which can differ for dynamic-p simulations (see e.g. [26]). The above form of the residual is identical to the one presented in Eqn. 9, and represents an implicit set of equations that must be solved on each time slab. The derivation of the GCL residuals from Eqn. A.2 follows the same logic, so we omit it for brevity.

Appendix B. Output Convergence Rates when Superconvergence Occurs

In Section 7.1, we claimed that for a superconverging output defined on a downwind time node, the expected convergence rate for the corrected output ($J - \delta J_{est}$) is $2(r+1)$, where r is the order of the DG time scheme. A corrected output typically converges at the fine-space rate, since the error estimate δJ_{est} should ideally correct the coarse-space output to the fine-space value. However, this is not necessarily the case when superconvergence occurs. This is because the error estimate δJ_{est} is inexact for nonlinear problems, which we can see by writing the true error (assuming no GCL for simplicity) as

$$J_H - J_h \;=\; \underbrace{-\boldsymbol{\Psi}_{\mathbf{U},h}^T\,\overline{\mathbf{R}}_{\mathbf{U},h}(\mathbf{U}_h^H)}_{\delta J_{est}} \;+\; \underbrace{R^{(2)}(\|\delta\mathbf{U}_h\|, \|\delta\boldsymbol{\Psi}_{\mathbf{U},h}\|)}_{remainder}\,. \tag{B.1}$$

Here, the remainder term, which is second order in $\delta\mathbf{U}_h$ and $\delta\boldsymbol{\Psi}_{\mathbf{U},h}\,(\equiv\ \boldsymbol{\Psi}_{\mathbf{U},h} - \boldsymbol{\Psi}_{\mathbf{U},h}^H)$, includes the high order terms neglected in the original Taylor expansions. The perturbations $\delta\mathbf{U}_h$ and $\delta\boldsymbol{\Psi}_{\mathbf{U},h}$ each converge at the coarse-space rate of $r+1$, so $R^{(2)}$ (being second order in these perturbations) is expected to converge at a rate of $2(r+1)$.

Since the corrected output can be written as

$$J_{corrected} = J_H - \delta J_{est} = J_h + R^{(2)}(\|\delta\mathbf{U}_h\|, \|\delta\boldsymbol{\Psi}_{\mathbf{U},h}\|), \tag{B.2}$$

we see that its convergence rate will correspond to the rate of *either* J_h or $R^{(2)}$; whichever is lower. The convergence rate of J_h is $2r+1$ when superconvergence occurs, so if an $r=2$ time scheme is used in the fine space, we expect J_h to converge at a rate of $2(2)+1=5$. On the other hand, since an $r=1$ scheme is used in the coarse space, $R^{(2)}$ will only converge at a rate of $2(1+1)=4$. So in this case, the fourth order convergence of $R^{(2)}$ is lower than the fifth order superconvergence of J_h, which means the corrected output will be limited to fourth order convergence. This is consistent with the results in Figure 9.

Appendix C. Relevance of the GCL for a 1D Shock Tube Problem

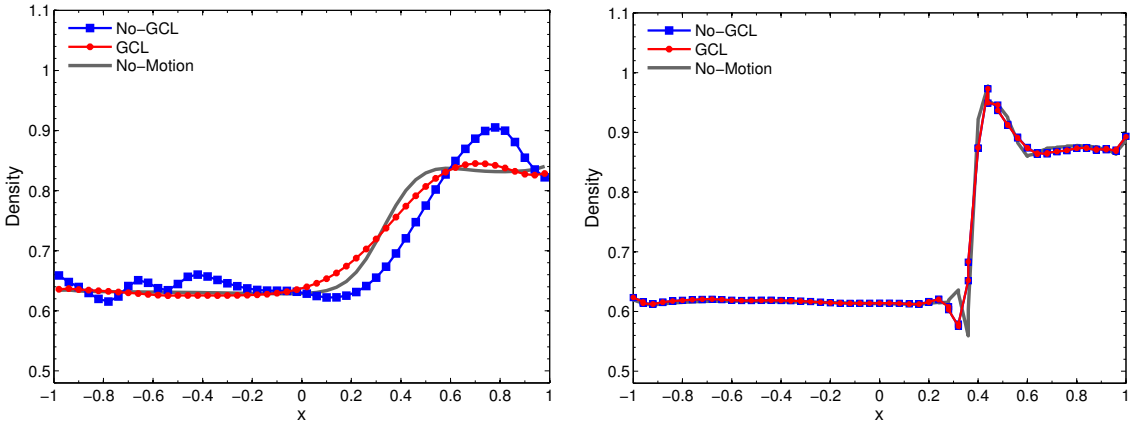
It is known that a lack of strict conservation can cause some numerical methods to give incorrect shock speeds and strengths, even upon mesh refinement [35, 36]. For simulations with mesh motion, we know that if we do not satisfy the GCL, conservation errors can (and do) arise. An important question is whether these conservation errors actually lead to incorrect shock properties. If they do, then satisfying the GCL would be a necessity for problems with shocks.

To investigate this issue, we solve a simple one-dimensional shock tube problem. The initial condition is a standard Riemann problem, with a discontinuity located at the center of the tube, and the left and right states shown in Table C.1. We solve the problem with (i) no mesh motion, (ii) mesh motion and GCL, and (iii) mesh motion without the GCL. For the motion cases, a strong left-right "plunge" motion is introduced into the 1D mesh, which consists of 50 elements and spans the domain $[-1, 1]$.

By the end of the simulation, the initial shock has reflected several times off of the domain boundaries, and the mesh has undergone 5 periods of motion. Snapshots of density at the final time are shown in Figure C.1. We see that satisfaction of the GCL makes a large difference for the under-resolved (i.e. $p=0$) runs, but essentially no difference at all as soon as $p=1$ is used. For the $p=0$ case, the no-GCL run gives an incorrect shock speed, as expected, while the run satisfying the GCL gives approximately correct shock properties. However, for $p=1$, both GCL and no-GCL runs give the correct shock position and strength, implying that satisfying the GCL is not critical so long as the mesh is sufficiently fine.

State	Left	Right
ρ	1.0	0.4
u	0.0	0.0
p	1.0	0.32

Table C.1: Left and right states for the shock tube initial condition. ρ, u, and p refer to density, velocity, and pressure, respectively.



(a) $p = 0$ (b) $p = 1$

Figure C.1: Density profiles for a 1D shock tube problem, solved at (a) $p = 0$ and (b) $p = 1$. The profiles shown are snapshots taken at time $t = 5$, after the shock has reflected several times off of the domain boundaries.

References

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

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

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

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

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

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

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

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

[9] M. P. Rumpfkeil, D. W. Zingg, A general framework for the optimal control of unsteady flows with applications, AIAA Paper 2007-1128 (2007).

[10] K.-H. Lee, J. J. Alonso, E. van der Weide, Mesh adaptation criteria for unsteady periodic flows using a discrete adjoint time-spectral formulation, AIAA Paper 2006-692 (2006).

[11] S. K. Nadarajah, A. Jameson, Optimum shape design for unsteady three-dimensional viscous flows using a nonlinear frequency-domain method, AIAA Journal of Aircraft 44 (5) (2007) 1513–1527.

[12] N. K. Yamaleev, B. Diskin, E. J. Nielsen, Local-in-time adjoint-based method for design optimization of unsteady flows, Journal of Computational Physics 229 (2010) 5394–5407.

[13] S. K. Nadarajah, A. Jameson, Optimum shape design for unsteady flows with time-accurate continuous and discrete adjoint methods, AIAA Journal 45 (7) (2007) 1478–1491.

[14] L. Wang, D. Mavriplis, W. K. Anderson, Adjoint sensitivity formulation for discontinuous galerkin discretizations in unsteady inviscid flow problems, AIAA Journal 48 (12) (2010) 2867–2883.

[15] D. Srinath, S. Mittal, An adjoint method for shape optimization in unsteady viscous flows, Journal of Computational Physics 229 (2010) 1994–2008.

[16] D. Meidner, B. Vexler, Adaptive space-time finite element methods for parabolic optimization problems, SIAM Journal on Control Optimization 46 (1) (2007) 116–142.
[17] M. Schmich, B. Vexler, Adaptivity with dynamic meshes for space-time finite element discretizations of parabolic equations, SIAM Journal on Scientific Computing 30 (1) (2008) 369–393.
[18] T. J. Barth, Space-time error representation and estimation in Navier-Stokes calculations, in: S. C. Kassinos, C. A. Langer, G. Iaccarino, P. Moin (Eds.), Complex Effects in Large Eddy Simulations, Springer Berlin Heidelberg, Lecture Notes in Computational Science and Engineering Vol 26, 2007, pp. 29–48.
[19] M. Besier, R. Rannacher, Goal-oriented space-time adaptivity in the finite element galerkin method for the compution of nonstationary incompressible flow, International Journal for Numerical Methods in Fluids 70 (2012) 1139–1166.
[20] K. Mani, D. J. Mavriplis, Discrete adjoint based time-step adaptation and error reduction in unsteady flow problems, AIAA Paper 2007-3944 (2007).
[21] K. Mani, D. J. Mavriplis, Error estimation and adaptation for functional outputs in time-dependent flow problems, Journal of Computational Physics 229 (2010) 415–440.
[22] A. Belme, A. Dervieux, F. Alauzet, Error estimation and adaptation for functional outputs in time-dependent flow problems, Journal of Computational Physics 231 (2012) 6323–6348.
[23] B. T. Flynt, D. J. Mavriplis, Discrete adjoint based adaptive error control in unsteady flow problems, AIAA Paper 2012-0078 (2012).
[24] K. J. Fidkowski, Y. Luo, Output-based space-time mesh adaptation for the compressible Navier-Stokes equations, Journal of Computational Physics 230 (2011) 5753–5773.
[25] Y. Luo, K. Fidkowski, Output-based space time mesh adaptation for unsteady aerodynamics, AIAA Paper 2011-491 (2011).
[26] K. J. Fidkowski, An output-based dynamic order refinement strategy for unsteady aerodynamics, AIAA Paper 2012-77 (2012).
[27] S. M. Kast, M. A. Ceze, K. J. Fidkowski, Output-adaptive solution strategies for unsteady aerodynamics on deformable domains, ICCFD7 -3802 (2012).
[28] P.-O. Persson, J. Bonet, J. Peraire, Discontinuous Galerkin solution of the Navier-Stokes equations on deformable domains, Computer Methods in Applied Mechanical Engineering 198 (2009) 1585–1595.
[29] F. Bassi, S. Rebay, GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations, in: K. Cockburn, Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation and Applications, Springer, Berlin, 2000, pp. 197–208.
[30] P. L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, Journal of Computational Physics 43 (1981) 357–372.
[31] T. Richter, Discontinuous Galerkin as time-stepping scheme for the Navier-Stokes equations, in: Fourth International Conference on High Performance Scientific Computing Modeling, Simulation and Optimization of Complex Processes, Hanoi, Vietnam, 2009.
[32] K. J. Fidkowski, Output error estimation strategies for discontinuous Galerkin discretizations of unsteady convection-dominated flows, International Journal for Numerical Methods in Engineeringdoi:10.1002/nme.3224.
[33] M. Yano, An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (2012).
[34] S. Adjerid, K. D. Devine, J. Flaherty, L. Krivodonova, A posteriori error estimation for discontinuous Galerkin solutions of hyperbolic problems, Computer Methods in Applied Mechanical Engineering 191 (2002) 1097–1112.
[35] T. Hou, P. LeFloch, Why non-conservative schemes converge to the wrong solutions: Error analysis, Mathematics of Computation 62 (1994) 497–530.
[36] R. J. LeVeque, Numerical methods for conservation laws, in: Lecture Notes in Mathematics, Birkhauser, 1992, pp. 122–129.