AIAA AVIATION Forum
5-9 June 2017, Denver, Colorado
23rd AIAA Computational Fluid Dynamics Conference

Output Error Control Using r-Adaptation

Kaihua Ding* and Krzysztof J. Fidkowski†
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109

We present an output-based adaptive error control strategy based on unsteady r-adaptation, i.e. mesh motion, for the discontinuous Galerkin finite element method. The method uses a discrete unsteady adjoint to compute an error estimate in a scalar output of interest. Localized to space-time elements, this estimate yields an error indicator that identifies regions in space and time where refinement is required to reduce the output error. Rather than employing standard h or p refinement techniques, we adapt the mesh by moving its nodes. This allows elements to grow or shrink without increasing the degrees of freedom. The mesh motion is performed using analytical contraction/expansion functions and node-interpolated motion driven by a spring analogy in an arbitrary Lagrangian-Eulerian framework, and we demonstrate the ability of such motion to reduce output error in scalar advection-diffusion problems.

I. Introduction

The purpose of mesh adaptation is to reduce numerical errors inherent in simulation. The error in consistent and stable numerical methods depends on the mesh size, h, which can refer to a measure of the size of elements or the spacing between nodes, and on the approximation order, p, which can refer to the order of polynomials in a finite-element method or to the order of a finite-difference scheme. To reduce the error, we must decrease h and/or increase p, and this gives rise to what we call standard h,[1–3] p,[4–7] or hp[8–12] refinement strategies.

Numerous techniques exist for adapting the size of a mesh. These include hanging-node refinement,[13–15] local mesh operations,[16–18] and global re-meshing.[19,20] All of these techniques have the capability to increase resolution in some areas and to decrease the resolution in other areas, so that the total number of degrees of freedom can grow, shrink, or remain the same. A related adaptation technique, that is more restricted, but relatively underutilized, is r adaptation[21–26] , in which the nodes of the mesh undergo motion in order to redistribute the resolution to areas that need it the most. The total number of degrees of freedom remains constant, and hence r adaptation is limited in the maximum possible error reduction that it can achieve. However, when coupled with h and/or p refinement, the combined strategy yields a flexible and efficient means for reducing the output error.

In steady-state calculations, r adaptation is expected to produce final meshes that are similar to other h-refinement mechanics, when the number of degrees of freedom is held constant. For example, if global re-meshing is available, there may be no incentive to use r refinement, except possibly to reduce the cost of mesh re-generation. However, in unsteady calculations, r adaptation presents a distinct advantage in its ability to dynamically and smoothly track important features without abrupt refinement changes (e.g. sudden element split) and without error-prone inter-grid

*Graduate Research Assistant, AIAA Member
†Associate Professor, AIAA Senior Member

solution transfers. As long as the mesh motion is smooth, solution techniques on deformable domains/meshes can be applied without the need for solution interpolation or projection.

In this paper, we use an arbitrary Lagrangian-Eulerian framework[27,28] in order to solve a system of unsteady equations on a deforming mesh. Our mesh motion takes two forms: 1) mesh motion governed by analytical mapping functions between the reference and physical space that cause the mesh to either contract or dilate in regions where mesh resolution is needed or not, respectively; 2) mesh motion directed by a spring analogy that indicates where resolutions is needed or not, with the motion's degrees-of-freedom equals to the number of nodes that can be moved. The finite-element discretization is applied on the static reference domain, so that the error reduction arises from the fact that the size of the mapped elements, in physical space, changes in time.

An error indicator is required to drive the r adaptation. This indicator must provide information about where and when the error is largest. In a node/slab-based time marching scheme, this translates to localization of the error to spatial elements and time slabs or nodes. For robust error control of scalar outputs, we use an unsteady adjoint-weighted residual,[29] where the discrete adjoint equations are solved after the primal solution through a reverse time marching procedure.

The remainder of this paper is organized as follows. Section II presents the primal discretization using the discontinuous Galerkin finite-element method and reviews the arbitrary Lagrangian-Eulerian framework. Section III introduce two mesh motion algorithms, 1) analytic function based mesh motion and 2) mesh motion based on interpolation of node data. Section IV presents the adjoint-based error estimation and mesh adaptation strategies. Section V shows preliminary results, and Section VI discusses conclusions.

II. Discretization and Mesh Motion

A. Discretization

We apply the discontinuous-Galerkin (DG) finite-element method to conservation laws of the form

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

where $\mathbf{u}(\vec{x}, t) \in \mathbb{R}^s$ is the state vector, $\mathbf{u}_0(\vec{x})$ is the initial condition, and $\vec{\mathbf{H}} = \vec{\mathbf{F}}(\mathbf{u}) + \vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u})$ is the total flux consisting of inviscid ($\vec{\mathbf{F}}$) and viscous ($\vec{\mathbf{G}}$) contributions. Boundary conditions are specified on $\partial\Omega$ in the form of knowledge of components of the state or the flux. The i^{th} spatial component of the viscous flux is linear in $\nabla \mathbf{u}$, $\mathbf{G}_i = -\mathbf{K}_{ij} \partial \mathbf{u}_j$ (summation implied on $j \in 1 \ldots \text{dim}$), where $\mathbf{K}_{ij}$ is the (i, j) component of the viscous diffusivity tensor.

Denote by T_h the set of N_e elements in a non-overlapping tessellation of the domain Ω. In DG, the state is approximated by polynomials on each element, with no continuity constraints imposed on the approximations on adjacent elements. $\mathbf{u}_h \in \boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$, where

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

and $\mathcal{P}^p$ denotes polynomials of order p on an element Ω_e.

We obtain the weak form of Eqn. 1 by multiplying the PDE by test functions $\mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h$ and integrating by parts to couple elements via fluxes. These consist of Roe's approximate Riemann solver[30] for $\widehat{\mathbf{F}}$, and the second form of Bassi and Rebay (BR2)[31] for the viscous flux, $\widehat{\mathbf{G}}$. Choosing a basis for the trial/test spaces turns the weak form into a nonlinear system of ordinary differential equations,

$$\mathbf{M} \frac{d\mathbf{U}}{dt} + \mathbf{R}(\mathbf{U}) \quad = \quad \mathbf{0}, \tag{2}$$

where $\mathbf{U}$ and $\mathbf{R}$ are the discrete state and residual vectors, respectively, and $\mathbf{M}$ is the spatial mass matrix. We solve this system using an implicit time marching method, presently DG-in-time, with a preconditioned Newton-GMRES solver at each time step.

B. Arbitrary Lagrangian-Eulerian Framework

In an arbitrary Lagrangian-Eulerian (ALE) method, the mesh can move at a velocity different from that of the flow, which is useful for modeling problems in which objects move or deform. In the present case, we are not specifically concerned with deforming domains, and instead use the motion of the mesh to redistribute resolution in the physical domain.

The ALE method uses a map between the static reference domain and the deforming physical domain and solves transformed equations on the reference domain.[27] This transformation is illustrated graphically in Figure 1, and Table 1 defines key quantities.

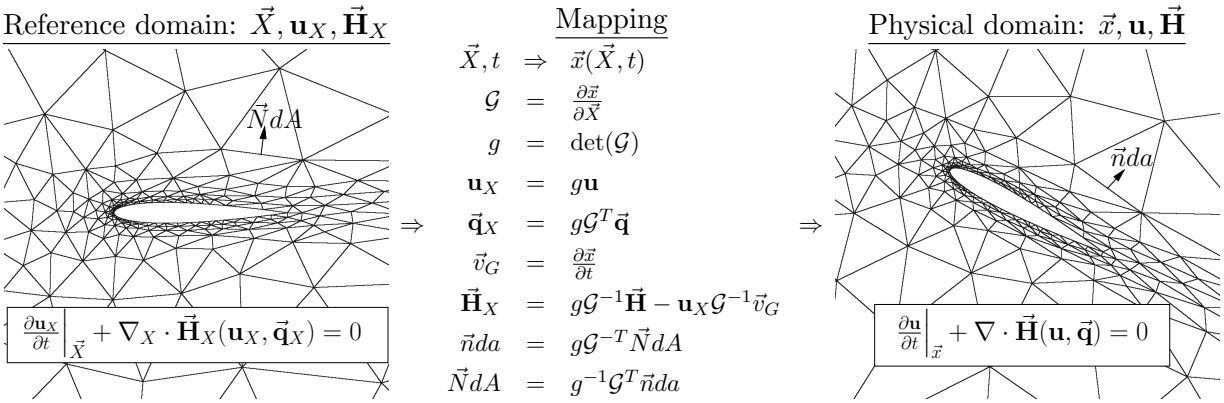


Figure 1. Summary of the mapping between reference and physical domains. The equations are solved on the reference domain, which remains fixed for all time. When denoting reference-domain quantities, we use a subscript X.

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

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

Using a general mapping, $\vec{x} = \vec{x}(\vec{X}, t)$, the PDE on the reference domain becomes

$$\left.\frac{\partial \mathbf{u}_X}{\partial t}\right|_{\vec{X}} + \nabla_X \cdot \vec{\mathbf{H}}_X(\mathbf{u}_X, \nabla_X \mathbf{u}_X) = \mathbf{0}, \tag{3}$$

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

$$\vec{\mathbf{H}}_X = \vec{\mathbf{F}}_X + \vec{\mathbf{G}}_X, \qquad \vec{\mathbf{F}}_X = g\mathcal{G}^{-1}\vec{\mathbf{F}} - \mathbf{u}_X\mathcal{G}^{-1}\vec{v}_G, \qquad \vec{\mathbf{G}}_X = g\mathcal{G}^{-1}\vec{\mathbf{G}}. \tag{4}$$

Eqn. 3 is the form to which the DG method is applied. The mapping Jacobian determinant, g, might not be a polynomial in $\vec{X}$, which leads to slight conservation errors that can be mitigated with a geometric conservation law (GCL).[27,28] Such a GCL is not used in this work as the conservation errors decrease with higher-order approximation and adaptation.[28]

III. Mesh motion algorithm for r-Adaptation

Mesh motion introduces additional error into the solution process comparing to a solve on a static mesh. For r-adaptation to be useful, the error introduced by mesh motion should be smaller than the scheme discretization error.

The choice of the mesh motion algorithm for r-adaptation is dictated by considerations, 1) introducing as little error as possible and 2) handling universal motions, because r-adaptation needs to be flexible enough to handle general error distributions.

Mesh motion based on analytic functions can be made smooth and hence of low error due to mesh motion, yet it is limited in the types of motion that can be realized to control an arbitrary error distribution. On the other hand, mesh motion based on interpolation of nodal velocities is much more flexible in addressing general error distributions, though it is more difficult to implement and it introduces more errors arising from non-smooth features of the mapping. In this work we present implementations of both methods and compare their results.

A. Analytic Function Based Mesh Motion

1. Mapping

In standard applications of ALE, the mapping from reference to global space is chosen to move or deform the geometry in a desired fashion. In the present work, we consider mesh motions that increase or decrease resolution in certain areas of the mesh, at certain targeted times. Such a mapping, categorized as contraction or dilation, follows the following form:

$$\vec{x} = \left(\vec{X} - \vec{X}_0(t)\right) f(\vec{X}, t) + \vec{X}_0(t), \tag{5}$$

where $\vec{X}$ represents the reference-space coordinates, $\vec{X}_0(t)$ represents the center of the contraction/dilation, and $f(\vec{X}, t)$ is a function that dictates the magnitude of the contraction or dilation. We henceforth call f the *contraction function*, though specifically, $f < 1$ indicates a contraction, while $f > 1$ indicates a dilation. Limiting cases include $f = 1$, in which $\vec{x} = \vec{X}$ (no motion), and $f = 0$, in which all of the mesh points coalesce to $\vec{x} = \vec{X}_0$.

We note that the mapping in Eqn. 5 is only a contraction/dilation locally near $\vec{X}_0$. In the surrounding region, the effect of the mapping is opposite, as a contraction near one point causes stretching of the mesh (larger elements) further away from the point, with the distance dependent on the form of $f(\vec{X}, t)$. We choose the following parametrized form for the contraction function,

$$f(\vec{X}, t) \;=\; 1 - A(t) \exp\left[-(\vec{X} - \vec{X}_0(t))^T \mathbf{W} (\vec{X} - \vec{X}_0(t))\right], \tag{6}$$

where $\mathbf{W}$ is a dim $\times$ dim symmetric, positive-definite matrix that encodes the stretching magnitudes and directions. Specifically, the eigenvectors of $\mathbf{W}$ give the direction of stretching, whereas the eigenvalues relate to the stretching magnitude. To make the contraction active for only a certain time, we choose a localized form of the contraction amplitude, $A(t)$,

$$A(t) \;=\; A_0 \exp[-w(t - t_0)^2]. \tag{7}$$

This is a Gaussian, illustrated in Figure 2, with w related to the duration of the contraction (i.e. the width of the Gaussian), and t_0 the time of peak deformation. The amplitude A_0 is set based on

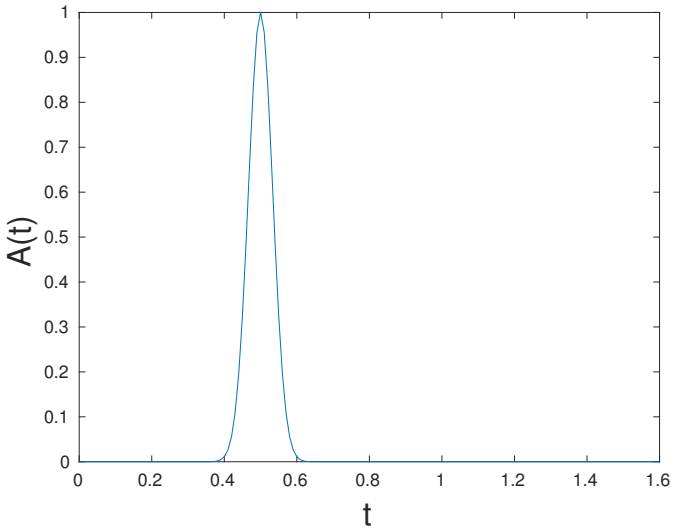


Figure 2. Sample temporal amplitude function for a contraction/dilation map.

user-defined maximum contraction ratio, g_0, which is defined as the volume fraction of an element in physical space relative to reference space. A calculation of the Jacobian of the mapping in Eqn. 6 reveals that the corresponding choice of A_0 is $1 - \sqrt{g_0}$.

2. Superposition

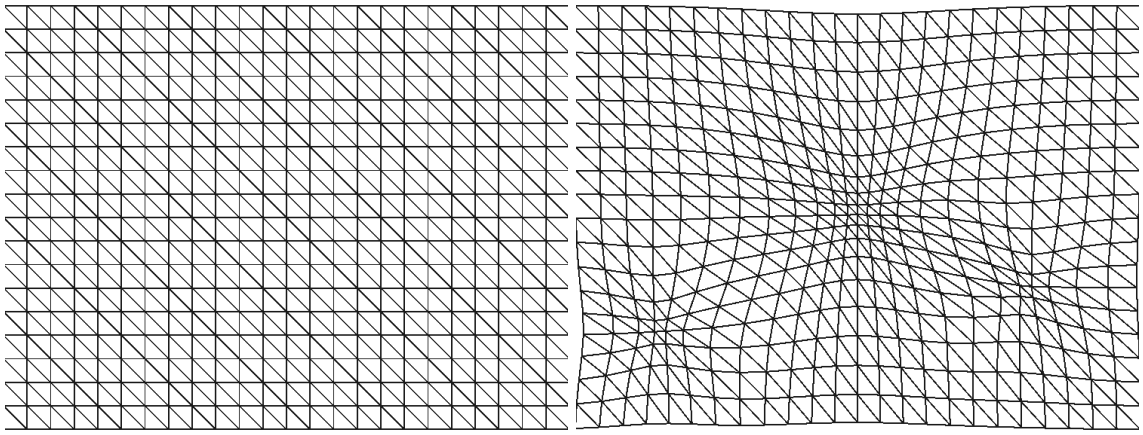
To handle multiple locations and times of large error, we superimpose several contraction mappings, centered at locations of largest error. When dealing with domains of finite extent, we have to take care not to inadvertently alter the geometry under consideration when applying the r-adaptation maps. As the presented spatial function in Eqn. 6 has global effect, we must clip its region of influence. Otherwise we will observe deformation of the computational boundary, as shown in Figure 3.

To avoid this problem, we measure the wall distance, defined as the distance to the closest wall, from each proposed contraction center, $\vec{X}_0$, and use this wall distance to limit the stretching magnitudes. Specifically, we clip the eigenvalues of $\mathbf{W}$ in Eqn. 6 so that the location of the closest wall corresponds to a minimum of three standard deviations of the Gaussian contraction function.

B. Node-Interpolated Mesh Motion Based Mesh Motion for r-Adaptation

In node-interpolated mesh motion (NIMM), the position and velocity of each node is are degrees of freedom that can be used to design a tailored mesh motion. In contrast to analytic function based mesh motion (AFBMM), NIMM does not rely on analytic functions and can hence express more complex motions that are often necessary in r-adaptation.

The NIMM approach to mesh motion is still based on the ALE framework, which requires the mapped coordinates, nodal velocities, and mapping Jacobians and determinant. These are obtained in two steps: 1) interpolation and 2) Jacobian $\underline{G}$ smoothing.



(a) Original static mesh

(b) Mesh with contractions, 3 contracting sources in effect at the same time and caused deformed boundaries

Figure 3. Effect of superimposed contractions without clipping on the computational domain.

1. Interpolation

Interpolation refers to mapping mesh motion information from the mesh nodes to element interiors. This introduces an error that depends on the order of interpolation: for example, a second order error will be introduced when using linear interpolation.

The ALE formulation requires six geometric properties: physical coordinates $\vec{x}$, velocities $\vec{v}$, mapping Jacobian $\underline{G}$, Jacobian determinant g, Jacobian derivatives $\partial G/\partial \vec{x}$ and $\partial G/\partial t$. Two of these, in our case $\vec{x}$ and $\vec{v}$, can be prescribed. With $\vec{x}$ and $\vec{v}$ information, $\underline{G} = \frac{\partial \vec{x}}{\partial \vec{\xi}} \times \left(\frac{\partial \vec{X}}{\partial \vec{\xi}}\right)^{-1}$ and $g = \det \underline{G}$. Lastly, $\partial \underline{G}/\partial \vec{X}$ and $\partial \underline{G}/\partial t$ are zero within an element because the prescribed velocity $\vec{v}$ is constant, which results in zero second-order derivatives with respect to $\vec{x}$.

$\vec{v}$ and $\vec{x}$ are prescribed at mesh nodes (vertices). For any other points where no information was provided, we calculate the information required for ALE through interpolation as explained in Figure 4, with the example of a $2D$ triangular reference element. This interpolation is used to obtain $\vec{x}$, $\vec{v}$, and $\underline{G}$ inside the element. The Jacobian determinant, g, is computed from the interpolated Jacobian: $g = \det \underline{G}$.

2. Jacobian smoothing

As a result of the linear interpolation of coordinates from vertices to the interior of the element, the reference-to-global mapping Jacobian matrix $\underline{G}$ is constant within each element. For general mesh motion, the discontinuous representation of $\underline{G}$ across elements causes errors that pollute the results and destroy the high-order accuracy of the method. We reduce this error by smoothing $\underline{G}$ to create a piecewise-linear, continuous representation.

The Jacobian is smoothed by first averaging the matrix from elements to nodes: $\underline{G}$ at each node is set equal to the average of $\underline{G}$ over the adjacent elements. The Jacobian is then interpolated linearly back to the element interiors to provide the desired piecewise-linear representation.

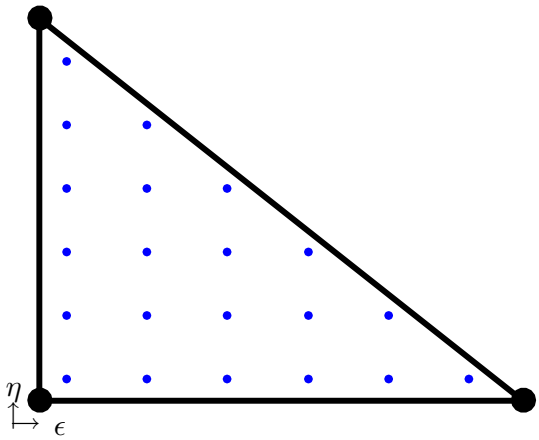


Figure 4. Interpolation in a $2D$ reference space, (ξ, η). Black dots represent linear Lagrange basis function nodes, which are used to interpolate ALE information from the nodes to any other points (blue dots) inside an element or on its edges.

As an example of the effect of smoothing, we consider a scalar advection simulation. The primal initial condition is a constant box function that is nonzero in a region defined by four vertices, [0.5 0.5], [1.5 0.5], [1.5 1.5] and [0.5 1.5]. This scalar initial condition advects through the computational domain at a velocity $\vec{a} = [2.0 \quad 0.1]$, for time $t = 1.0$ with a prescribed sinusoidal mesh motion,

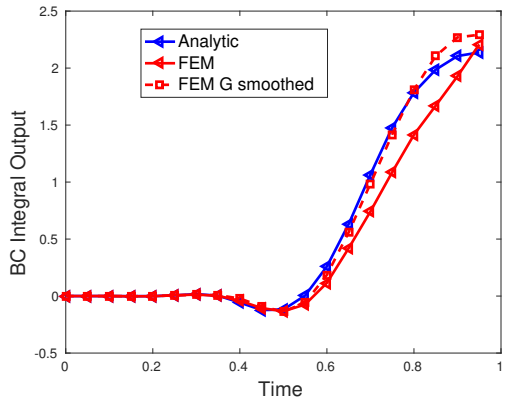
$$\begin{aligned} x_0 &= X_0 + A_x \sin(2\pi x_0/X_0) \sin(2\pi x_1/X_1) \sin(2\pi t/t_0) \\ x_1 &= X_1 + A_y \sin(2\pi x_0/X_0) \sin(2\pi x_1/X_1) \sin(4\pi t/t_0) \end{aligned} \tag{8}$$

where, X_i are reference-space coordinates and x_i are physical-space coordinates, $A_x = 0.3$, $A_y = 0.2$ and $t_0 = 1.0$, and the computational domain is a 3 *by* 2 rectangle. The output is a boundary integral at the right boundary of the computational domain measured at every time step.

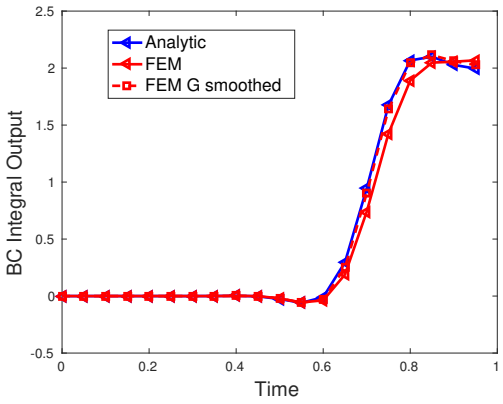
The prescribed motion in Equation 8 is an analytic expression, and hence we can run AFBMM: we treat the result as a reference solution for the NIMM runs. We perform an L_2 error convergence study on NIMM with and without Jacobian smoothing. For the refinement study, the L_2 error is defined as

$$L_2 = \frac{1}{N_t} \sqrt{\sum_{i=0}^{i=N_t} \left(U_i^{\text{analytic}} - U_i^{\text{NIMM}} \right)^2}, \tag{9}$$

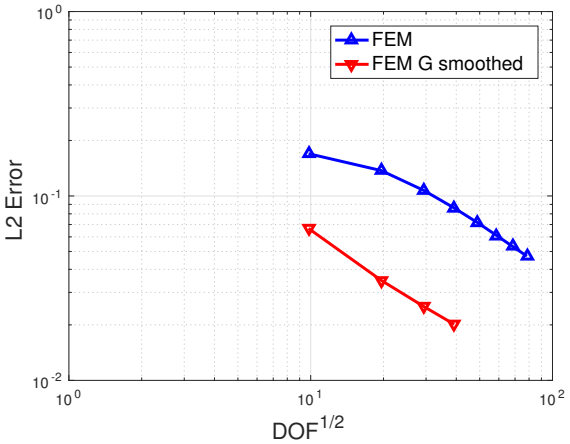
where N_t is the total number of time steps.



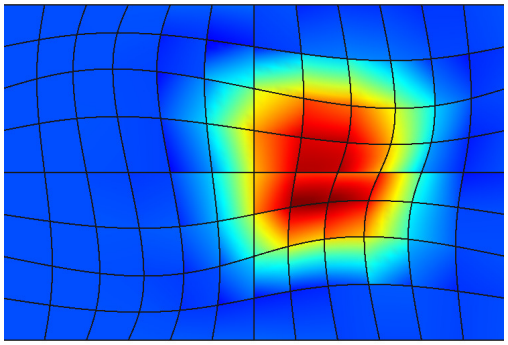
(a) Output evolution with time, first refinement cycle, element number = 96



(b) Output evolution with time, fourth refinement cycle, element number = 1536



(c) Refinement study,



(d) Primal problem is a scalar advection problem, snapshot at t=0.45

Figure 5. Error reduction through a smoother nodal $\underline{G}$ values.

Figure 5 (a) and (b), shows output time histories for both runs, using two different meshes. The output of interest in this case is the integral of the state on the right boundary of the domain. We see that with Jacobian smoothing, NIMM approximates the solution better throughout the whole time history.

Figure 5 (c) shows the L_2 error convergence study with mesh refinement. We see that Jacobian smoothing significantly reduces the l_2 error, though the convergence rate is not affected. In subsequent runs we by default employ Jacobian smoothing.

IV. Error Estimation and Adaptation

A. Error Estimation

An adjoint solution allows us to estimate the numerical error in the corresponding output of interest, $\bar{J}$, through the adjoint-weighted residual.[32,33] The equation governing the discrete unsteady adjoint is obtained by linearizing the residual and output and solving a transposed system backwards in time. To estimate the error, we use this adjoint to weight the unsteady residuals obtained by

injecting the primal solution into a finer space, both spatially and temporally.

Denote by $\mathbf{U}_H(t)$ the approximate primal solution obtained from a chosen time integration method and time step size. To estimate the numerical error in $\mathbf{U}_H(t)$, we solve for the adjoint on a discretization space that is refined in both space and time. For the present study, we use a unit order increment in both space and time to obtain this *fine space*, denoted by subscript h. The resulting final form of the error estimate is

$$\delta \bar{J} \approx -\int_0^T \mathbf{\Psi}_h^T \bar{\mathbf{R}}_h(\mathbf{U}_h^H)\, dt, \tag{10}$$

where $\mathbf{U}_h^H$ is the injection of the primal from space H to space h. The integral in Eqn. 10 is a summation of integrals over time intervals, each performed with appropriate quadrature.

The output error estimate in Eqn. 10 is separated into spatial and temporal components by selectively refining the fine space only in space or only in time.[34] This separation then drives the decision of whether to refine in space or in time.

B. Error Localization and Adaptation

For r-adaptation instructed by analytic function based mesh motion, we employ a simplified localization and adaptation procedure in which a separate contraction mapping is created for every space-time element in the top $f^{\text{adapt}} = 10\%$ fraction of elements with the largest error. The spatial centroid of each selected element serves as the center of the contraction, $\vec{X}_0$. The magnitude of the contraction is set to be as large as possible, subject to the no-wall motion constraint discussed in Section 2. In addition, we heuristically set w in Eqn. 7 to produce a temporal Gaussian function with standard deviation equal to two time steps.

For node-interpolated r-adaptation, the adaptation is driven by a spring analogy previously used for the Active Flux method.[23] The localized error within each element is re-distributed onto element edges via averaging, and we relate this error to the equilibrium length of a spring on the edge. The whole mesh is treated as a web of springs, and a force balance is used to equidistribute the error. At each time step, we move nodes by balancing the forces inside the spring web, where the force on an edge is given by

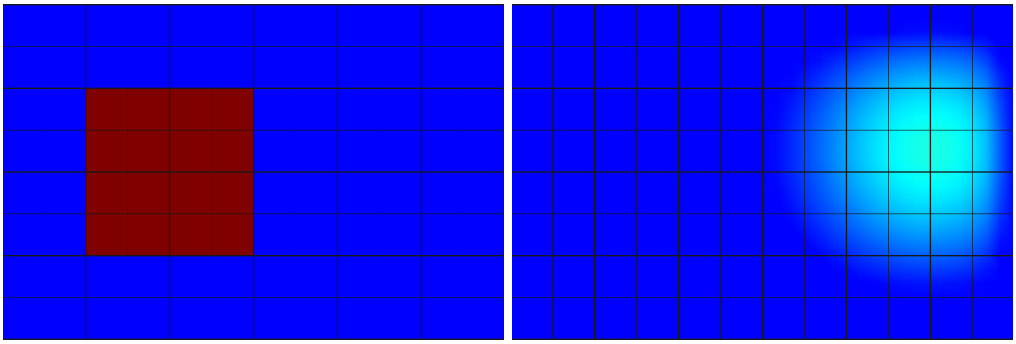
$$F = -K \times (L_{\text{edge length}} - L_{\text{edge-wise error}}) \tag{11}$$

The force balance process is enforced via several iterations during each time step of the unsteady solve. After mesh movement, the error indicator information obtained from the previous adaptive iteration will not be accurate, though after many adaptation iterations, the variation in error indicators becomes smaller and the motion converges. Presently, a termination condition consists of a fixed number of adaptation iterations.

V. Results

We consider a scalar advection-diffusion problem for the unknown concentration u, with advection velocity $\vec{V} = [1, 0.1]$, viscosity $= 1.0$ and homogeneous Dirichlet boundary conditions. The domain is a rectangle, and the initial condition is a zero distribution of the state throughout the domain, with the exception of a square region on the left portion, where $u = 1$. Figure 6 shows this initial condition, as well as the solution at the final time.

The spatial domain is discretized with quadrilaterals in reference space, and the order of approximation is $p_{\text{spatial}} = 1$. The total simulation time is $t_{\text{total}} = 1.6$. The problem is solved using DG-in-time with 20 time steps of order $p_{\text{temporal}} = 1$ initially. r-adaptation may generate smaller



(a) Initial condition (b) Final time

Figure 6. Initial condition and final-time primal solution of the scalar advection-diffusion problem.

elements and result in additional time steps, as determined by the space-time adaptation algorithm. The output of interest is a boundary integral of the state on the right-hand side boundary at the final time.

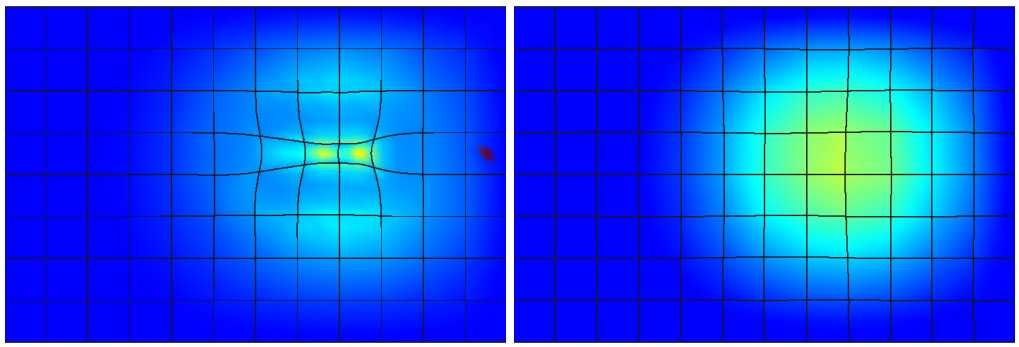
We apply the output-based error estimation procedure to this problem to identify the localized elemental error. We then use the space-time indicators to drive the proposed r-adaptation strategy. Figure 7 and Figure 8 show the resulting mesh at $t = 0.88$ and $t = 1.6$ respectively in the simulation. The contractions of the mesh near the output measurement for both cases are clearly evident.

Figure 7 shows that AFBMM tends to locally alternate mesh resolution while NIMM tends to re-distribute mesh resolution globally. Similarly, Figure 8 displays the same observation, where AFBMM picks the single element with the largest error to place a contraction source, while NIMM shrinks the whole column of elements at the right boundary of the computational domain to achieve a more accurate right-hand-side boundary integral output.

For a quantitative comparison, we measure the error estimate without mesh motion as -1.03×10^{-2}. After three iterations of r-adaptation, i.e. using the no-motion error indicators to drive the r refinement procedure, the error estimate drops to -1.31×10^{-3} for AFBMM, which is almost an order of magnitude error reduction. The error estimate drops to -6.04×10^{-3} for NIMM, which is around a 50% error reduction.

VI. Conclusions

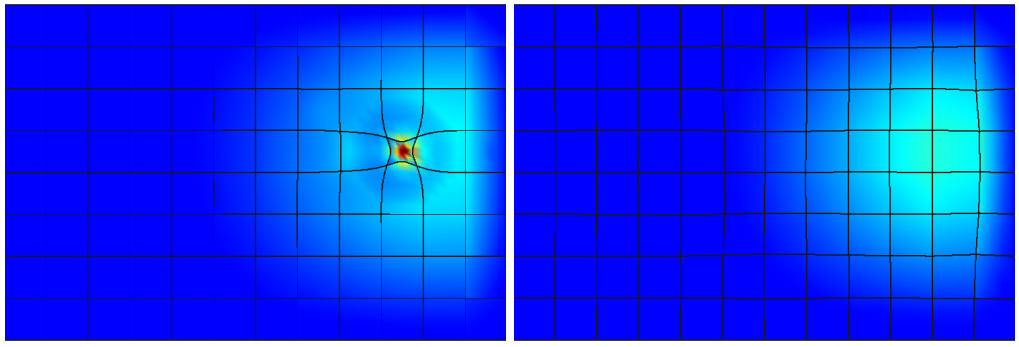
In this paper, we have described a method for mesh adaptation using r-refinement and presented results from an implementation in a discontinuous Galerkin finite element method. The method uses an arbitrary Lagrangian-Eulerian framework for moving the mesh, with two different mesh movement algorithm, AFBMM and NIMM. Both methods show positive results in dropping total error in the simulation, though the NIMM method is expected to be more robust for arbitrary error indicator distributions. In future work, three dimensional cases and complicated physics will be considered.



(a) AFBMM, snapshot at $t = 0.88$. Several contraction sources are placed to redistribute the mesh resolution.

(b) NIMM, snapshot at $t = 0.88$. The mesh is mildly contracted near the center of the scalar profile and a contracting wave that propagates through the domain is observed in the animation.

Figure 7. Snapshot of AFMM and NIMM at $t = 0.88$. AFBMM tends to pick out elements with the highest error and places contraction sources on these elements. On the other hand, NIMM motion deforms the mesh in a more uniform/global fashion.



(a) AFBMM, snapshot at $t = 1.6$. One contraction function is placed around the element with the highest error.

(b) NIMM, snapshot at $t = 1.6$. Elements near the right boundary are warped by NIMM.

Figure 8. Snapshot of AFMM and NIMM at $t = 1.6$.

References

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

[2]Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid solution of high–order discontinuous Galerkin discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 207, 2005, pp. 92–113.

[3]Hartmann, R., "Adaptive discontinuous Galerkin methods with shock-capturing for the compressible Navier-Stokes equations," *International Journal for Numerical Methods in Fluids*, Vol. 51, No. 9–10, 2006, pp. 1131–1156.

[4]Kast, S. M., Ceze, M. A., and Fidkowski, K. J., "Output-adaptive solution strategies for unsteady aerodynamics on deformable domains," Seventh International Conference on Computational Fluid Dynamics ICCFD7-3802, 2012.

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

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

[7]"Adjoint-based error estimation and adaptive mesh refinement for the RANS and $k-\omega$ turbulence model equations," *Journal of Computational Physics*, Vol. 230, No. 11, 2011, pp. 4268 – 4284.

[8]Houston, P. and Süli, E., "hp-adaptive discontinuous Galerkin finite element methods for first-order hyperbolic problems," *SIAM Journal on Scientific Computing*, Vol. 23, No. 4, 2001, pp. 1226–1252.

[9]Wang, L. and Mavriplis, D., "Adjoint-based $h-p$ adaptive discontinuous Galerkin methods for the 2D compressible Euler, equations," *Journal of Computational Physics*, Vol. 228, 2009, pp. 7643–7661.

[10]Burgess, N. K. and Mavriplis, D. J., "An hp-adaptive discontinuous Galerkin, solver for aerodynamic flows on mixed-element meshes," AIAA Paper 2011-490, 2011.

[11]Ceze, M. A. and Fidkowski, K. J., "An anisotropic hp-adaptation framework for functional prediction," *American Institute of Aeronautics and Astronautics Journal*, Vol. 51, 2013, pp. 492–509.

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

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

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

[15]Ceze, M. A. and Fidkowski, K. J., "Output-driven anisotropic mesh adaptation for viscous flows using discrete choice optimization," AIAA Paper 2010-0170, 2010.

[16]Buscaglia, G. C. and Dari, E. A., "Anisotropic mesh optimization and its application in adaptivity," *International Journal for Numerical Methods in Engineering*, Vol. 40, No. 22, November 1997, pp. 4119–4136.

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

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

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

[20]Yano, M., *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.

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

[22]Bank, R. E. and Smith, R. K., "Mesh smoothing using a posteriori error estimates," *SIAM Journal on Numerical Analysis*, Vol. 34, No. 3, 1997, pp. 979–997.

[23]Ding, K., Fidkowski, K. J., and Roe, P. L., "Continuous adjoint based error estimation and r-refinement for the active-flux method," AIAA Paper 2016–0832, 2016.

[24]McRae, D. S., "r-refinement grid adaptation algorithms and issues," *Computer Methods in Applied Mechanics and Engineering*, Vol. 1, No. 189, 2000, pp. 1161–1182.

[25]Zegeling, P. A., "r-refinement for evolutionary pdes with finite elements or finite differences," *Applied Numerical Mathematics*, Vol. 26, 1998, pp. 97–104.

[26]G. Beckett, J. A. Mackenzie, A. R. and Solan, D. M., "On the numerical solution of one-dimensional pdes using adaptive methods based on equidistribution," *Journal of Computational Physics*, Vol. 167, 2001, pp. 372–392.

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

[28]Kast, S. M. and Fidkowski, K. J., "Output-based mesh adaptation for high order Navier-Stokes simulations on deformable domains," *Journal of Computational Physics*, Vol. 252, No. 1, 2013, pp. 468–494.

[29]Fidkowski, K. J. and Luo, Y., "Output-based space-time mesh adaptation for the compressible Navier-Stokes

equations," *Journal of Computational Physics*, Vol. 230, 2011, pp. 5753–5773.

[30]Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, 1981, pp. 357–372.

[31]Bassi, F. and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by K. Cockburn and Shu, Springer, Berlin, 2000, pp. 197–208.

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

[33]Fidkowski, K. J. and Darmofal, D. L., "Review of output-based error estimation and mesh adaptation in computational fluid dynamics," *American Institute of Aeronautics and Astronautics Journal*, Vol. 49, No. 4, 2011, pp. 673–694.

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