AIAA SciTech Forum
4-8 January 2016, San Diego, California, USA
54th AIAA Aerospace Sciences Meeting

10.2514/6.2016-0832

Continuous adjoint based error estimation and r-refinement for the active-flux method

Kaihua Ding[*], Krzysztof J. Fidkowski[†] and Philip L. Roe[‡]
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109

This paper explores the possibility of using continuous adjoints to estimate errors in scalar outputs and to drive mesh adaptation for the active-flux method. Compared to a discrete adjoint approach, the continuous adjoint offers a few attractive advantages for the active-flux method, such as a naturally fully-discrete, explicit implementation that parallels the simplicity of the primal system evolution and error estimates obtained from integrals over the space-time control volumes. The latter point is important because the active flux method does not possess a variational formulation. In addition, we present novel unsteady adaptation mechanics that rely on dynamically moving mesh nodes, i.e. r-refinement, in order to minimize the output error.

I. Introduction

Both discrete and continuous adjoint formulations have been used in various fields, ranging from shape optimization to uncertainty quantification and output-based error estimation. In the discrete formulation, the discrete adjoint vector is obtained by solving a system that results from numerically transposing the primal linearized system and driving it with the output linearization. In the continuous formulation, the adjoint is obtained by separately discretizing and solving the adjoint PDE.

An advantage of the discrete adjoint is that this approach separates the adjoint solve from the "physics" of the problem: the adjoint system results from a purely mathematical transformation of the primal system. Furthermore, when the linearized system is already available, e.g. in implicit primal solvers, the overhead of the discrete adjoint is minor. However, the discrete adjoint masks some potential pitfalls, in particular in ensuring well-posedness and consistency of the adjoint. In contrast, a continuous adjoint approach explicitly addresses these points by requiring appropriate initial and boundary conditions for the adjoint.

In output-based error estimation, most approaches so far have used the discrete adjoint [1–3], though experiments with the continuous adjoint have also been attempted [4]. For light-weight solvers, such as the active flux method, the continuous adjoint approach may be advantageous as linearizing and transposing the primal solver may be computationally burdensome compared to discretizing the adjoint PDE "from scratch" and applying the same light-weight solver. In this work, we take this approach to error estimation and we couple the error estimates with novel mesh adaptation mechanics based on mesh motion.

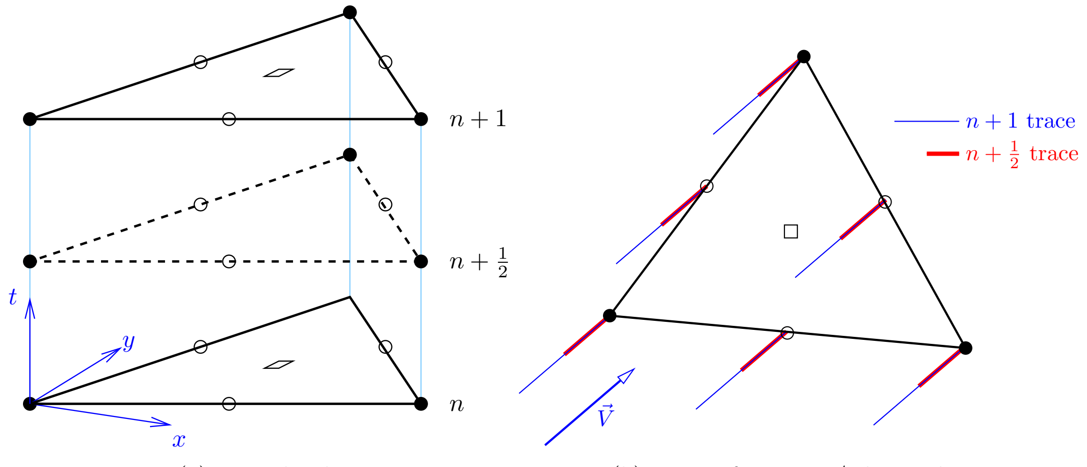
[*]Graduate Research Assistant, AIAA Member
[†]Associate Professor, AIAA Senior Member
[‡]Professor, AIAA Fellow

II. The Active Flux Method

The active flux method is a third-order finite volume method developed by Eymann and Roe [5–8], building on the work of van Leer [9]. We briefly review the method for a fist-order conservation law of the form

$$\mathbf{r}(\mathbf{u}) \equiv \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}} = \mathbf{0}, \tag{1}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the state vector, $\vec{\mathbf{F}} \in [\mathbb{R}^s]^{\text{dim}}$ is the flux, and $\mathbf{r}(\cdot)$ is the continuous/strong-form residual operator. At present, we consider scalar advection, i.e. $s = 1$ and $\vec{\mathbf{F}} = \mathbf{V}u$. The active flux method is an inherently unsteady discretization in which a third-order accurate solution representation at time level n is propagated to time level $n+1$ through an intermediate level $n+\frac{1}{2}$. Figure 1 illustrates the time levels and unknowns for one element of a triangular mesh. Seven unknowns pertain to each triangular element: one at each vertex, one at each edge midpoint,



(a) Time levels (b) Traces for vertex/edge updates

Figure 1. Illustration of the three time levels and unknown placement for one element in the active flux method. Shaded circles are vertex unknowns, open circles are edge unknowns, and the squares represent the cell average unknown.

and one cell average, $\bar{u}$. The vertex and edge unknowns are not unique to the element – they are shared with the neighbors to yield a continuous solution representation. At each time level, these unknowns support an augmented quadratic spatial representation of the solution on the element. Specifically, the six vertex and edge unknowns are used as coefficients in an expansion with quadratic Lagrange basis functions. This quadratic representation is augmented by a cubic bubble function that vanishes on the element perimeter and whose magnitude is uniquely defined by the requirement that the cell average is $\bar{u}$, the seventh unknown.

With the spatial representation in hand, the update procedure for linear advection is relatively simple. It consists of three steps:

1. Determine values for the edge and vertex unknowns at time levels $n+\frac{1}{2}$ and $n+1$ by "tracing back" the solution along the velocity direction to the known augmented quadratic representation at time level n. This is illustrated schematically in Figure 1(b).
2. On each edge of the element, we now have nine solution values: three at each time level, n, $n+\frac{1}{2}$, and $n+1$. These nine values define a quadratic state, and hence flux, in space and time. In this step, we integrate this flux to obtain a third-order accurate net flux through the edge over the time step.

3. Using the integrated fluxes from all edges of the element, obtain the cell average at $n+1$ from the cell average at n via a standard finite-volume discrete conservation statement.

The active-flux method is inherently explicit and fully-discrete. The adjoint discretization follows the same approach but is marched backwards in time as final-time conditions are specified instead of initial conditions.

III. The Active Flux Method with Mesh Motion

A. One Spatial Dimension

In one dimension, the scalar advection equation with flow speed a becomes,

$$\frac{\partial(au)}{\partial x} + \frac{\partial u}{\partial t} = 0, \tag{2}$$

and this can be re-written in compact form as,

$$\nabla \cdot (\mathbf{V}u) \quad = \quad 0, \tag{3}$$

where $\mathbf{V} = [a \ 1]$, $\mathbf{X} = [x \ t]$, and $\nabla = [\frac{\partial}{\partial x}, \frac{\partial}{\partial t}]$. Consider now a cell undergoing mesh motion, as illustrated in Figure 2.

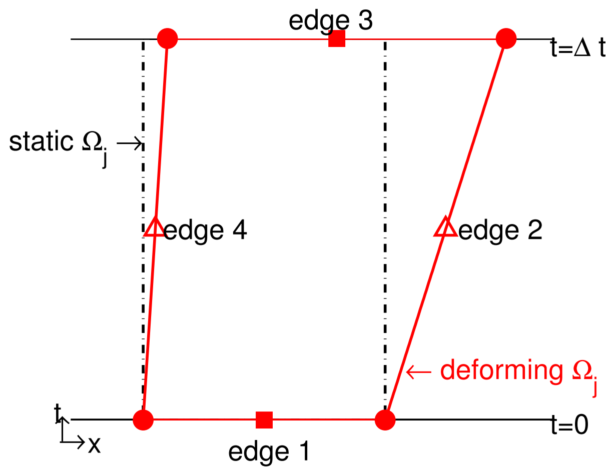


Figure 2. One dimensional element undergoing mesh motion: the node positions do not stay fixed as time progresses, so that the element is no longer rectangular in space-time.

Integrating Eqn. 3 over the non-rectangular space-time volume, Ω_j, yields,

$$\int_{\Omega_j} \nabla \cdot (\mathbf{V}u) = 0 \ \Rightarrow \ \int_{\partial\Omega_j} \mathbf{n} \cdot (\mathbf{V}u) = 0 \tag{4}$$

In the present case, with four space-time edges, the above space-time surface integral leads to a sum of integrals over the four edges,

$$I_1 = \text{Integral on edge 1} \quad = \quad \int_{x_L(t=0)}^{x_R(t=0)} \begin{bmatrix} n_x & n_t \end{bmatrix} \cdot \begin{bmatrix} a & 1 \end{bmatrix} u ds \qquad (n_x, n_t) = [0, -1] \tag{5}$$

$$I_2 = \text{Integral on edge 2} \quad = \quad \int_{0}^{\Delta t} \begin{bmatrix} n_x & n_t \end{bmatrix} \cdot \begin{bmatrix} a & 1 \end{bmatrix} u \left| \frac{ds}{dt} \right| ds \tag{6}$$

$$I_3 = \text{Integral on edge 3} \quad = \quad \int_{x_L(t=\Delta t)}^{x_R(t=\Delta t)} \begin{bmatrix} n_x & n_t \end{bmatrix} \cdot \begin{bmatrix} a & 1 \end{bmatrix} u ds \qquad (n_x, n_t) = [0, 1] \tag{7}$$

$$I_4 = \text{Integral on edge 4} \quad = \quad \int_{0}^{\Delta t} \begin{bmatrix} n_x & n_t \end{bmatrix} \cdot \begin{bmatrix} a & 1 \end{bmatrix} u \left| \frac{ds}{dt} \right| ds \tag{8}$$

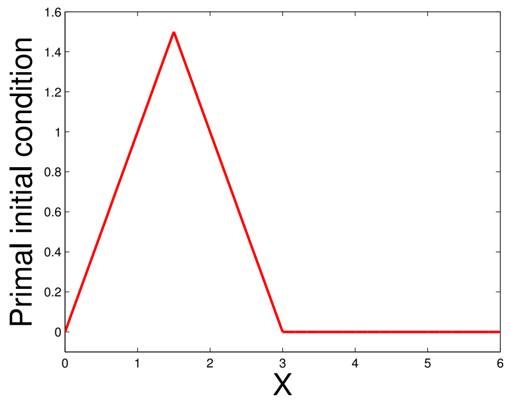
Summing these integrals and setting the result to zero lets us solve for the cell average at $t = \Delta t$,

$$\begin{array}{rcl} I_3 & = & -I_1 - I_2 - I_4 \\ \Longrightarrow \quad [(x_R - x_L)\, \bar{u}]_{t=\Delta t} & = & -I_1 - I_2 - I_4 \\ \Longrightarrow \bar{u}_{t=\Delta t} & = & \dfrac{-I_1 - I_2 - I_4}{(x_R - x_L)_{t=\Delta t}} \end{array} \tag{9}$$

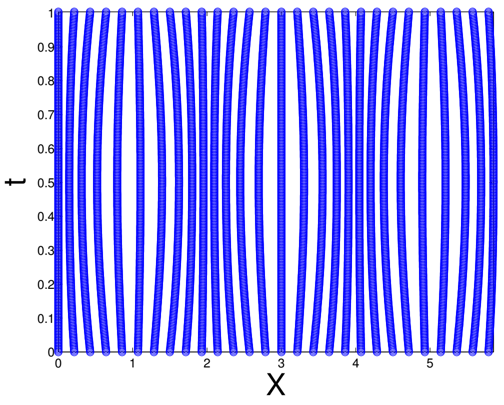
To test the formulation for the active flux method with mesh motion, we consider a problem with a prescribed motion of the form,

$$x(t) = \sin(x_0 \pi)(t - 0.5), \tag{10}$$

where x_0 is the original, time $t = 0$, coordinate. Figure 3(b) shows the resulting motion of 30 initially equally-spaced nodes under this mapping. On this deforming mesh, we consider a primal problem in which a linear hat primal state, shown in Figure 3(a), advects with constant velocity and subject to periodic boundary conditions, for one period.



(a) Primal problem initial condition



(b) Nodes subject to prescribed motion

Figure 3. Setup for a convergence study of the accuracy in the active flux method for a moving mesh.

At the end of one period, we measure the L_2 error norm of the solution error (the exact solution is just the initial condition),

$$e_{L_2} = \sqrt{\frac{1}{L}\int_0^L \left[u_{\text{AF}}(x) - u_{\text{exact}}(x)\right]^2 dx} = \sqrt{\frac{1}{L}\sum_{j=1}^{n_{cell}} \left\{ \int_{(j-1)\Delta x}^{j\Delta x} \left[u_{\text{AF}}(x) - u_{\text{exact}}(x)\right]^2 dx \right\}} \tag{11}$$

In Eqn. 11, n_{cell} is the the total number of cells, and Gauss-Legendre quadrature (order 7) is used to evaluate the integrals. Figure 4 shows the behavior of the error with mesh refinement. We observe that the L_2 error norm converges at a rate of approximately 3, consistent with our expectations, as the active-flux method is third-order accurate.

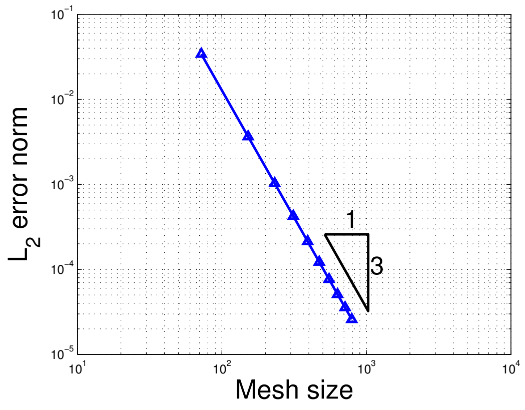


Figure 4. Convergence study for the active flux method on a moving mesh in one spatial dimension, with CFL=0.1.

B. Two Spatial Dimensions

In two spatial dimensions, the scalar advection equation reads

$$\frac{\partial(a_1 u)}{\partial x_1} + \frac{\partial(a_2 u)}{\partial x_2} + \frac{\partial u}{\partial t} = 0, \tag{12}$$

which could be re-written as,

$$\nabla \cdot (\mathbf{V} u) = 0. \tag{13}$$

Here, $\mathbf{V} = \begin{bmatrix} a_1 & a_2 & 1 \end{bmatrix}$ and the space-time coordinate system is $\mathbf{X} = \begin{bmatrix} x_1 & x_2 & t \end{bmatrix}$. Integrating Eqn. 13 in a space-time control volume Ω_j, we obtain,

$$\int_{\Omega_j} \nabla \cdot (\mathbf{V} u) \, d\Omega \equiv \oint_{\partial \Omega_j} [(u\mathbf{V}) \cdot \mathbf{n}] \, dS, \tag{14}$$

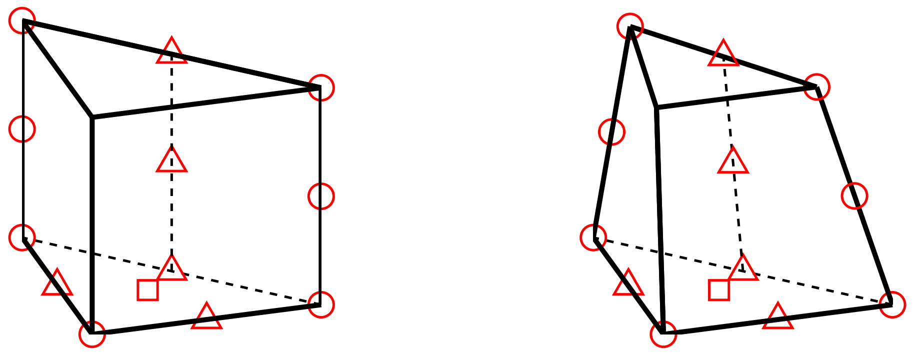
where, $\mathbf{n}$ is the outward pointing unit space-time normal on each face of control volume Ω_j. As Figure 5 shows, solving the governing equations becomes a problem of evaluating five surface integrals.

With or without motion, the top and bottom surfaces of the space-time control volume, Figure 5(a) and Figure 5(b), are flat because the fully discrete active flux method requires the unknown resides at the same time point after each time step. Integrals on these faces can be obtained directly by using quadrature, a 6th order rule in our work.

$$I_{\text{top}} = \int_{\text{top surface}} \mathbf{V} u_H \cdot \mathbf{n}_{\text{top}} dS \qquad \mathbf{n}_{\text{top}} = [0 \;\; 0 \;\; 1] \tag{15}$$

$$I_{\text{bottom}} = \int_{\text{bottom surface}} \mathbf{V} u_H \cdot \mathbf{n}_{\text{bottom}} dS \qquad \mathbf{n}_{\text{bottom}} = [0 \;\; 0 \;\; -1] \tag{16}$$

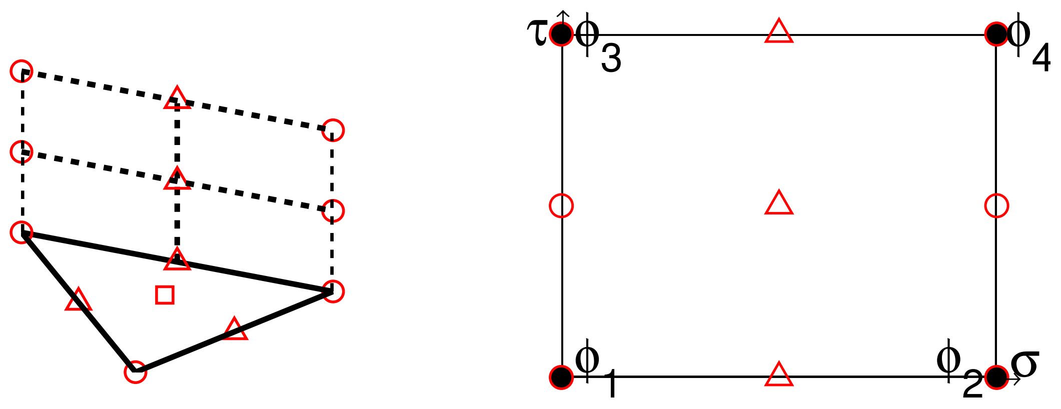
$$\tag{17}$$



(a) Space-time control volume without mesh motion, a right triangular prism.
(b) Space-time control volume with mesh motion, deformed prism.

Figure 5. Space-time control volume in two dimensions

On the other hand, the remaining three surfaces of the prism in Figure 5(b), will generally not be flat in the presence of mesh motion. To integrate on these surfaces, we use a bilinear map to transform the face to a 2D reference square, and we then apply Simpson's rule. The mapping between the physical face and reference square is illustrated in Figure 6.



(a) Surface unknowns
(b) Reference space surface

Figure 6. Mapping between a 3D space-time face in physical space and a reference square.

In Figure 6(b), the black dots represent nodes of bilinear basis functions in reference $[\sigma, \tau]$ space, namely,

$$\begin{aligned} \phi_1 &= (1-\sigma)(1-\tau) \\ \phi_2 &= \sigma(1-\tau) \\ \phi_3 &= (1-\sigma)\tau \\ \phi_4 &= \sigma\tau \ . \end{aligned} \tag{18}$$

The mapping from reference ($\boldsymbol{\xi} = [\sigma, \tau]$) to physical ($\mathbf{X}_{\text{face}}$) space then reads,

$$\mathbf{X}_{\text{face}} = \sum_{i=1}^{4} \mathbf{X}_i \phi_i(\boldsymbol{\xi}) \tag{19}$$

The integral on the 3D surface then transforms into an integral on the reference square, Γ, via

$$\int_{\text{surface}_j} \left[(\mathbf{V} u_H) \cdot \mathbf{n}_{\text{surface}_j} \right] dS = \int_{\Gamma} \left[(\mathbf{V} u_H) \cdot \mathbf{n}_{\text{surface}_j} J_{\text{surface}_j} \right] d\Gamma, \qquad j = 1, 2, 3 \tag{20}$$

where

$$\mathbf{n} J = \frac{\partial \mathbf{X}_{\text{face}}}{\partial \sigma} \times \frac{\partial \mathbf{X}_{\text{face}}}{\partial \tau},$$

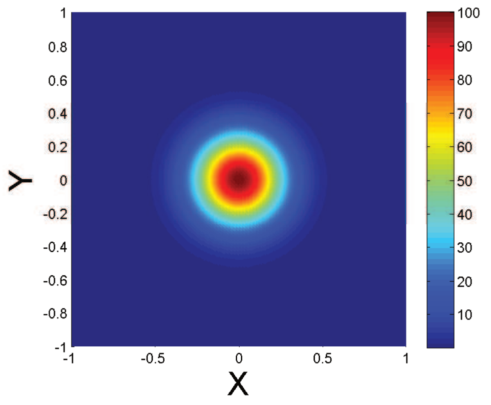
and J is the determinant of the Jacobian matrix,

$$\underline{J} = \frac{\partial \mathbf{X}_{\text{face}}}{\partial \boldsymbol{\xi}} \tag{21}$$

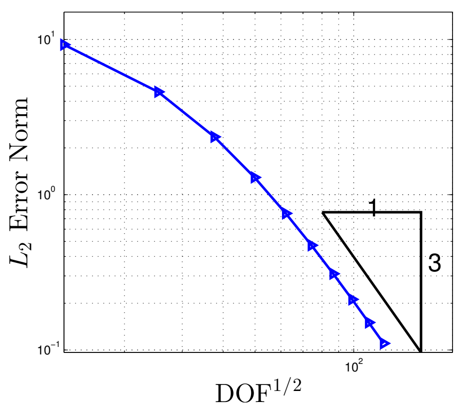
The scheme remains fully conservative because the fluxes between space-time elements remain uniquely defined. As in the case of one spatial dimension, we perform a convergence study for the active flux method with mesh motion. The problem of interest is two-dimensional advection with periodic boundary conditions, illustrated in Figure 7(a). The initial condition is a Gaussian pulse at the domain center, and the advection velocity is [2, 2]. The output of interest is the L_2 error norm,

$$e_{L_2} = \sqrt{\frac{1}{\text{domain area}} \int_{\Omega} \left[u_{\text{AF}} - u_{\text{exact}} \right]^2 d\Omega}, \tag{22}$$

where a 6th order Gauss-Legendre quadrature rule is used for numerical integration.



(a) Primal problem initial condition

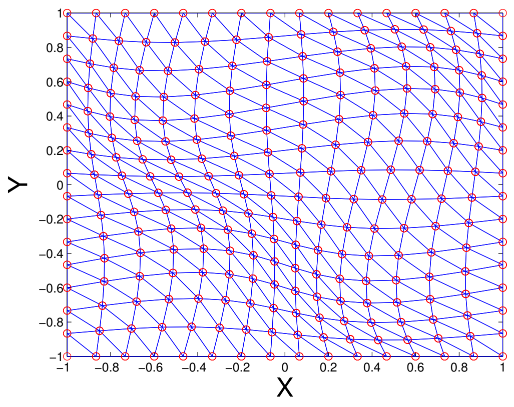


(b) Scheme convergence study

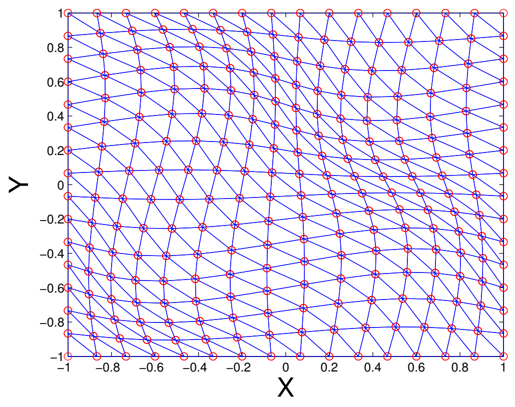
Figure 7. Convergence study for the active flux method with motion in two dimensions, periodic BCs, $a = [2\ 2]^T$, simulation time of one period, and CFL = 0.1.

The motion prescribed for the convergence study is a modified version of that presented by Persson *et al* [10],

$$\begin{aligned} x_1(X_1, X_2, t) &= X_1 + 0.1 \sin(\pi X_1) \sin(\pi X_2) \sin(2\pi t / t_0), \\ x_2(X_1, X_2, t) &= X_2 + 0.1 \sin(\pi X_2) \sin(\pi X_1) \sin(2\pi t / t_0), \end{aligned} \tag{23}$$



(a) $t = 0.3t_0$



(b) $t = 0.7t_0$

Figure 8. Snapshots of the mapping used in the two-dimensional convergence-rate verification study.

where X_1 and X_2 are nodal coordinates on the static mesh, $t_0 = 1.0$ period and t is arbitrary time point between 0 and t_0.

Figure 7(b) shows the result of the convergence study. The error appears to converge at approximately third order, which again agrees with our expectations, as the active flux method is third-order accurate.

IV. The Continuous Adjoint

The continuous adjoint equations are derived from Eqn. 1 using an augmented Lagrangian, which for an unsteady simulation requires integration over the entire space-time domain,

$$\mathcal{L} \equiv J(\mathbf{u}) - \int_T \int_\Omega (\boldsymbol{\psi})^T \mathbf{r}(\mathbf{u}). \tag{24}$$

In this expression T denotes the temporal domain, Ω is the spatial domain, and $\boldsymbol{\psi} \in \mathbb{R}^s$ is the adjoint. linearizing the Lagrangian and requiring that it is stationary with respect to permissible state variations, $\delta\mathbf{u} \in \mathcal{V}^{\text{permissible}}$, we have

$$J'[\mathbf{u}](\delta\mathbf{u}) - \int_T \int_\Omega (\boldsymbol{\psi})^T \mathbf{r}'[\mathbf{u}](\delta\mathbf{u})\, d\Omega\, dt = 0 \qquad \forall\ \delta\mathbf{u} \in \mathcal{V}^{\text{permissible}}, \tag{25}$$

where the primes denote Fréchét linearization with respect to the arguments in square brackets. The second term can be integrated by parts to yield a PDE for the adjoint equation; boundary terms from this integral then provide boundary conditions for the adjoint.

In the case of scalar advection, the adjoint PDE is identical to the primal one. We can see this by carrying out the integration by parts in Eqn. 25,

$$\begin{aligned}
J'[u](\delta u) &= \int_T \int_\Omega \psi \left[\frac{\partial(\delta u)}{\partial t} + \nabla \cdot (\mathbf{V}\delta u) \right] d\Omega\, dt \\
&= \int_T \int_\Omega \psi \frac{\partial(\delta u)}{\partial t} d\Omega\, dt + \int_T \int_\Omega \psi \nabla \cdot (\mathbf{V}\delta u)\, d\Omega\, dt \\
&= -\int_T \int_\Omega \frac{\partial \psi}{\partial t} \delta u\, d\Omega\, dt + \left[\int_\Omega \psi \delta u\, d\Omega \right]_0^T - \int_T \int_\Omega \mathbf{V} \cdot \nabla \psi\, \delta u\, d\Omega\, dt + \int_T \int_{\partial\Omega} \psi \delta u \mathbf{V} \cdot \mathbf{n}\, ds\, dt \\
&= -\int_T \int_\Omega \left[\frac{\partial \psi}{\partial t} + \mathbf{V} \cdot \nabla \psi \right] \delta u\, d\Omega\, dt + \left[\int_\Omega \psi \delta u\, d\Omega \right]_0^T + \int_T \int_{\partial\Omega} \psi \delta u \mathbf{V} \cdot \mathbf{n}\, ds\, dt \qquad (26)
\end{aligned}$$

The first term on the right-hand side is now the domain-interior adjoint equation; it may have additional source terms if the output J depends on the space-time interior state. For the sake of simplicity, we assume that the output J only depends on the state at the final time $t = T$, via

$$J(u) = \int_\Omega j(u)\, d\Omega \Big|_{t=T}.$$

Then, by matching terms in Eqn. 26, we obtain the adjoint PDE,

$$\frac{\partial \psi}{\partial t} + \mathbf{V} \cdot \nabla \psi = 0, \qquad (27)$$

subject to *terminal* conditions,

$$J'[u](\delta u) = \int_\Omega j'[u](\delta u)\, d\Omega \Big|_{t=T} = \int_\Omega \psi \delta u\, d\Omega \Big|_{t=T} \quad \Rightarrow \quad \psi(T) = j'[u]. \qquad (28)$$

Thus, the linearized output weight provides the terminal condition for the adjoint, which is then propagated back to $t = 0$ according to the adjoint PDE in Eqn. 27.

V. Adjoint Verification: Sensitivity Tests

We test the continuous adjoint using a scalar advection problem with periodic boundary conditions. Note, periodic BCs eliminate the last term on the right hand side in Eqn. 26. To test the adjoint, we consider perturbing the initial condition, $u(0)$, by δu_0. Assuming that our adjoint ψ satisfies Eqn. 27 and Eqn. 28, Eqn. 26 reduces to the following expression for the expected output perturbation:

$$\delta J \equiv J'[u](\delta u_0) = -\int_\Omega \psi\, \delta u_0\, d\Omega \Big|_{t=0}. \qquad (29)$$

Given an approximation of the adjoint, ψ_H (what we solve for), in a finite discretization space denoted by H, the error estimate becomes,

$$\delta J \approx -\int_\Omega \psi_H\, \delta u_0\, d\Omega \Big|_{t=0}. \qquad (30)$$

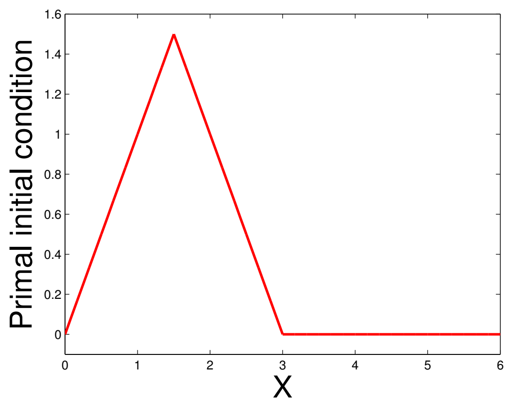
A. Sensitivity Test in 1D

For the one-dimensional sensitivity test, the primal initial condition, output weight function, and state perturbation are chosen as shown in Table 1.

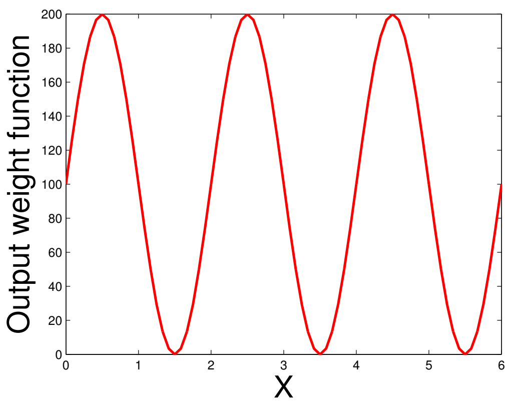
Table 1. Adjoint sensitivity test setup for a 1D scalar advection problem.

Primal initial condition	Output weight function	State perturbation
$u_0 = \begin{cases} x & x \in [0, 1.5) \\ 3 - x & x \in [1.5, 3) \\ 0 & x \in [3, 6] \end{cases}$	$j' = 100\,(\sin(\pi x) + 1)$	$\delta u_0 = 0.5\,(\sin(\pi x) + 1)$

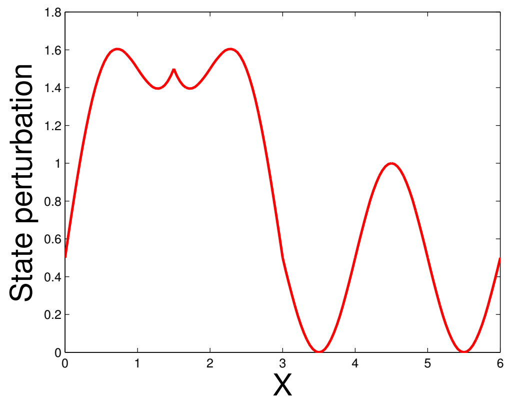
Figure 9 plots these functions and shows the adjoint space-time field – note that this is just the propagation of the output weight-function backwards in time from $t = T$ to $t = 0$. Computing the



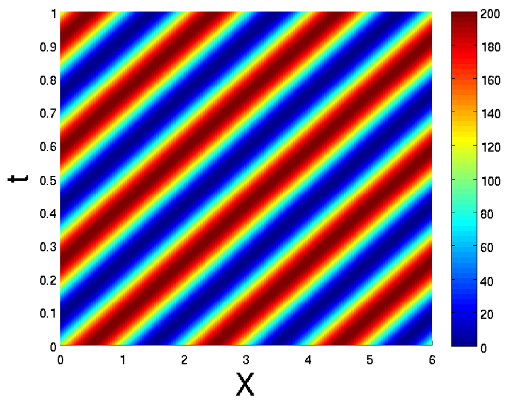
(a) Primal initial condition



(b) Output weight function



(c) Perturbed primal initial condition



(d) Adjoint propagation in space-time

Figure 9. Primal and adjoint initial/terminal conditions and the adjoint space-time field for a 1D scalar advection simulation.

adjoint-based sensitivity in Eqn. 29, we obtain $\delta J = 445.56604\ldots$, which agrees with the actual output perturbation as the space-time mesh is refined.

1. *Sensitivity Test in 1D with Motion*

When mesh motion is present, the sensitivity formulation remains exactly the same as described in Eqn. 30. The sensitivity calculations occur at the time point $t = 0$, and hence mesh motion does not affect the formulation of the sensitivity test. The only effect of mesh motion is on the computation of adjoint $(\psi_H)_{t=0}$, which is time-marched backwards on the deformed mesh.

With the motion prescribed by Figure 3(b), the sensitivity test reaches machine precision values as well. This is a great test to pass to prove the accuracy of the code. However, we need to point out this phenomenon only occurs in one dimension and only occurs when the adjoint terminal condition, i.e. the output weight, is a continuous and slope continuous function throughout the periodic computational domain.

B. Sensitivity Test in 2D

For the sensitivity test in two dimensions, the primal problem is the same as illustrated in Figure 7(a), and the prescribed motion remains as given in Eqn. 23. Figure 10 shows the primal perturbation used to calculate sensitivities. The simulation time is one period at CFL = 0.2. For the adjoint, the output is an area integral at the final time,

$$J = \int_\Omega u\, w(x, y) d\Omega, \tag{31}$$

where the smooth weight, $w(x, y)$, which becomes the adjoint terminal condition, is

$$w(x, y) = \sin\left((x + 1)\,\pi\right) \sin\left((y + 1)\,\pi\right) \qquad x \in [0\ 1],\ y \in [0\ 1], \tag{32}$$

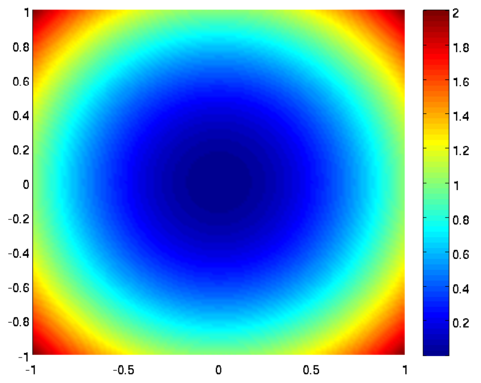


Figure 10. Primal perturbation used in the two-dimensional sensitivity study: $\delta u_0(x, y) = x^2 + y^2$.

Table 2 shows the sensitivity test result for the continuous adjoint. The relative error of the sensitivity test stays around 10%.

	$J_H - J_{\text{perturb}}$	adjoint prediction
$N_{\text{cell}} = 450$	-7.58936×10^{-1}	-9.33519×10^{-1}
$N_{\text{cell}} = 800$	-6.73992×10^{-1}	-7.15801×10^{-1}
$N_{\text{cell}} = 1250$	-6.53913×10^{-1}	-6.62186×10^{-1}

Table 2. Results of the two-dimensional sensitivity test with mesh motion present.

VI. Error Estimation

The continuous adjoint can be used for error estimation via an adjoint-weighted residual. Consider a "coarse" discretization level denoted by the subscript H. This just refers to the fact that finite-size elements and time-steps are used in the discretization. Let's assume that the coarse-space solution has a space-time interpretation, $u_H(x,t)$. The strong-form residual of this coarse-space solution will generally not be zero. Assuming linear equations, we can write,

$$r(u_H) = r(u+\delta u) = r'[u](\delta u) \neq 0, \tag{33}$$

where u is the exact primal solution, and $\delta u \equiv u_H - u$. From the linearized Lagrangian in Eqn. 25, we can compute the error in the output as a result of using u_H instead of u:

$$\delta J = \int_T \int_\Omega \psi\, r(u_H)\, d\Omega\, dt. \tag{34}$$

This expression requires the exact adjoint, ψ, which for general problems will be unavailable. Instead, we approximate the exact adjoint by solving it on a *fine space*, denoted by the subscript h. In the active flux method, this fine space is obtained by uniformly refining the space-time mesh. Substituting the fine-space adjoint into Eqn. 34 gives,

$$\delta J \approx \int_T \int_\Omega \psi_h\, r(u_H)\, d\Omega\, dt \tag{35}$$

Writing $r(u_H)$ in compact space-time form as $r(u_H) = \nabla\cdot(\mathbf{V}u)$, we can integrate the above error estimation formula by parts over the entire space-time domain, denoted by ΩT, to obtain,

$$\delta J \;\approx\; \int_{\Omega T} \psi_h\, r(u_H)\, d\Omega T = -\underbrace{\int_{\Omega T} \nabla\psi_h \cdot \mathbf{V} u_H\, d\Omega T}_{0\text{ by the adjoint eqn.}} + \int_{\partial\Omega T} \psi_h(\mathbf{V}u_H)\cdot\mathbf{n}\, dS. \tag{36}$$

This surface-integral error estimate is applied on each cell of the fine-space mesh.

A. Error Estimation with the Continuous Adjoint on a Moving Mesh in 1D

Consider the one-dimensional advection problem described in Section V and illustrated in Figure 9. In this case, the error estimate obtained from an adjoint on a uniformly-refined fine space gives $\delta J \approx 0.6673$, whereas the true output error is 0.5931, a difference of about 13%. We now show how this error converges with uniform mesh refinement.

We consider the primal initial condition shown in Figure 9(a). Using the same output weighting function as in the sensitivity study, we compute the adjoint-based error estimate at various levels of mesh refinement. Figure 11 shows the convergence of the error estimate, with mesh motion off (Figure 11(a)) and on (Figure 11(b)).

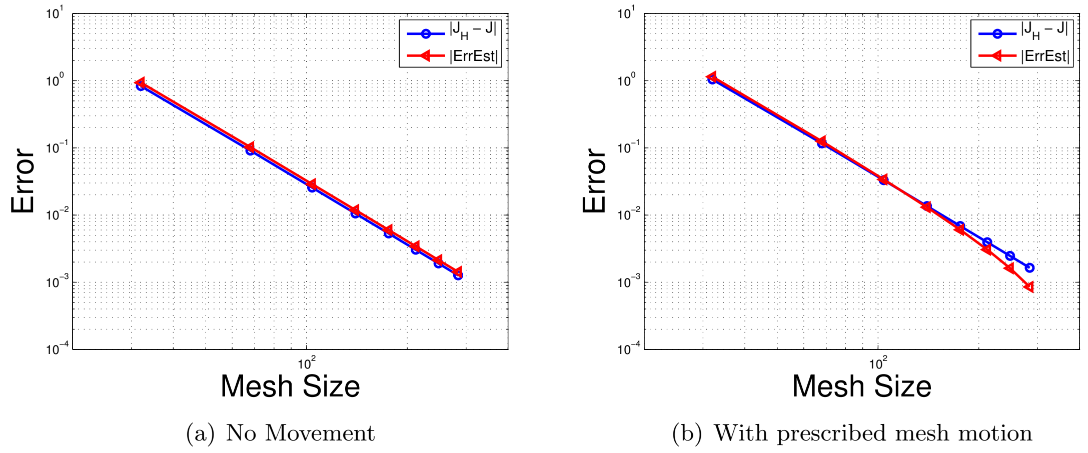


Figure 11. Error convergence study for a one-dimensional scalar advection problem. The mesh motion prescribed for Figure 11(b) is the same as in Figure 4(b). The slopes of all lines are approximately 3.

We note that with mesh motion present, both the output of interest and the error estimate exhibit the expected third order convergence rate.

B. Error Estimation with the Continuous Adjoint on a Moving Mesh in 2D

For the two dimensional problem, we show the error convergence study for the mesh movement problem directly. The output is exactly the same as in the sensitivity study, Eqn. 31, a weighted integral of the primal state at the final time. The mesh motion also remains the same, as given in Eqn. 23. Figure 12 shows the results of the error convergence study.

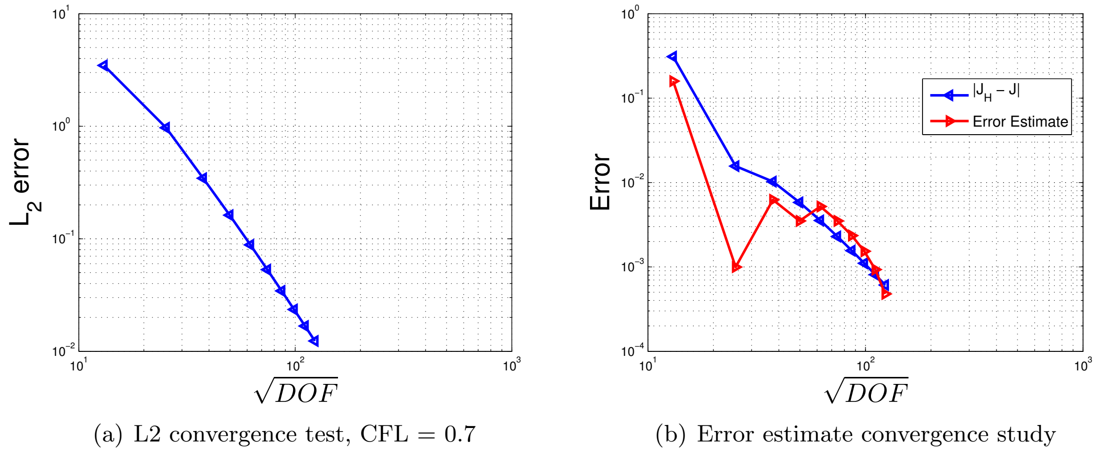


Figure 12. Two-dimensional scalar advection: error estimate convergence study at CFL = 0.7, simulation time of one period. The slope of all lines is approximately 3.

Figure 12(a) shows convergence of the primal L_2 error norm to confirm that the case is converging correctly at CFL = 0.7, (CFL = 0.7 is the largest CFL number our solver can run without mesh entanglement occurring for 'Persson Mapping', which is indicated by negative cell area in our code, noting the CFL condition will change with mesh motion present.) Figure 12(b) shows the convergence of the continuous adjoint error estimate, which is approximately third order.

VII. Adaptive Indicator

To obtain an adaptive indicator, we localize the error estimate in Eqn. 35 to space-time elements of the fine-space discretization:

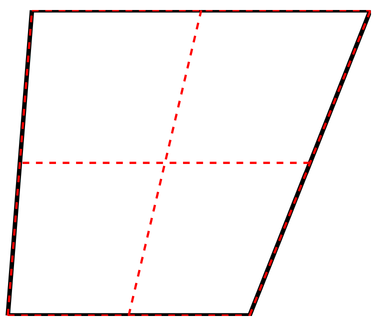
$$\delta J_{\text{remaining}} = -\int_T \int_\Omega \psi_h \, r(u_H) \, d\Omega \, dt = -\sum_{k=1}^{N_t} \sum_{e=1}^{N_{cell}} \int_{t^k}^{t^{k+1}} \int_{\Omega_e} \psi_h \, r(u_H) \, d\Omega \, dt, \tag{37}$$

where N_t is the number of time steps and N_{cell} is the number of elements. On each space-time element, we can integrate the contribution to the error estimate by parts. For example, this contribution is

$$\varepsilon_{e,k} = -\int_{\Omega_e} \delta\psi_h u_H \, dx \Big|_{t^k}^{t^{k+1}} - \int_{\partial\Omega_e} \int_{t^k}^{t^{k+1}} \delta\psi_h u_H \vec{V} \cdot \mathbf{n} \, dt \, ds. \tag{38}$$

Taking the absolute value of this contribution yields an adaptive indicator, the contributions to which are shown in Figure 13(a) for 1D. Similarly, Figure 13b shows the adaptive indicator contribution in 2D.

$$\epsilon_{e,k} = |\varepsilon_{e,k}| = \left| \int_{\Omega_e} \delta\psi_h u_H \, dx \Big|_{t^k}^{t^{k+1}} + \int_{\partial\Omega_e} \int_{t^k}^{t^{k+1}} \delta\psi_h u_H \mathbf{V} \cdot \mathbf{n} \, dt \, ds \right|. \tag{39}$$


(a) Error indicators contribution in 1D

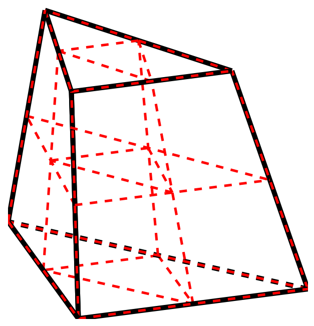

(b) Error indicators contribution in 2D

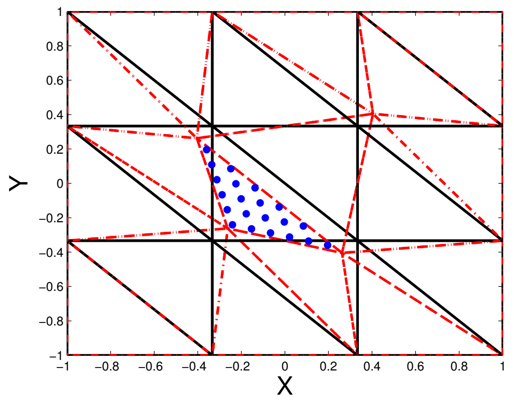
Figure 13. In 1D, the error indicator contributions come from 4 quadrilateral sub-cells of each cell. In 2D, the error contributions come from 8 prismatic sub-cells of each cell.

VIII. Adaptation: r-Refinement for Unsteady Problems

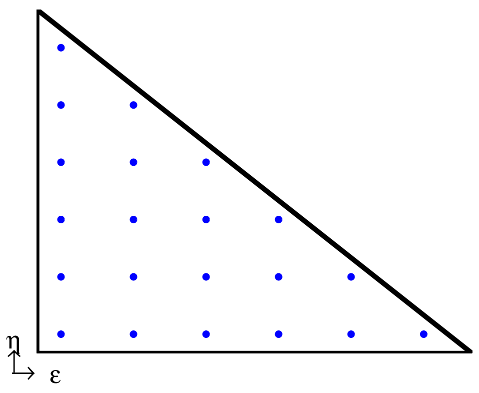
Generally, we expect more advantages from mesh refinement at higher dimensions. For the present study, we applied many different strategies of r-refinement on one dimensional problems to gain insights applicable to two-dimensional r-refinement. Nevertheless, for the final adaptation results, we only show the two dimensional strategy.

For an unsteady problem, r-refinement constitutes efficient adaptation mechanics, as no degrees of freedom get added. We strive for a strategy of error equidistribution over the space-time elements, and since the space-time mesh changes with each iteration, we need a way to map states and indicators from one mesh to another.

Figure 14(a) illustrates such a mapping. The error indicators computed at the previous iteration reside on a mesh, the black mesh, that is different from the r-refined (red) mesh. To obtain error indicator information, we use 21 evenly distributed sampling nodes inside each triangle, Figure 14(b) shows the distribution of sampling nodes in a reference triangle inside reference space. The average of the indicator at all of the sampled nodes is taken as the approximate error quantification on each cell. To locate the sampling nodes, we implement a fast line-based search technique to accelerate the sampling procedure.



(a) Practical mesh and mapping illustration



(b) Error sampling nodes in a reference triangle: 21 sampling nodes in each cell

Figure 14. Error mapping mechanics illustration: the black mesh is the baseline mesh, the red mesh indicates the r-refined mesh, which is dynamically changing throughout the simulation time, and the blue dots are the sampling points for the error indicator transfer.

A. r-Refinement Mechanics for the Active-Flux Method

Given an adaptive indicator on each space-time element, a spring analogy is used to drive the mesh motion. In the spring analogy, edges of the mesh are treated as springs and the mesh is a web of springs. The error information is associated to each 'spring' in the web via averaging of the element-based error indicator, and a force (via Hooke's law) is created by relating the error indicator to the spring equilibrium length, through a prescribed scaling. Presently, a simple inverse relationship between the equilibrium length and the error indicator is employed. The balance of forces in the mesh is then analogous to error equal-distribution, which is a goal of mesh adaptation.

The parameter that influences how well the spring analogy work is the stiffness over nodal mass ratio, k/m. After fixing $m = 1$, the only parameter matters is k. k is a tunable parameter in the spring analogy and a determining factor of how fast the error drops. Its value could be related to the time step size through one-dimensional harmonic oscillator analyses – however, in the present work, we simply tune it to produce reasonable results.

Figure 15 is the flow chart of how the r-refinement mechanics works for our problem, i.e., how we couple the adaptive indicators computation and spring analogy for the code implementation,

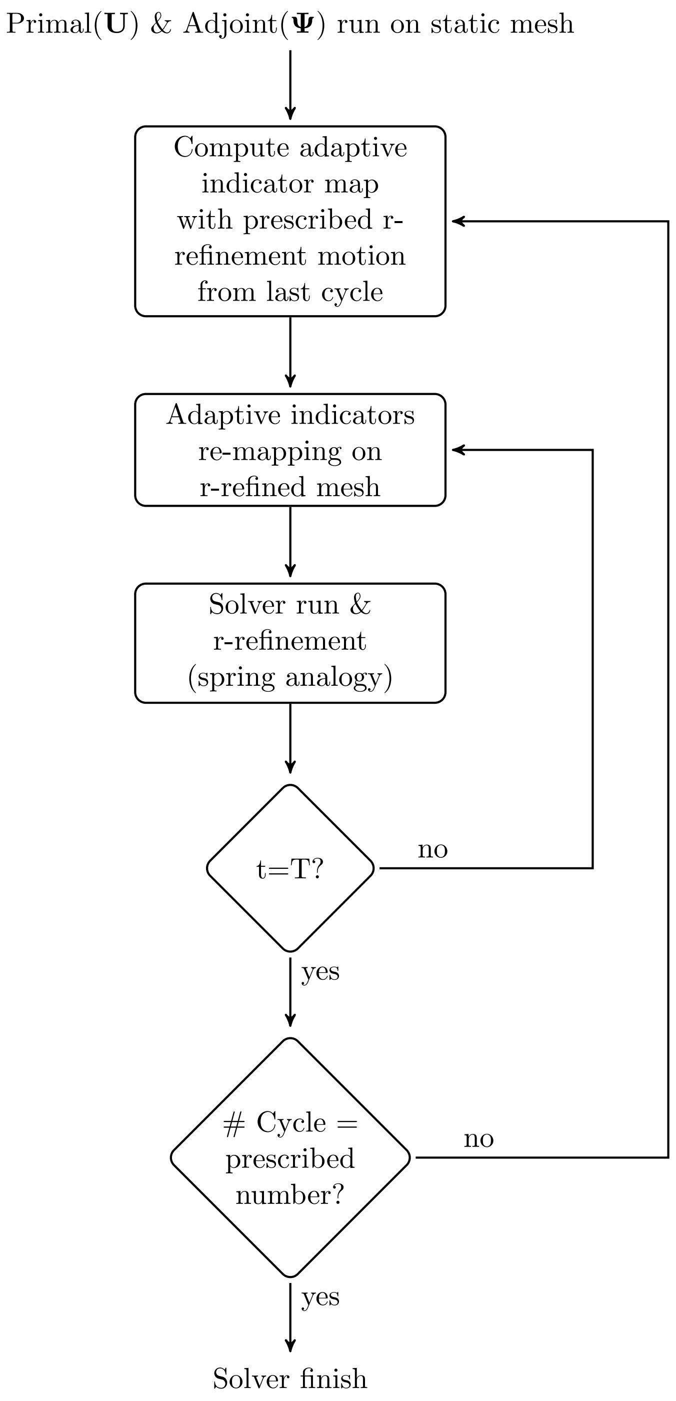
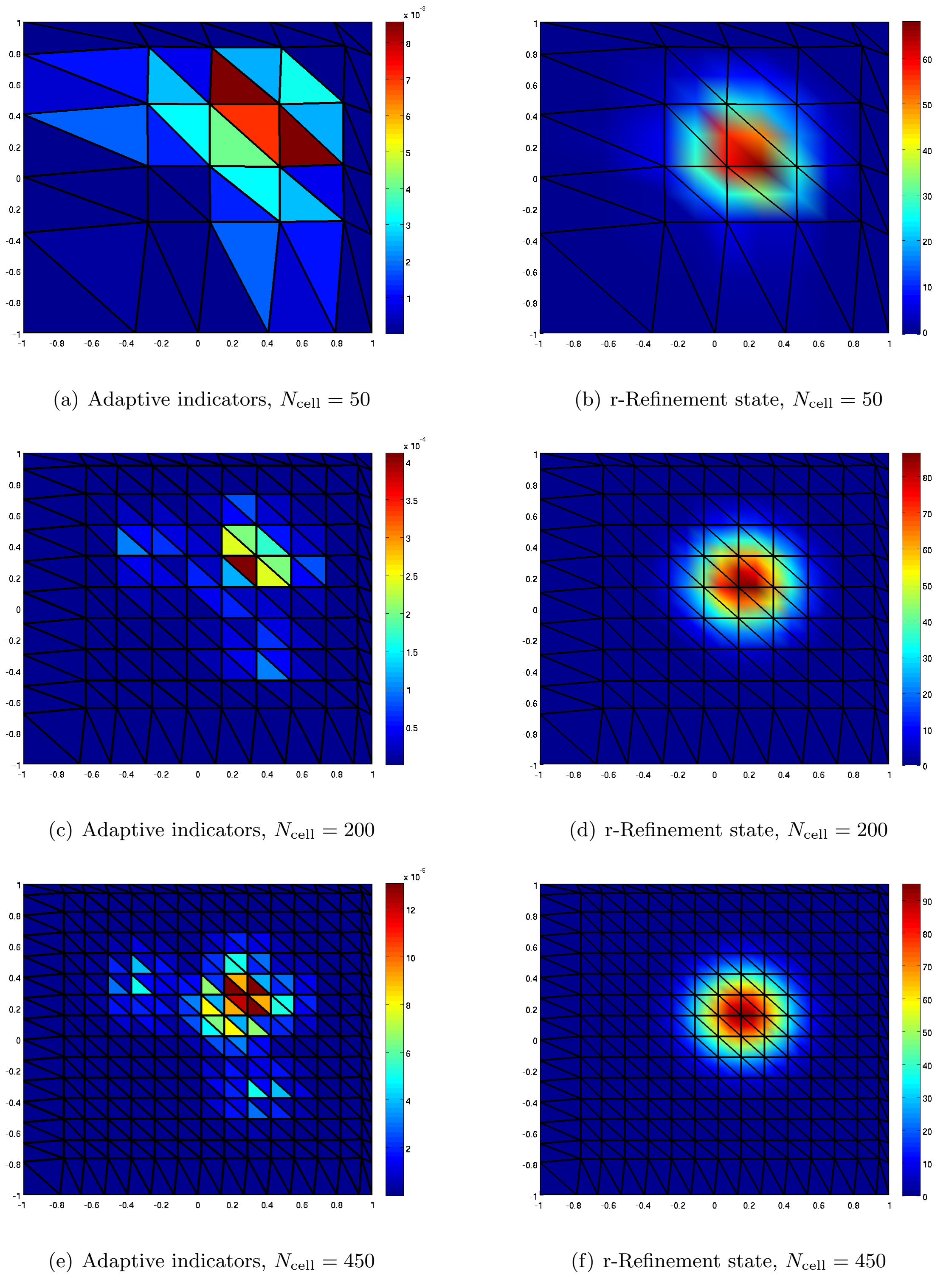

Primal($\mathbf{U}$) & Adjoint($\mathbf{\Psi}$) run on static mesh
Compute adaptive indicator map with prescribed r-refinement motion from last cycle
Adaptive indicators re-mapping on r-refined mesh
Solver run & r-refinement (spring analogy)
t=T?
no
yes
# Cycle = prescribed number?
no
yes
Solver finish

Figure 15. r-Refinement mechanics for the active-flux method.

B. r-Refinement Results



(a) Adaptive indicators, $N_{\text{cell}} = 50$

(b) r-Refinement state, $N_{\text{cell}} = 50$

(c) Adaptive indicators, $N_{\text{cell}} = 200$

(d) r-Refinement state, $N_{\text{cell}} = 200$

(e) Adaptive indicators, $N_{\text{cell}} = 450$

(f) r-Refinement state, $N_{\text{cell}} = 450$

Figure 16. Comparison of r-refinement adaptive indicators and r-refinement primal snapshots, both captured at $t = 0.8 \times$ (total simulation time) for the third r-refinement cycle. In this case, CFL = 0.7, and the simulation time = 0.1 period.

	$\lvert J_{\text{exact}} - J_H \rvert_{(\text{Static mesh solve})}$	$\lvert J_{\text{exact}} - J_H \rvert_{(\text{r refinement})}$
$N_{\text{cell}} = 50$	2.88346×10^{-1}	2.68664×10^{-2}
$N_{\text{cell}} = 200$	2.32982×10^{-2}	7.25353×10^{-3}
$N_{\text{cell}} = 450$	1.29288×10^{-2}	5.73822×10^{-3}

Table 3. Error comparison of output error on a static mesh solve and output error on a r-refinement solve, CFL = 0.7, simulation time = 0.1 period, the third cycle, $k = \dfrac{2}{\textbf{time step size}}$

The sequence of error indicators and the r-refinement history in Figure 16 shows clear convection dominance in this problem. The mesh nodes agglomerate diagonally, toward the same direction the scalar is moving (up and to the right). This means that the adaptive indicators captured some of the physics of the scalar advection problem. The error prone area is along the diagonal of the computational domain. If the mesh were printed at every time step of r-refinement, the mesh nodes would appear to move with the advecting scalar. This shows that the r-refinement strategy is dynamically adjusting the degrees of freedom towards the error prone area(s).

Table 3 shows the comparison of error levels on static meshes and r-refined meshes. With exactly the same number of degrees of freedom, r-refinement decreases the error level by more than 50% within three cycles of r-refinement.

IX. Conclusions

The authors' previous work has already investigated h-refinement with various adaptation strategies for the active flux scheme [11, 12] using static mesh refinement. In the present work, we demonstrated the active-flux method with mesh motion and dynamic mesh adaptation using r-refinement for unsteady simulations, driven by a continuous adjoint. The active-flux method with mesh motion was shown to be third-order accurate. Two key aspects of the r-refinement strategy were discussed: the error indicator mapping to create an adaptive indicator on an arbitrarily-deformed mesh, and the spring analogy for driving mesh motion. The final result shows that with these models, the error in the desired output could be quickly dropped by at least 50% within the first few r-refinement cycles.

Acknowledgments

The authors acknowledge support from the University of Michigan and from the National Aeronautics and Space Administration under grant number NNX12AJ70A.

References

[1]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.

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

[3]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.

[4]Duraisamy, K., Alonso, J., Palacios, F., and Chandrashekar, P., "Error estimation for high speed flows using continuous and discrete adjoints," AIAA Paper 2010-128, 2010.

[5]Eymann, T. A. and Roe, P. L., "Active flux schemes," 49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2011–382, 2011.

[6]Eymann, T. A. and Roe, P. L., "Active flux schemes for systems," 20th AIAA Computational Fluid Dynamics Conference 2011–3840, 2011.

[7]Eymann, T. A., *Active Flux Schemes*, Ph.D. thesis, The University of Michigan, Ann Arbor, 2013.

[8]Eymann, T. A. and Roe, P. L., “Multidimensional active flux schemes,” 21st AIAA Computational Fluid Dynamics Conference 2011–3840, 2013-2940.

[9]van Leer, B., “Towards the ultimate conservative difference scheme iv. a new approach to numerical convection,” *Journal of Computational Physics*, Vol. 23, 1977, pp. 276–299.

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

[11]Kaihua Ding, K. J. F. and Roe, P. L., “Adjoint-based error estimation and mesh adaptation for the active flux method,” 21st AIAA Computational Fluid Dynamics Conference AIAA 2013-2942, 2013.

[12]Kaihua Ding, K. J. F. and Roe, P. L., “Acceleration techniques for adjoint-based error estimation and mesh adaptation,” Eighth International Conference on Computational Fluid Dynamics (ICCFD8) ICCFD8-2014-0249, 2014.