ADAPTIVITY AND ERROR ESTIMATION FOR DISCONTINUOUS GALERKIN METHODS

SLIMANE ADJERID* AND MAHBOUB BACCOUCH†

Abstract. We test the a posteriori error estimates of discontinuous Galerkin (DG) discretization errors (Adjerid and Baccouch, J. Sci. Comput. 33(1):75–113, 2007; Adjerid and Baccouch, J. Sci. Comput. 38(1):15–49, 2008; Adjerid and Baccouch Comput. Methods Appl. Mech. Eng. 200:162–177, 2011) for hyperbolic problems on adaptively refined unstructured triangular meshes. A local error analysis allows us to construct asymptotically correct a posteriori error estimates by solving local hyperbolic problems on each element. The Taylor-expansion-based error analysis (Adjerid and Baccouch, J. Sci. Comput. 33(1):75–113, 2007; Adjerid and Baccouch, J. Sci. Comput. 38(1):15–49, 2008; Adjerid and Baccouch Comput. Methods Appl. Mech. Eng. 200:162–177, 2011) does not apply near discontinuities and shocks and lead to inaccurate estimates under uniform mesh refinement. Here, we present several computational results obtained from adaptive refinement computations that suggest that even in the presence of shocks our error estimates converge to the true error under adaptive mesh refinement. We also show the performance of several adaptive strategies for hyperbolic problems with discontinuous solutions.

Key words. Adaptive discontinuous Galerkin method, Hyperbolic problems, A posteriori error estimation, Unstructured meshes

AMS(MOS) subject classifications. Primary 65N30, 65N50.

1. Introduction.

The DG method was first developed for the neutron equation [20]. Since then, DG methods have been used to solve hyperbolic [11, 13–15, 17], parabolic [16], and elliptic [10] partial differential equations.

The DG methods are a family of locally conservative, stable, and high-order accurate methods that are easily coupled with other well-known methods and are well suited to adaptive strategies. For these reasons, they have attracted the attention of many researchers working in computational mechanics, computational mathematics, and computer science. They provide an appealing approach to address problems having discontinuities, such as those arising in hyperbolic conservation laws. The DG method does not require the approximate solutions to be continuous across element boundaries, it instead involves a flux term to account for the discontinuities. For a more complete list of citations on DG methods and its applications, consult [12]. A main advantage of using discontinuous finite element basis is to simplify adaptive p- and h-refinement with hanging nodes.

The DG method has a simple communication pattern between elements with a common face that makes it useful for parallel computation.

*Department of Mathematics, Virginia Tech, Blacksburg, VA 24061, USA, adjerids@vt.edu

†Department of Mathematics, University of Nebraska, Omaha, NE 68182, USA, mbaccouch@unomaha.edu

X. Feng et al. (eds.), *Recent Developments in Discontinuous Galerkin Finite Element Methods for Partial Differential Equations*, The IMA Volumes in Mathematics and its Applications 157, DOI 10.1007/978-3-319-01818-8_3,
© Springer International Publishing Switzerland 2014

Furthermore, it can handle problems with complex geometries to high order. Regardless of the type of DG method, we need to know how well our computed solution approximates the exact solution. In practice, the exact solution of the problem is not available and a method to estimate the discretization error is needed. For these reasons, a posteriori error estimates have been developed for DG methods and provide some initial guidance for deciding on the degree of the approximation and the size of the mesh that guarantee a prescribed level of accuracy. Furthermore, error estimates may be used to guide *hp*-adaptive refinement.

The first superconvergence result for DG solutions of hyperbolic partial differential equations appeared in Adjerid et al. [4]. The authors showed that DG solutions of one-dimensional linear and nonlinear hyperbolic problems using p-degree polynomial approximations exhibit an $O(h^{p+2})$ superconvergence rate at the roots of $(p+1)$-degree Radau polynomial. This lead to the conclusion that the leading term of the DG error on each element is proportional to $(p+1)$-degree Radau polynomial which was used to construct asymptotically correct a posteriori error estimates. They further established a strong $O(h^{2p+1})$ superconvergence at the downwind end of every element. Later, Krivodonova and Flaherty [19] showed that the leading term of the local discretization error on triangles having one *outflow* edge is spanned by a suboptimal set of orthogonal polynomials of degree p and $p+1$ and computed DG error estimates by solving local problems involving numerical fluxes, thus requiring information from neighboring *inflow* elements. Adjerid and Massey [5] extended these results to multi-dimensional problems using rectangular meshes and presented an error analysis for linear and nonlinear hyperbolic problems. They showed that the leading term in the true local error is spanned by two $(p+1)$-degree Radau polynomials in the x and y directions, respectively. They further discovered that a p-degree discontinuous finite element solution exhibits an $O(h^{p+2})$ superconvergence at Radau points obtained as a tensor product of the roots of Radau polynomial of degree $p+1$. Using these results, they were able to compute asymptotically exact a posteriori error estimates for linear and nonlinear hyperbolic problems on Cartesian meshes.

Adjerid and Baccouch [1, 2] investigated the superconvergence properties of discontinuous Galerkin solutions of a scalar first-order hyperbolic problem on triangular meshes. They presented a detailed discussion on the superconvergence properties versus the choice of finite element polynomial spaces. First, they classified triangular elements into three types: (i) type I with one *inflow* edge and two *outflow* edges, (ii) type II with two *inflow* edges and one *outflow* edge, and (iii) type III with one *inflow* edge, one *outflow* edge, one edge parallel to the characteristics. Through computations, they showed that the local superconvergence results [1] extend to global DG solutions on general meshes with a corrected *inflow* boundary condition. In particular, they showed that the discontinuous finite element solution is $O(h^{p+2})$ superconvergent at the Legendre points on the outflow edge for triangles having one outflow edge. For triangles having two outflow

edges the finite element error is $O(h^{p+2})$ superconvergent at the end points of the inflow edge.

Adjerid and Weinhart [7–9] studied the asymptotic behavior of the local DG error for multi-dimensional first-order linear symmetric and symmetrizable hyperbolic systems of partial differential equations. They performed a local error analysis by writing the local error as a series and showing that its leading term can be expressed as a linear combination of Legendre polynomials of degree p and $p+1$. They were able to compute efficient and asymptotically exact estimates of the discontinuous finite element error.

In this manuscript we consider the modified discontinuous Galerkin method [2] with a corrected inflow flux and an enriched polynomial space $\mathcal{U}_p$ with adaptive mesh refinement. We consider several mesh refinement strategies guided by both discretization error estimates and local residuals. Since $\mathcal{L}^2$ a posteriori error estimates based on Taylor expansions fail to be asymptotically exact under uniform mesh refinement in the presence of shocks [3, 5, 6], we present several numerical results which suggest that such error estimates converge to the true errors under adaptive mesh refinement on general unstructured triangular meshes in the presence of discontinuities. Numerical results further suggest that using local residuals to guide the adaptive mesh refinement yield more efficient algorithms when compared to using the error estimate itself in the presence of discontinuities. Thus, we recommend an adaptive strategy that combines the local residuals or any other explicit estimators to guide mesh refinement and the proposed error estimate to assess solution accuracy and terminate the adaptive refinement process.

This paper is organized as follows: In Sect. 2 we state the modified DG formulation for linear and nonlinear hyperbolic problem and present our a posteriori error estimation procedures for linear and nonlinear problems. In Sect. 3 we describe several adaptive mesh refinement strategies that will be used to test the performance of our error estimates. Finally, in Sect. 4 we present numerical results for several linear and nonlinear hyperbolic problems with discontinuous solutions and conclude with a few remarks in Sect. 5.

2. Discontinuous Galerkin Formulation and Error Estimation. In this section we present an adaptive modified discontinuous Galerkin method [1–3] combined with a posteriori error estimation procedure. In addition to being used to steer the adaptive process, the a posteriori error estimate is also used to correct the numerical flux needed to compute the DG solution on downwind elements. This modified DG method maintains the structure of the local discretization error on each element of the mesh which makes the error estimation both efficient and accurate.

In order to further simplify the error estimation procedure we use the augmented polynomial space $\mathcal{U}_p$ given by

$$\mathcal{U}_p = \mathcal{P}_p \cup \text{span}\{x^{p+1}, y^{p+1}\},\ p \geq 1,$$

where

$$\mathcal{P}_k = \left\{ q \,|\, q = \sum_{m=0}^{k} \sum_{i=0}^{m} c_i^m x^i y^{m-i} \right\}, \qquad k = 0, 1, \ldots, p. \tag{2.1a}$$

The finite element space $\mathcal{U}_p$ is suboptimal, *i.e.*, it contains $p+1$-degree terms that do not contribute to global convergence rate but simplifies the a posteriori error estimation procedures described later in this manuscript.

We consider a reference triangle Δ_0 defined by the vertices $(0,0)$, $(1,0)$, and $(0,1)$ and define the following orthogonal polynomials [18]

$$\varphi_k^l(\xi,\eta) = 2^k \hat{L}_k\left(\frac{2\xi}{1-\eta} - 1\right)(1-\eta)^k \hat{P}_l^{2k+1,0}(2\eta - 1),\ k, l \geq 0\ k+l = p \geq 0, \tag{2.2a}$$

where $\hat{P}_n^{\alpha,\beta}(x),\ -1 \leq x \leq 1$, is the Jacobi polynomial

$$\hat{P}_n^{\alpha,\beta}(x) = \frac{(-1)^n}{2^n n!}(1-x)^{-\alpha}(1+x)^{-\beta}\frac{d^n}{dx^n}[(1-x)^{\alpha+n}(1+x)^{\beta+n}], \quad \alpha, \beta > -1, \tag{2.2b}$$

and $\hat{L}_n(x) = \hat{P}_n^{0,0}(x),\ -1 \leq x \leq 1$ is the nth-degree Legendre polynomial.

We note that these polynomials satisfy the $\mathcal{L}^2$ orthogonality

$$\int_0^1 \int_0^{1-\eta} \varphi_k^l \varphi_p^q d\xi d\eta = c_{kp}^{lq} \delta_{kp} \delta_{lq}. \tag{2.3}$$

Radau polynomials are defined by

$$\hat{R}_{p+1}(x) = (1-x)\hat{P}_p^{1,0}(x) = C(\hat{L}_{p+1}(x) - \hat{L}_p(x)),\ -1 \leq x \leq 1. \tag{2.4}$$

We drop the hat and let L_p, $P_p^{\alpha,\beta}$ and R_p, respectively, denote the Jacobi, Legendre and Radau polynomials shifted to $[0,1]$.

Let us note that U on the physical element is in the modified polynomial space

$$\mathcal{U}_p = \mathcal{P}_p + \text{span}\{\xi(x,y)^{p+1}, \eta(x,y)^{p+1}\},$$

where $(x,y) \to (\xi(x,y), \eta(x,y))$ is the standard affine mapping from the physical element Δ to the reference element Δ_0.

First, let us consider a divergence-free linear stationary hyperbolic problem on an open bounded convex polygonal domain $\Omega \subseteq R^2$. Let $\boldsymbol{a} = [a_1(x,y), a_2(x,y)]^T$ denote a nonzero velocity vector. If $\boldsymbol{n}$ denotes the outward unit normal vector, the domain boundary $\partial\Omega = \partial\Omega^+ \cup \partial\Omega^- \cup \partial\Omega^0$,

where the inflow, outflow, and characteristic boundaries, respectively, are $\partial\Omega^- = \{(x,y) \in \partial\Omega \mid \boldsymbol{a}\cdot\boldsymbol{n} < 0\}$, $\partial\Omega^+ = \{(x,y) \in \partial\Omega \mid \boldsymbol{a}\cdot\boldsymbol{n} > 0\}$, and $\partial\Omega^0 = \{(x,y) \in \partial\Omega \mid \boldsymbol{a}\cdot\boldsymbol{n} = 0\}$.
Let $u(x,y)$ denote a smooth function on Ω and consider the following hyperbolic boundary value problem

$$L(u) = \boldsymbol{a}.\nabla u + cu = f, \quad (x,y) \in \Omega = (0,1)^2, \tag{2.5a}$$

$$\nabla.\boldsymbol{a} = \frac{\partial a_1}{\partial x} + \frac{\partial a_2}{\partial y} = 0, \tag{2.5b}$$

subject to the boundary conditions

$$u(x,0) = g_0(x), \quad u(0,y) = g_1(y), \tag{2.5c}$$

where the functions $\boldsymbol{a}(x,y)$, $c(x,y)$, $f(x,y)$, $g_0(x)$, and $g_1(y)$ are selected such that the exact solution $u(x,y) \in C^\infty(\Omega)$.

In order to obtain the weak discontinuous Galerkin formulation for (2.5), we partition the domain Ω into N triangular elements Δ_j, $j = 1,\ldots,N$, such that on every edge $\boldsymbol{a}\cdot\boldsymbol{n}$ does not change sign. Thus, every edge is either *inflow*, *outflow*, or *characteristic*, respectively, if $\boldsymbol{a}\cdot\boldsymbol{n} < 0$, $\boldsymbol{a}\cdot\boldsymbol{n} > 0$ or $\boldsymbol{a}\cdot\boldsymbol{n} = 0$. Using this mesh orientation, a triangle can be classified into type I having one *inflow* edge and 2 *outflow* edges, type II having two *outflow* and one *inflow* edges, or type III having one *inflow*, one *outflow* and one *characteristic* edges. The problem is solved on each element starting from the upwind elements and proceeding to the neighboring elements in the downwind direction, i.e., we order the elements such that the inflow boundary Γ_j^- of an element Δ_j is contained in the inflow boundary $\partial\Omega^-$ of the domain or in the outflow boundary Γ_i^+ of Δ_i, $i < j$.

In the remainder of this paper we omit the element index and refer to an arbitrary element by Δ whenever confusion is unlikely. Note that Γ^+ and Γ^-, respectively, denote the outflow and inflow boundaries of Δ.

Thus, our discontinuous Galerkin method consists of finding $U \in \mathcal{U}_p$ such that

$$\int_{\Gamma^-} \boldsymbol{a}\cdot\boldsymbol{n}\hat{U}V\,ds + \int_{\Gamma^+} \boldsymbol{a}\cdot\boldsymbol{n}UV\,ds - \iint_\Delta (\boldsymbol{a}\cdot\nabla V - cV)U\,dxdy = \iint_\Delta fV\,dxdy, \quad \forall\, V \in \mathcal{U}_p. \tag{2.6a}$$

In order to complete the definition of the DG method we need to select the corrected upwind numerical flux $\hat{U}$ on Γ^- as

$$\hat{U} = \begin{cases} u, & \text{on } \Gamma^- \bigcap \partial\Omega^- \\ U^- + E^-, & \text{elsewhere,} \end{cases} \tag{2.6b}$$

where U^- and E^-, respectively, are the limit of U and E from the *inflow* element sharing Γ^- , i.e., if $(x,y) \in \Gamma^-$, then

$$U^-(x,y) = \lim_{s\to 0^+} U((x,y) + s\boldsymbol{n}), E^-(x,y) = \lim_{s\to 0^+} E((x,y) + s\boldsymbol{n}). \tag{2.6c}$$

Here, E is an a posteriori error estimate that will be defined by following [2, 3] to write the leading term of the local DG error on each triangle as a linear combination of the following p error basis functions

$$(u-U)(x,y) \approx E(x,y) = \sum_{i=1}^{p} d^i \chi^i_{p+1-i}(\xi(x,y),\eta(x,y)), \tag{2.7a}$$

where χ_j are given in [2]. For the sake of completeness we include them in Tables 1 and 2 for types I and II and $[\alpha,\beta]$ as given in (2.11) while for type III the leading term of the error can be written as

$$Q_{p+1} = \sum_{\substack{i,j\geq 0\\ i+j=p}} c_i^j\,(1-\xi-\eta)P_i^{2j+2,0}(2\eta-1)(1-\eta)^j P_j^{1,0}\left(\frac{2\xi}{1-\eta}-1\right), \tag{2.7b}$$

with

$$c_0^p = \frac{1}{p+1}\sum_{i=1}^{p}(-1)^{i+1}(p+1-i)c_i^{p-i}. \tag{2.7c}$$

After computing the finite element solution U on an element Δ, we compute an error estimate by solving the problem for $i = 1, 2, \ldots, p$,

$$\iint_{\Delta} \left(\boldsymbol{a}\cdot\nabla(U+E) + c(U+E)\right)\chi^i_{p+1-i}dxdy = \iint_{\Delta} f\chi^i_{p+1-i}dxdy. \tag{2.7d}$$

Next we consider nonlinear problems of the form

$$L(u) = \nabla\cdot\mathbf{F}(u) = h(u)_x + g(u)_y = f(x,y), \quad (x,y)\in\Omega \;=\; (0,1)^2, \tag{2.8a}$$

subject to the boundary conditions

$$u(x,0) = g_0(x), \quad u(0,y) = g_1(y). \tag{2.8b}$$

In our analysis [3] we assume $\mathbf{F}:\mathbb{R}\to\mathbb{R}^2$, $u:\mathbb{R}^2\to\mathbb{R}$, f, g_0 and g_1 to be analytic functions such that $g'(u) > 0$ and $h'(u) > 0$ over the domain Ω. The *inflow*, *outflow*, and *characteristic* boundaries, respectively, are defined by $\partial\Omega^- = \{(x,y)\in\partial\Omega \mid \mathbf{F}'(u)\cdot\boldsymbol{n} = [h'(u),g'(u)]^t\cdot\boldsymbol{n} < 0\}$, $\partial\Omega^+ = \{(x,y)\in\partial\Omega \mid \mathbf{F}'(u)\cdot\boldsymbol{n} > 0\}$, and $\partial\Omega^0 = \{(x,y)\in\partial\Omega \mid \mathbf{F}'(u)\cdot\boldsymbol{n} = 0\}$, such that $\partial\Omega = \partial\Omega^-\cup\partial\Omega^+\cup\partial\Omega^0$ and $\boldsymbol{n}$ is the outward unit normal to $\partial\Omega$. We further assume that the unstructured triangular mesh is such that $\mathbf{F}'(u).\boldsymbol{n}$ does not change sign on all edges, i.e., every edge is either *inflow*, *outflow*, or *characteristic*.

The discrete DG formulation consists of determining $U\in\mathcal{U}_p$ such that

$$\begin{aligned}\int_{\Gamma^-}\boldsymbol{n}\cdot\mathbf{F}(\hat{U})V ds + \int_{\Gamma^+}\boldsymbol{n}\cdot\mathbf{F}(U)V ds - \iint_{\Delta}\mathbf{F}(U)\cdot\nabla V dxdy\\ = \iint_{\Delta} fV dxdy, \quad \forall\, V\in\mathcal{U}_p,\end{aligned} \tag{2.9a}$$

TABLE 1
Error basis functions for the spaces $\mathcal{U}_p$ for $p = 1$ to 3 on the reference triangle of type I where $s = \alpha/\beta$

$p = 1$
$\chi_1^1 = (12(\varphi_1^0 + \varphi_1^1)s^2 + 2(10\varphi_0^0 - 2\varphi_1^0 - 3\varphi_1^0 - 2\varphi_0^2 + 12\varphi_1^1 + 5\varphi_2^0)s + 12\varphi_1^0 + 6\varphi_1^0 - 8\varphi_0^2 + 6\varphi_1^1 + 5\varphi_2^0)/10s + 5$
$p = 2$
$\chi_2^1 = (75(2\varphi_1^1 + \varphi_2^0 + 2\varphi_1^2 + \varphi_2^1)s^4 + 5(28\varphi_0^0 + 28\varphi_0^1 - 42\varphi_0^1 - 12\varphi_0^2 + 54\varphi_1^1 + 51\varphi_2^0 - 12\varphi_0^3 + 96\varphi_1^2 + 51\varphi_2^1)s^3 + (-140\varphi_0^0 + 532\varphi_0^1 + 126\varphi_1^0 - 68\varphi_0^2 + 36\varphi_1^1 + 140\varphi_2^0 - 180\varphi_0^3 + 540\varphi_1^2 + 315\varphi_2^1)s^2 + (-140\varphi_0^0 + 280\varphi_0^1 + 168\varphi_1^0 + 100\varphi_0^2 - 12\varphi_1^1 - 10\varphi_2^0 - 180\varphi_0^3 + 240\varphi_1^2 + 165\varphi_2^1)s + 5(16\varphi_0^2 + 6\varphi_1^1 - \varphi_2^0 - 12\varphi_0^3 + 6\varphi_1^2 + 6\varphi_2^1))/15(s+1)^3(5s+2)$ $\chi_1^2 = (-240(\varphi_1^1 + \varphi_1^2)s^5 + (-448\varphi_0^0 - 112\varphi_0^1 + 840\varphi_1^0 + 128\varphi_0^2 - 780\varphi_1^1 - 350\varphi_2^0 + 72\varphi_0^3 - 1200\varphi_1^2 + 105\varphi_3^0)s^4 - 21(24\varphi_0^0 + 56\varphi_0^1 - 68\varphi_1^0 - 24\varphi_0^2 + 12\varphi_1^1 + 30\varphi_2^0 - 16\varphi_0^3 + 108\varphi_1^2 - 17\varphi_3^0)s^3 + (616\varphi_0^0 - 2072\varphi_0^1 - 84\varphi_1^0 + 408\varphi_0^2 + 936\varphi_1^1 + 576\varphi_0^3 - 2004\varphi_1^2 + 441\varphi_3^0)s^2 + (392\varphi_0^0 - 784\varphi_0^1 - 336\varphi_1^0 - 184\varphi_0^2 + 624\varphi_1^1 + 280\varphi_2^0 + 432\varphi_0^3 - 804\varphi_1^2 + 231\varphi_3^0)s + 2(-80\varphi_0^2 + 30\varphi_1^1 + 35\varphi_2^0 + 60\varphi_0^3 - 54\varphi_1^2 + 21\varphi_3^0))/21(s+1)^3(5s+2)$
$p = 3$
$\chi_3^1 = (420(2\varphi_1^3 + \varphi_2^2 + 2\varphi_1^2 + \varphi_2^1)s^5 + 4(84\varphi_0^0 + 228\varphi_0^1 - 80\varphi_0^4 + 870\varphi_1^3 + 455\varphi_2^2 - 54\varphi_1^0 + 108\varphi_0^2 - 324\varphi_1^1 - 116\varphi_0^3 + 600\varphi_1^2 + 455\varphi_2^1)s^4 + (-2352\varphi_0^0 + 2976\varphi_0^1 - 1280\varphi_0^4 + 5520\varphi_1^3 + 3080\varphi_2^2 + 4608\varphi_1^0 + 2736\varphi_0^2 - 2592\varphi_1^1 - 2520\varphi_2^0 - 1352\varphi_0^3 + 1596\varphi_1^2 + 2450\varphi_2^1 + 441\varphi_3^0)s^3 + 6(8\varphi_0^0 - 232\varphi_0^1 - 320\varphi_0^4 + 680\varphi_1^3 + 420\varphi_2^2 + 156\varphi_1^0 + 888\varphi_0^2 + 216\varphi_1^1 - 240\varphi_2^0 - 152\varphi_0^3 - 112\varphi_1^2 + 120\varphi_2^1 + 63\varphi_3^0)s^2 + (720\varphi_0^0 - 1440\varphi_0^1 - 1280\varphi_0^4 + 1320\varphi_1^3 + 980\varphi_2^2 - 864\varphi_1^0 + 2160\varphi_0^2 + 1296\varphi_1^1 + 520\varphi_2^0 - 372\varphi_0^3 - 190\varphi_1^2 + 63\varphi_3^0)s - 20(16\varphi_0^2 - 6\varphi_1^3 - 7\varphi_2^2 - 20\varphi_0^3 - 6\varphi_1^2 + 2\varphi_2^1))/140(s+1)^4(3s+1)$ $\chi_2^2 = (-168(4\varphi_1^3 - \varphi_3^1 + 4\varphi_1^2 - \varphi_3^0)s^5 - 4(168\varphi_0^0 + 240\varphi_0^1 - 40\varphi_0^4 + 708\varphi_1^3 - 182\varphi_3^1 - 216\varphi_1^0 - 486\varphi_1^1 + 135\varphi_2^0 - 112\varphi_0^3 + 438\varphi_1^2 + 135\varphi_2^1 - 182\varphi_3^0)s^4 + 2(1512\varphi_0^0 - 2088\varphi_0^1 + 320\varphi_0^4 - 2304\varphi_1^3 + 616\varphi_3^1 - 2916\varphi_1^0 - 648\varphi_0^2 + 2484\varphi_1^1 + 1350\varphi_2^0 + 752\varphi_0^3 - 180\varphi_1^2 - 540\varphi_2^1 + 175\varphi_3^0)s^3 + 3(48\varphi_0^0 + 288\varphi_0^1 + 320\varphi_0^4 - 1184\varphi_1^3 + 336\varphi_3^1 - 576\varphi_1^0 - 1392\varphi_0^2 + 144\varphi_1^1 + 600\varphi_2^0 + 488\varphi_0^3 + 724\varphi_1^2 + 30\varphi_2^1 - 21\varphi_3^0)s^2 + (-528\varphi_0^0 + 1056\varphi_0^1 + 640\varphi_0^4 - 1248\varphi_1^3 + 392\varphi_3^1 + 432\varphi_1^0 - 1584\varphi_0^2 - 648\varphi_1^1 + 180\varphi_2^0 - 8\varphi_0^3 + 1236\varphi_1^2 + 450\varphi_2^1 - 49\varphi_3^0)s + 160\varphi_0^4 - 144\varphi_1^3 + 56\varphi_3^1 - 200\varphi_0^3 + 108\varphi_1^2 + 90\varphi_2^1 - 7\varphi_3^0)/56(s+1)^4(3s+1)$ $\chi_1^3 = (3360(\varphi_1^3 + \varphi_1^2)s^6 + 14(384\varphi_0^0 + 240\varphi_0^1 - 64\varphi_0^4 + 1440\varphi_1^3 + 54\varphi_4^0 - 648\varphi_1^0 - 648\varphi_1^1 + 540\varphi_2^0 - 136\varphi_0^3 + 1116\varphi_1^2 + 270\varphi_2^1 - 189\varphi_3^0)s^5 + (-6720\varphi_0^0 + 29280\varphi_0^1 - 5216\varphi_0^4 + 48240\varphi_1^3 + 3276\varphi_4^0 + 13968\varphi_1^0 + 4320\varphi_0^2 - 44712\varphi_1^1 + 3780\varphi_2^0 - 9680\varphi_0^3 + 22320\varphi_1^2 + 17640\varphi_2^1 - 4410\varphi_3^0)s^4 - 6(4984\varphi_0^0 - 6248\varphi_0^1 + 1984\varphi_0^4 - 9760\varphi_1^3 - 924\varphi_4^0 - 10452\varphi_1^0 - 4728\varphi_0^2 + 9108\varphi_1^1 + 3990\varphi_2^0 + 2944\varphi_0^3 - 196\varphi_1^2 - 4200\varphi_2^1 - 441\varphi_3^0)s^3 + (144\varphi_0^0 - 11232\varphi_1^0 - 13376\varphi_0^4 + 37440\varphi_1^3 + 4536\varphi_4^0 + 16416\varphi_1^0 + 44208\varphi_0^2 - 5184\varphi_1^1 - 14400\varphi_2^0 - 10856\varphi_0^3 - 18900\varphi_1^2 + 11610\varphi_2^1 + 5985\varphi_3^0)s^2 + (4848\varphi_0^0 - 9696\varphi_1^0 - 7424\varphi_0^4 + 11520\varphi_1^3 + 1764\varphi_4^0 - 3600\varphi_1^0 + 14544\varphi_0^2 + 5400\varphi_1^1 - 1260\varphi_2^0 + 2008\varphi_0^3 - 9180\varphi_1^2 + 1530\varphi_2^1 + 2835\varphi_3^0)s + 3(-544\varphi_0^4 + 400\varphi_1^3 + 84\varphi_4^0 + 680\varphi_0^3 - 188\varphi_1^2 + 30\varphi_2^1 + 147\varphi_3^0))/252(s+1)^4(3s+1)$

where $\hat{U}$ is as defined in (2.6b). We estimate the error by solving the linearized problem

$$\iint_{\Delta} [h'(U), g'(U)]^T \cdot \nabla(U + E)\chi_{p+1-i}^i dxdy = \iint_{\Delta} f\chi_{p+1-i}^i dxdy, \quad i = 1, \ldots, p. \tag{2.9b}$$

TABLE 2
Error basis functions for the spaces $\mathcal{U}_p$ for $p = 1$ to 4 on the reference element of type II, $s = \alpha/\beta$

$p = 1$
$\chi_1^1 = ((-24\varphi_0^1 - 12\varphi_1^0 + 16\varphi_0^2 + 18\varphi_1^1 + 5\varphi_2^0)s^2 - 3(8\varphi_0^1 + 12\varphi_1^0 + 8\varphi_0^2 - 18\varphi_1^1 - 5\varphi_2^0)s - 24\varphi_1^0 + 6\varphi_1^1 + 15\varphi_2^0)5(s^2 + 3s + 3)$
$p = 2$
$\chi_2^1 = -((32\varphi_0^2 + 18\varphi_1^1 + \varphi_2^0 - 24\varphi_0^3 - 24\varphi_2^1 - 6\varphi_3^0)s^2 + 4(8\varphi_0^2 + 18\varphi_1^1 + \varphi_2^0 + 8\varphi_0^3 - 24\varphi_2^1 - 6\varphi_3^0)s + 6(10\varphi_1^1 + \varphi_2^0 - 4\varphi_2^1 - 6\varphi_3^0))6(s^2 + 4s + 6)$ $\chi_1^2 = ((320\varphi_0^2 + 90\varphi_1^1 - 35\varphi_2^0 - 240\varphi_0^3 - 204\varphi_2^1 + 21\varphi_1^2)s^2 + 4(80\varphi_0^2 + 90\varphi_1^1 - 35\varphi_2^0 + 80\varphi_0^3 - 204\varphi_2^1 + 21\varphi_1^2)s + 6(10\varphi_1^1 - 35\varphi_2^0 - 4\varphi_2^1 + 21\varphi_1^2))21(s^2 + 4s + 6)$
$p = 3$
$\chi_3^1 = ((32\varphi_0^4 + 30\varphi_1^3 + 7\varphi_2^2 - 40\varphi_0^3 - 24\varphi_2^1 - 2\varphi_3^0)s^2 - 5(8\varphi_0^4 - 30\varphi_1^3 - 7\varphi_2^2 + 8\varphi_0^3 + 24\varphi_2^1 + 2\varphi_3^0)s + 10(6\varphi_1^3 + 7\varphi_2^2 - 2(6\varphi_2^1 + \varphi_3^0)))7(s^2 + 5s + 10)$ $\chi_2^2 = ((-800\varphi_0^4 - 624\varphi_1^3 + 56\varphi_3^1 + 1000\varphi_0^3 + 348\varphi_2^1 - 90\varphi_3^0 - 7\varphi_1^2)s^2 + 5(200\varphi_0^4 - 624\varphi_1^3 + 56\varphi_3^1 + 200\varphi_0^3 + 348\varphi_2^1 - 90\varphi_3^0 - 7\varphi_1^2)s - 10(24\varphi_1^3 - 56\varphi_3^1 - 48\varphi_2^1 + 90\varphi_3^0 + 7\varphi_1^2))56(s^2 + 5s + 10)$ $\chi_1^3 = ((2656\varphi_0^4 + 2000\varphi_1^3 + 84\varphi_4^0 - 3320\varphi_0^3 - 1012\varphi_2^1 + 30\varphi_3^0 - 147\varphi_1^2)s^2 - 5(664\varphi_0^4 - 2000\varphi_1^3 - 84\varphi_4^0 + 664\varphi_0^3 + 1012\varphi_2^1 - 30\varphi_3^0 + 147\varphi_1^2)s + 10(8\varphi_1^3 + 84\varphi_4^0 - 16\varphi_2^1 + 30\varphi_3^0 - 147\varphi_1^2))84(s^2 + 5s + 10)$
$p = 4$
$\chi_4^1 = ((-48\varphi_0^4 - 30\varphi_1^3 - 3\varphi_2^2 + 40\varphi_0^5 + 36\varphi_1^4 + 8\varphi_2^3)s^2 - 6(8\varphi_0^4 + 30\varphi_1^3 + 3\varphi_2^2 + 8\varphi_0^5 - 36\varphi_1^4 - 8\varphi_2^3)s - 15(14\varphi_1^3 + 3\varphi_2^2 - 8(\varphi_1^4 + \varphi_2^3)))8(s^2 + 6s + 15)$ $\chi_3^2 = ((720\varphi_0^4 + 282\varphi_1^3 - 55\varphi_2^2 - 8\varphi_3^1 - 600\varphi_0^5 - 444\varphi_1^4 + 36\varphi_3^2)s^2 + 6(120\varphi_0^4 + 282\varphi_1^3 - 55\varphi_2^2 - 8\varphi_3^1 + 120\varphi_0^5 - 444\varphi_1^4 + 36\varphi_3^2)s + 15(42\varphi_1^3 - 55\varphi_2^2 - 8\varphi_3^1 - 24\varphi_1^4 + 36\varphi_3^2))36(s^2 + 6s + 15)$ $\chi_2^3 = ((-4176\varphi_0^4 - 1434\varphi_1^3 + 55\varphi_2^2 - 154\varphi_3^1 - 9\varphi_4^0 + 3480\varphi_0^5 + 2460\varphi_1^4 + 90\varphi_4^1)s^2 - 6(696\varphi_0^4 + 1434\varphi_1^3 - 55\varphi_2^2 + 154\varphi_3^1 + 9\varphi_4^0 + 696\varphi_0^5 - 2460\varphi_1^4 - 90\varphi_4^1)s - 15(42\varphi_1^3 - 55\varphi_2^2 + 154\varphi_3^1 + 9\varphi_4^0 - 24\varphi_1^4 - 90\varphi_4^1))90(s^2 + 6s + 15)$ $\chi_1^4 = ((47376\varphi_0^4 + 15834\varphi_1^3 - 55\varphi_2^2 + 154\varphi_3^1 - 891\varphi_4^0 - 39480\varphi_0^5 - 27660\varphi_1^4 + 495\varphi_5^0)s^2 + 6(7896\varphi_0^4 + 15834\varphi_1^3 - 55\varphi_2^2 + 154\varphi_3^1 - 891\varphi_4^0 + 7896\varphi_0^5 - 27660\varphi_1^4 + 495\varphi_5^0)s + 15(42\varphi_1^3 - 55\varphi_2^2 + 154\varphi_3^1 - 891\varphi_4^0 - 24\varphi_1^4 + 495\varphi_5^0))495(s^2 + 6s + 15)$

Next, we consider transient hyperbolic problems of the form

$$L(u) = u_t + \nabla \cdot \mathbf{F}(u) = f(x, y, t), \quad (x, y) \in \Omega \ = \ (0, 1)^2, \ t > 0,$$

subject to initial condition $u_0(x, y)$ and inflow boundary conditions.

The semi-discrete DG formulation consists of determining $U \in \mathcal{U}_p$ such that

$$\iint_{\Delta}(U_t V - \mathbf{F}(U)\cdot\nabla V)dxdy + \int_{\Gamma^-} \boldsymbol{n}\cdot\mathbf{F}(\hat{U})Vds + \int_{\Gamma^+} \boldsymbol{n}\cdot\mathbf{F}(U)Vds = \iint_{\Delta} fVdxdy, \quad \forall\ V \in \mathcal{U}_p, \tag{2.10a}$$

where $\hat{U}$ is as defined in (2.6b). We compute an error estimate by solving the linearized problem

$$\iint_{\Delta}[h'(U), g'(U)]^T \cdot \nabla(U+E)\chi^k_{p+1-k}dxdy = \iint_{\Delta}(f - U_t)\chi^k_{p+1-k}dxdy, \quad k = 1, 2, \ldots, p. \tag{2.10b}$$

In order for the DG error to have the same structure for all times, we approximate the initial conditions u_0 by $U_0 \in \mathcal{U}_p$ computed from the stationary problem

$$\int_{\Gamma^-} \boldsymbol{n}\cdot\mathbf{F}(\hat{U}_0)Vds + \int_{\Gamma^+} \boldsymbol{n}\cdot\mathbf{F}(U_0)Vds - \iint_{\Delta}\mathbf{F}(U_0)\cdot\nabla V dxdy = \iint_{\Delta}\nabla\cdot\mathbf{F}(u_0)Vdxdy, \quad \forall\ V \in \mathcal{U}_p, \tag{2.10c}$$

with $\hat{U}_0$ being u_0 on the boundary edges and $U_0 + E_0$ on the interior edges. We estimate the error E_0 at $t = 0$ on Δ by solving the linearized problem

$$\iint_{\Delta}[h'(U_0), g'(U_0)]^T \cdot \nabla(U_0+E_0)\chi^i_{p+1-i}dxdy = \iint_{\Delta}\nabla\cdot\mathbf{F}(u_0)\chi^i_{p+1-i}dxdy, \quad i = 1, 2, \ldots, p. \tag{2.10d}$$

Basic calculus shows that, if $h = \mathrm{diam}(\Delta)$, the Jacobian of the affine transformation from Δ to Δ_0 can be written as

$$\mathbf{J} = \begin{bmatrix}\xi_x & \eta_x \\ \xi_y & \eta_y\end{bmatrix} = \frac{1}{h}\mathbf{J}_0,$$

where $\mathbf{J}_0$ is a 2×2 matrix independent of h.

Applying Taylor's theorem we expand $\mathbf{J}_0\boldsymbol{a}$ as

$$\check{\boldsymbol{a}}(\xi,\eta,h) = \boldsymbol{a}_0 + \sum_{k=1}^{\infty} h^k \boldsymbol{a}_k(\xi,\eta),$$

where $\boldsymbol{a}_k \in [\mathcal{P}_k]^2$, and

$$\boldsymbol{a}_0 = [\alpha,\beta]^T = \begin{cases}\mathbf{J}_0\tilde{\boldsymbol{a}}(1/2,1/2), & \text{if } \Delta \text{ is of type I,}\\ \mathbf{J}_0\tilde{\boldsymbol{a}}(0,0), & \text{if } \Delta \text{ is of type II or III}\end{cases}.$$

The sign of $[\alpha,\beta]^T\cdot\mathbf{n}$ is used to determine inflow and outflow edges.

An accepted efficiency measure of a posteriori error estimates is the effectivity index. In this paper we use the effectivity indices in the $\mathcal{L}^2$ norm defined as

$$\theta = \frac{||E||_{\mathcal{L}^2(\Omega)}}{||e||_{\mathcal{L}^2(\Omega)}}. \tag{2.11}$$

Ideally, the effectivity indices should approach unity under mesh refinement. We note that for transient problems the effectivity index is denoted by $\theta(t)$.

3. Adaptive Mesh Refinement. In this section we test our error estimation procedures presented in the previous section on adaptively refined meshes. We implement several h-refinement strategies and adaptive algorithms to compute DG solutions and error estimates on successively refined meshes.

Again, we recall that our modified DG method solves steady hyperbolic problems by first creating a mesh and arranging its elements into a list $M = \{\Delta_1, \Delta_2, \ldots, \Delta_j, \ \ldots\}$ such that

- Rule 1: All inflow elements, whose inflow edges are on the domain inflow $\partial\Omega^-$, are put first in the list M.
- Rule 2: The inflow edges of an element Δ_j in M are either on the domain inflow boundary $\partial\Omega^-$ or outflow edges of an element $\Delta_i, i < j$.

The modified DG method starts by computing the solution on the first element Δ_1 and proceeds downwind by computing the solution on elements in Δ_2, $\Delta_3, \ldots$ until the last element in M. Next we discuss several adaptive refinement algorithms that subdivide element having large "errors."

Algorithm 1 solves hyperbolic problems on a succession of locally refined meshes obtained by subdividing elements with errors larger than a specified threshold *delta*.

• **Algorithm 1** consists of the following steps

```
(i)  Set delta and Maxiter and k=0
(ii) construct an initial mesh Omega_0, order its elements
     in a list M of elements satisfying Rules 1 and 2
while k < Maxiter
    a- Solve the DG problem on Omega_k as described above.
    b- Compute errors ||E||_{Delta} for each element Delta in
       Omega_k
    c- For all elements Delta in M
          if ||E||_{Delta} < delta
              accept the DG solution on Delta
          else
             reject the DG solution on Delta
```

```
            subdivide Delta into 4 congruent triangles
          endif
    d- Complete triangulation by eliminating hanging nodes
       to create new mesh Omega_k+1 and order its elements
       in a list M satisfying Rules 1 and 2.
     e- k <-- k+1
endwhile
```

• **Algorithm 2a** solves the whole problem on an initial mesh and then goes back and solves the problem on each element and applies a local refinement algorithm to obtain a more accurate DG solution. It consists of the following steps:

```
(i)   Set omega and Create an initial mesh Omega
(ii)  Solve discrete DG problem on Omega as described above
(iii) Compute errors ||E||_{Delta} for each element Delta
      and compute error Emax = max_{Delta} ||E||_{Delta}
(iv)  For all elements Delta in M satisfying Rules 1 and 2
         if ||E||_{Delta} < omega*Emax
            Accept the DG solution on Delta
         else
            Reject the DG solution
            Subdivide Delta into 4 congruent triangles
            Complete triangulation to eliminate hanging nodes
            Update the list M satisfying Rules 1 and 2
         end
```

• **Algorithm 2b** follows the same steps as Algorithm 2a except for the refinement selection strategy. An element is selected for refinement if the local error exceeds a fraction of the average error Eavg following the steps:

```
(i)   Set omega and create a mesh Omega with N elements
(ii)  Solve discrete DG problem on Omega described above
(iii) Compute errors ||E||_{Delta} for each element Delta
      Compute average error Eavg=sum_{Delta}||E||_{Delta}/N
(iv)  For all elements in M satisfying Rules 1 and 2
        if ||E||_{Delta} < omega*Eavg
           Accept the DG solution on Delta
        else
           Reject the DG solution
           Subdivide Delta into 4 congruent triangles
           Complete triangulation to eliminate hanging nodes
           Update the list M satisfying Rules 1 and 2
        end
```

• **Algorithm 2c** is similar to Algorithm 2a. However, an element is selected for refinement if its residual exceeds a fraction of the maximum residual and follows the steps:

```
(i)   Set omega and Create an initial mesh Omega
(ii)  Solve discrete DG problem on Omega described above
(iii) Compute residual ||r||_{Delta}=||L(U)-f||_{Delta}
      on each element Delta and compute maximum residual
      rmax= max_{Delta} ||r||_{Delta}
(iv)  For all elements in M satisfying Rules 1 and 2
         if ||r||_{Delta} < omega*rmax
            Accept the DG solution on Delta
         else
            Reject the DG solution
            Subdivide Delta into 4 congruent triangles
            Complete triangulation to eliminate hanging nodes
            Update the list M satisfying Rules 1 and 2
         end
```

• **Algorithm 2d** follows the same steps as Algorithm 2a with one exception: an element is selected for refinement if the element residual exceeds a fraction of the average residual and follows the steps:

```
(i)   Set omega and Create a mesh Omega with N elements
(ii)  Solve discrete DG problem on Omega described above
(iii) Compute residual ||r||_{Delta}=||L(U)-f||_{Delta}
       on each element Delta and compute average residual
       ravg= sum_{Delta} ||r||_{Delta}/N
(iv)  For all elements in M satisfying Rules 1 and 2
          if ||r||_{Delta} < omega*ravg
            Accept the DG solution on Delta
          else
            Reject the DG solution
            Subdivide Delta into 4 congruent triangles
            Complete triangulation to eliminate hanging nodes
            Update the list M satisfying Rules 1 and 2
          end
```

• **Algorithm 3** prevents the errors on elements near the discontinuity from polluting elements having smooth solutions. Each element having a large error is immediately refined after computing its DG solution, while the refinement in Algorithm 1 is performed after computing the solution on all elements. Thus, this algorithm will reduce the errors as they appear and is expected to reduce the pollution errors observed with Algorithm 1. This algorithm consists of the following steps:

```
(i)   Set delta
(ii)  Construct an initial mesh
(iii) Order elements in a list M satisfying Rules 1 and 2
```

```
(iv)  For all elements Delta in M
          a- Compute  DG solution on Delta
          b- Compute error ||E||_{Delta}

          c- If ||E||_{Delta} < delta
                Accept the DG solution on Delta
             else
                Reject DG solution on Delta
                Subdivide Delta into 4 congruent triangles
                Eliminate hanging nodes
                Update the list M satisfying Rules 1 and 2
             endif
```

• **Algorithm 4** is similar to Algorithm 3 except for the refinement selection criteria. Here an element Delta is selected for refinement if the maximum residual over all elements before Delta in the list M exceeds a user specified tolerance delta and follows the following steps:

```
(i)   Set delta
(ii)  Construct an initial mesh
(iii) Order elements in a list M satisfying Rules 1 and 2
(iv)  For all elements Delta in M
         a- Compute  DG solution on Delta
         b- Compute maximum residual rmax over all elements
            in M before element Delta
         c- If rmax < delta
                Accept the DG solution on Delta
            else
                Reject DG solution on Delta
                Subdivide Delta into 4 congruent triangles
                Eliminate hanging nodes
                Update the list M satisfying Rules 1 and 2
            endif
```

• **Algorithm 5** is similar to Algorithm 4 except for the refinement selection criteria. Here an element Delta is selected for refinement if the average residual over all elements before Delta exceeds a user specified tolerance delta and follows the following steps:

```
(i)   Set delta
(ii)  Construct an initial mesh
(iii) Order elements in a list M satisfying Rules 1 and 2
(iv)  For all elements Delta in M
          a- Compute  DG solution on Delta
          b- Compute average residual ravg over all elements
             in M and before element Delta
```

```
          c- If ravg < delta
                Accept the DG solution on Delta
             else
                Reject DG solution on Delta
                Subdivide Delta into 4 congruent triangles
                Eliminate hanging nodes
                Update the list M satisfying Rules 1 and 2
             endif
```

• **Algorithm 6** is used with a time-marching scheme that converges to a steady state solution and consists of the following steps:

```
(i) Set delta, NStep, k=0 and construct a mesh Omega_k
    Set dt and t_k = k*dt
while k < Nstep
   (ii)  Integrate from t_k to t_{k+1} on Omega_k
   (iii) Compute errors ||E||_{Delta} on each element Delta
   (iv)  For all elements Delta
            If ||E||_{Delta} < delta
               Accept the DG solution on Delta
            Else
               Subdivide Delta into 4 congruent triangles
            endif

    (v)   Complete triangulation to eliminate hanging nodes
           Create a new mesh Omega_{k+1}

    (vi)  Increment k <--k+1 and return to step (ii)
endwhile
```

4. Computational Examples. In this section, we present numerical results for several hyperbolic problems showing the convergence properties of DG solutions and a posteriori error estimates in the presence of discontinuities on unstructured meshes. The error estimates are tested on linear and nonlinear problems with discontinuous solutions to show their efficiency and accuracy under adaptive mesh refinement. For all examples we use exact boundary conditions at the *inflow* boundary. In all our examples we use the space $\mathcal{U}_p$ on unstructured triangular meshes. Furthermore, for the transient problem we apply the MATLAB ode45 to perform the temporal integration and assume the temporal discretization errors to be negligible.

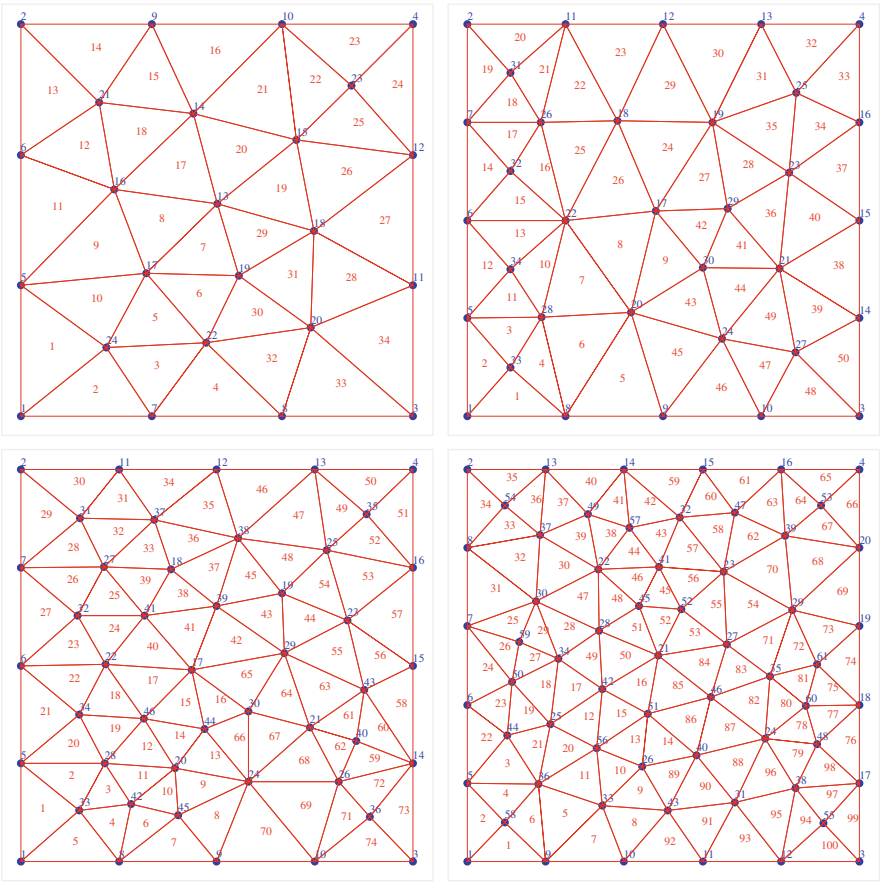

FIG. 1. Unstructured meshes having $N = 34, 50, 74, 100$ triangles

EXAMPLE 1. Let us consider the initial-boundary value problem for the inviscid Burgers' equation

$$u_y + \left(\frac{u^2}{2}\right)_x = u_y + uu_x = 0, \ (x, y) \in [0, 1] \times [0, 0.999], \tag{4.1a}$$

subject to the initial conditions

$$u(x, 0) = g_0(x) = 1 + \frac{1}{2}\sin(2\pi x). \tag{4.1b}$$

and select $u(0, y) = g_1(y)$ such that the true solution is periodic and forms a shock discontinuity at the point $(\frac{1}{\pi}, \frac{1}{\pi})$ which propagates along $y = x$. We perform several tests on this example to study both the accuracy and efficiency of our a posteriori error estimates for the modified DG method on adaptively refined unstructured meshes.

First, we apply the modified DG method (2.10) and (2.6b) to solve problem (4.1) on the unstructured meshes shown in Fig. 1 having $N = 34, 50, 74, 100$ triangular elements with $\mathcal{U}_p$, $p = 1, 2, 3, 4$. We observe that the proposed error estimates do not converge to the true error under regular mesh refinement. The global error is underestimated is due to the fact that errors on elements near the discontinuity, which have an important contribution to the global error, are underestimated (Table 3).

TABLE 3
$\mathcal{L}^2$ errors and global effectivity indices on for problem (4.1) on unstructured meshes having $N = 34, 50, 74, 100$ elements using $p = 1, 2, 3, 4$

N	$p=1$		$p=2$	
	$\|\|e\|\|_{\mathcal{L}^2}$	θ	$\|\|e\|\|_{\mathcal{L}^2}$	θ
34	8.0543e−1	0.87505	1.2714e−1	0.67312
50	5.1083e−1	0.88783	4.1497e−2	0.68294
74	2.6779e−1	0.89174	2.0175e−2	0.68595
100	1.2956e−1	0.89585	1.1439e−2	0.68912
	$p=3$		$p=4$	
34	1.5127e−1	0.36461	1.1168e−1	0.25002
50	4.6004e−2	0.36993	2.2925e−2	0.25366
74	1.6925e−2	0.37156	6.9803e−3	0.25478
100	8.0424e−3	0.37327	2.8216e−3	0.25596

We apply the modified DG method (2.9) and (2.6b) to solve the inviscid Burgers' equation (4.1) on $[0, 1] \times [0, 0.999]$ on unstructured meshes having $N = 432$ elements shown in Fig. 2 with $\mathcal{U}_p$ and with no special treatment at the shock such as stabilization or limiting. Then, we apply Algorithm 1 to locally refine the mesh by performing 6 iterations to generate a sequence of refined meshes. Algorithm 1 subdivides triangles for which the local error in the $\mathcal{L}^2$ norm is larger than a prescribed tolerance δ. In Figs. 3 and 4 we show the sequence of meshes obtained by applying Algorithm 1 with $\delta = 0.001$ for $p = 1, 2$ where elements near the shock discontinuity are refined. However, we notice that a large portion of elements away from the discontinuity are refined which suggest that this algorithm is not efficient.

In Table 4 we present the number of elements in each mesh obtained at every refinement iteration of Algorithm 1, the true $\mathcal{L}^2$ errors and the effectivity indices for $p = 1, 2$ for each refinement iteration. These results suggest that the global $\mathcal{L}^2$ error estimates converge to the true error under a "crude" adaptive mesh refinement of Algorithm 1 in the presence of shock discontinuities.

We now consider the same problem (4.1) and use Algorithm 2a with the modified DG method (2.9) and (2.6b). In Figs. 5 and 6 we plot the meshes obtained from Algorithm 2a with $\omega = 0.85$, $\omega = 0.75$, $\omega = 0.5$, $\omega = 0.25$,

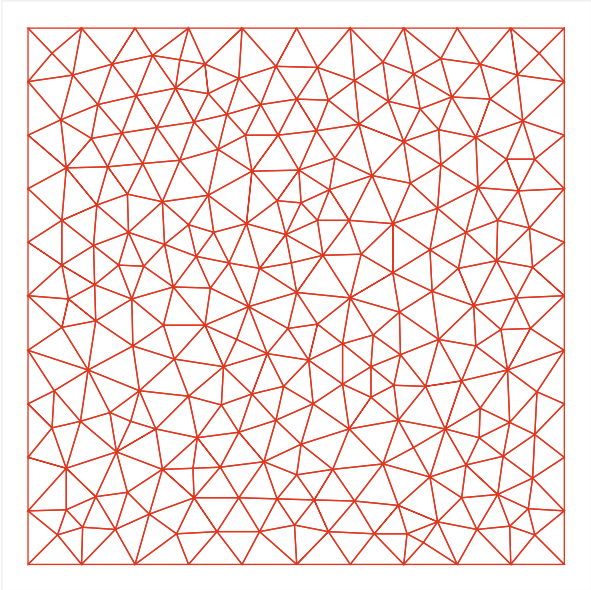

FIG. 2. An unstructured mesh having $N = 432$ triangles

TABLE 4
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 1 with 6 refinement iterations

	$p=1$			$p=2$		
Iteration	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
1	432	2.5104e−2	0.5462	432	3.0244e−3	0.4189
2	841	1.4230e−2	0.6253	711	1.3184e−3	0.5491
3	1,778	9.2502e−3	0.7555	1,323	6.9683e−4	0.6393
4	3,012	6.4997e−3	0.8508	2,544	4.0806e−4	0.7595
5	5,514	4.2522e−4	0.9448	3,970	2.5819e−5	0.8698
6	8,148	2.0173e−4	0.9747	5,783	1.5164e−5	0.9499

and $p = 1, 2$. In Table 5 we present the number of elements N, the true $\mathcal{L}^2$ errors, and the global $\mathcal{L}^2$ effectivity indices with $p = 1, 2$ which suggest that the proposed error estimates converge to the true error under adaptive refinement of unstructured triangular meshes.

Now, we use Algorithm 2b to solve (4.1) with the modified DG method (2.9) and (2.6b). In Fig. 7 we plot the meshes obtained by applying the adaptive algorithm with $\omega =$ 2, 1.75, 1.5, 1.25, 1, 0.75, 0.5, 0.25 and $p = 1, 2$. In Table 6 we present the number of elements N, the true $\mathcal{L}^2$ errors, and the global $\mathcal{L}^2$ effectivity indices with $p = 1, 2$ for $\omega = 2$, 1.75,

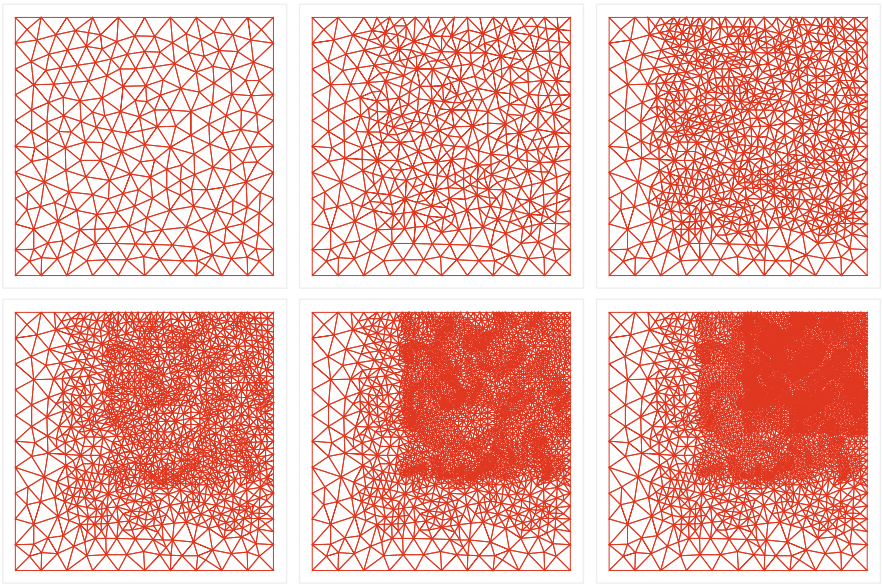

FIG. 3. Adaptive meshes obtained by Algorithm 1 for problem (4.1) and $p = 1$

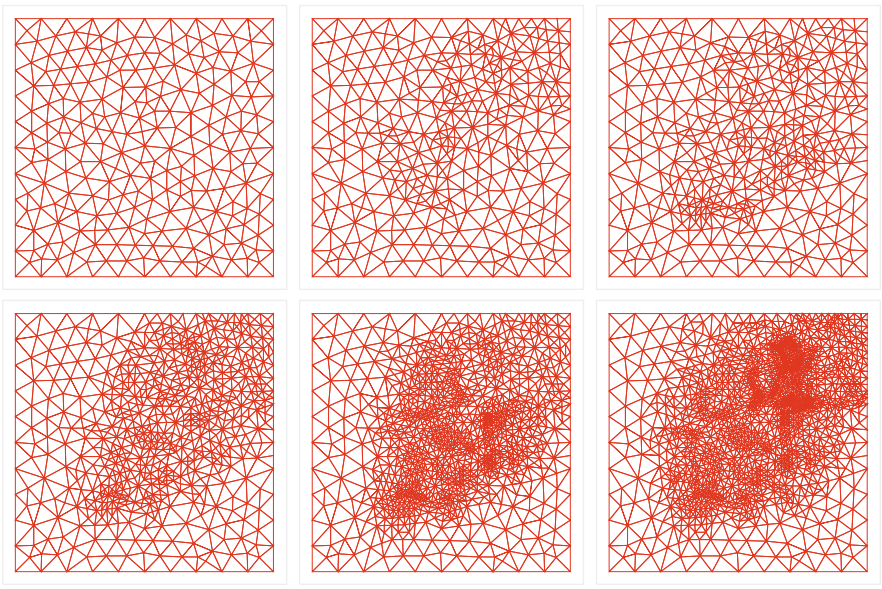

FIG. 4. Adaptive meshes obtained by Algorithm 1 for problem (4.1) and $p = 2$

TABLE 5
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 2a with $\omega = 0.85, 0.75, 0.5, 0.25, 0.15, 0.1$ and $p = 1, 2$

p	$p=1$			$p=2$		
ω	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
0.85	627	2.3903e−2	0.6528	842	2.9155e−3	0.5561
0.75	917	1.3549e−2	0.7744	1,262	6.7175e−4	0.6898
0.5	1,433	8.8076e−3	0.8704	2,156	3.9337e−4	0.8674
0.25	2,094	4.0488e−4	0.9761	5,011	4.5249e−5	0.9686
0.15	3,450	1.9546e−4	0.9798	7,213	2.2231e−5	0.9752
0.1	4,343	1.5432e−4	0.9812	8,982	1.1856e−5	0.9765

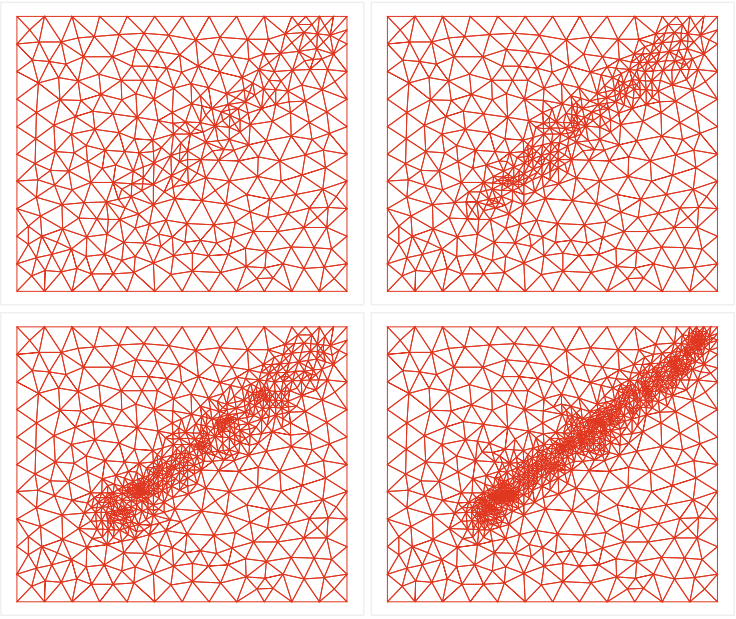

FIG. 5. Meshes generated by Algorithm 2a for the problem (4.1) with $\omega = 0.85$, $\omega = 0.75$, $\omega = 0.5$, $\omega = 0.25$ (*upper left to lower right*) and $p = 1$

1.5, 1.25, 1, 0.75, 0.5, 0.25 which show that our error estimates converge to the true error under adaptive refinement of unstructured triangular meshes.

This adaptive mesh-refinement strategy also yields an efficient adaptive algorithm.

Next, we solve problem (4.2) using Algorithm 2c with the modified DG method (2.9) and (2.6b). In Figs. 8 and 9 we plot the meshes obtained by Algorithm 2c with $\omega = 0.85$, $\omega = 0.75$, $\omega = 0.5$, $\omega = 0.25$ and $p = 1, 2$ to the problem (4.1). In Table 7 we present the number of elements N, the global $\mathcal{L}^2$ norm of the error, and the global $\mathcal{L}^2$ effectivity indices with $p = 1, 2$

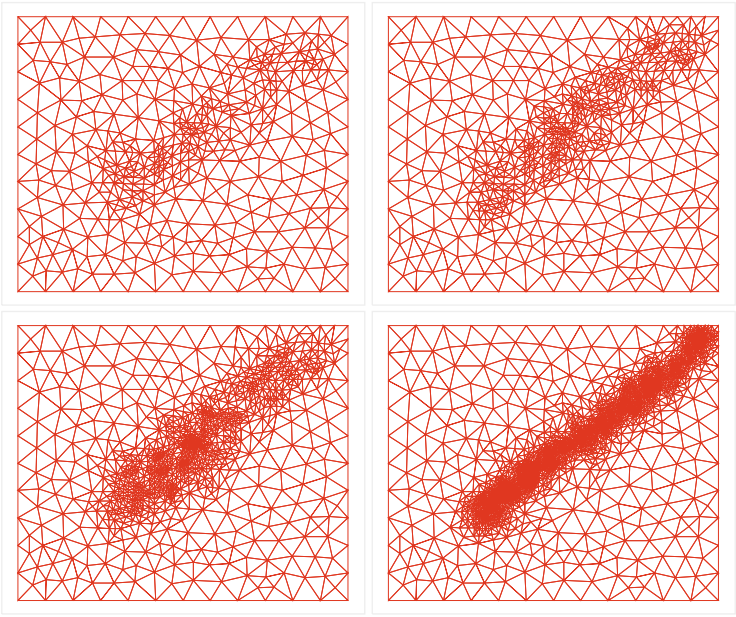

FIG. 6. Meshes generated by Algorithm 2a for problem (4.1) with $\omega = 0.85$, $\omega = 0.75$, $\omega = 0.5$, $\omega = 0.25$ (*upper left to lower right*) and $p = 2$

which show that the error estimates converge to the true error under adaptive mesh refinement. They further show that the proposed error estimates are accurate on adaptively refined unstructured triangular meshes.

TABLE 6
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 2d with $\omega = 2, 1.75, 1.5, 1.25, 1, 0.75, 0.5, 0.25$ and $p = 1, 2$

	$p=1$			$p=2$		
ω	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
2.00	705	1.8498e−2	0.5351	803	1.7897e−3	0.5448
1.75	865	1.0485e−2	0.5819	967	1.0145e−3	0.5938
1.50	1,426	6.8160e−3	0.6959	1,756	6.5947e−4	0.7167
1.25	1,941	5.7147e−3	0.7813	2,305	4.6337e−4	0.7978
1.00	3,140	4.7892e−3	0.8918	3,922	3.3465e−4	0.9186
0.75	5,462	3.1332e−4	0.9631	6,431	3.0315e−5	0.9435
0.50	11,142	2.0637e−4	0.9722	8,670	2.0813e−5	0.9627
0.25	19,280	1.2973e−4	0.9828	14,548	1.0637e−5	0.9749

Next, we apply Algorithm 2d with the modified DG method (2.9) and (2.6b) to solve (4.1). In Fig. 10 we plot the meshes obtained by applying Algorithm 2d with $\omega = 2, 1.75, 1.5, 1.25, 1, 0.75, 0.5, 0.25$ for $p = 1, 2$. In

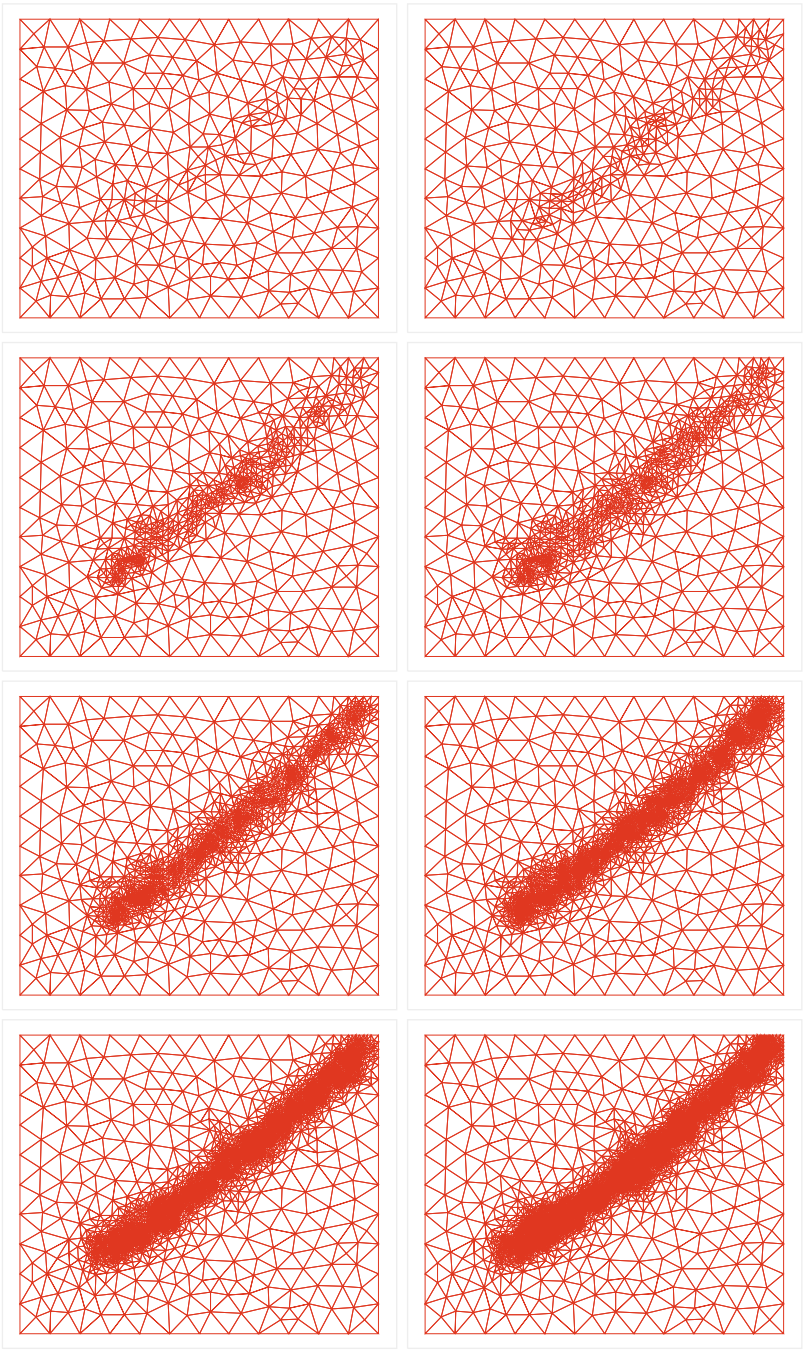

FIG. 7. Meshes obtained by Algorithm 2b for problem (4.1) with $\omega = 2,\ 1.75,\ 1.5,\ 1.25,\ 1,\ 0.75,\ 0.5,\ 0.25$ (*upper left to lower right*) and $p = 1$

TABLE 7
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 2c with $\omega = 0.85, 0.75, 0.5, 0.25, 0.15, 0.1$ and $p = 1, 2$

Degree	$p=1$			$p=2$		
ω	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
0.85	677	2.3062e−2	0.5942	817	3.0276e−3	0.5860
0.75	937	1.3072e−2	0.7368	1,362	1.3199e−3	0.7395
0.50	1,505	8.4978e−3	0.8477	2,130	4.0850e−4	0.8453
0.25	2,543	3.9064e−4	0.9664	3,385	3.5191e−5	0.9599
0.15	3,143	2.1354e−4	0.9713	4,328	1.6104e−5	0.9674
0.10	3,951	1.4983e−4	0.9784	5,235	1.1467e−5	0.9733

Table 8 we present the number of elements N, the $\mathcal{L}^2$ errors, and the global $\mathcal{L}^2$ effectivity indices for $p = 1, 2$ and $\omega = 2$, 1.75, 1.5, 1.25, 1, 0.75, 0.5, 0.25 which suggest that the proposed error estimates converge to the true error under adaptive mesh refinement on unstructured triangular meshes.

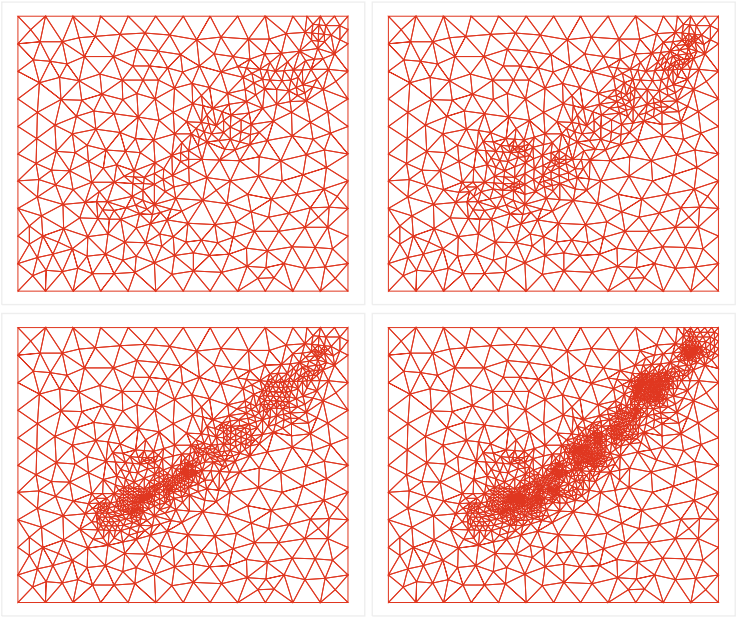

FIG. 8. Meshes generated by Algorithm 2c for problem (4.1) with $\omega = 0.85$, $\omega = 0.75$, $\omega = 0.5$, $\omega = 0.25$ (*upper left to lower right*) and $p = 1$

Now we apply the modified DG method (2.9) and (2.6b) with Algorithm 3 to solve (4.1). Again, the final mesh is constructed through a sequence of successively refined meshes by refining triangles whose local

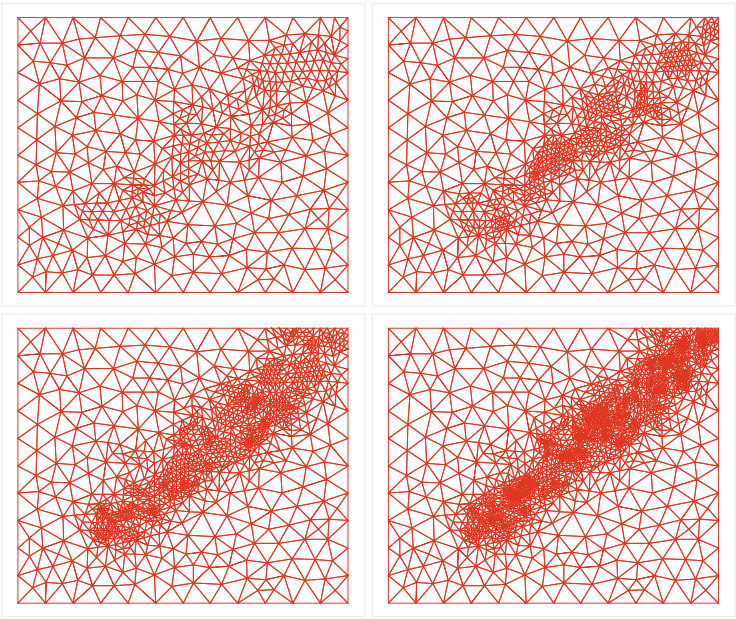

FIG. 9. Meshes generated by Algorithm 2c for problem (4.1) with $\omega = 0.85$, $\omega = 0.75$, $\omega = 0.5$, $\omega = 0.25$ (*upper left to lower right*) and $p = 2$

TABLE 8
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 2b with $\omega = 2,\ 1.75,\ 1.5,\ 1.25,\ 1,\ 0.75,\ 0.5,\ 0.25$ and $p = 1, 2$

Degree	$p = 1$			$p = 2$		
ω	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
2.00	580	2.0179e−2	0.5263	678	2.0660e−3	0.4536
1.75	749	2.7182e−2	0.5631	957	1.1710e−3	0.5138
1.50	1,399	1.1438e−2	0.6860	1,832	7.6126e−4	0.6313
1.25	1,836	7.4356e−3	0.7539	2,450	5.3490e−4	0.6868
1.00	3,094	5.2246e−3	0.8962	3,985	3.6675e−4	0.8171
0.75	5,430	3.4181e−4	0.9648	5,043	8.3722e−5	0.8791
0.50	10,272	2.2616e−4	0.9781	8,086	3.1995e−5	0.9409
0.25	17,841	1.4217e−4	0.9787	15,479	1.1250e−5	0.9629

error in the $\mathcal{L}^2$ norm is larger than some prescribed δ. For instance, in Figs. 11 and 12 we show the final meshes obtained by applying Algorithm 3 with $\delta = 0.01,\ 0.001$, for $p = 1, 2$. The $\mathcal{L}^2$ errors and effectivity indices for $\delta = 0.05,\ 0.01,\ 0.005,\ 0.001,\ 0.0005$ and $p = 1, 2$ shown in Table 9 suggest that the proposed error estimates converge to the true error under adaptive mesh refinement. Furthermore, we observe that the adaptive

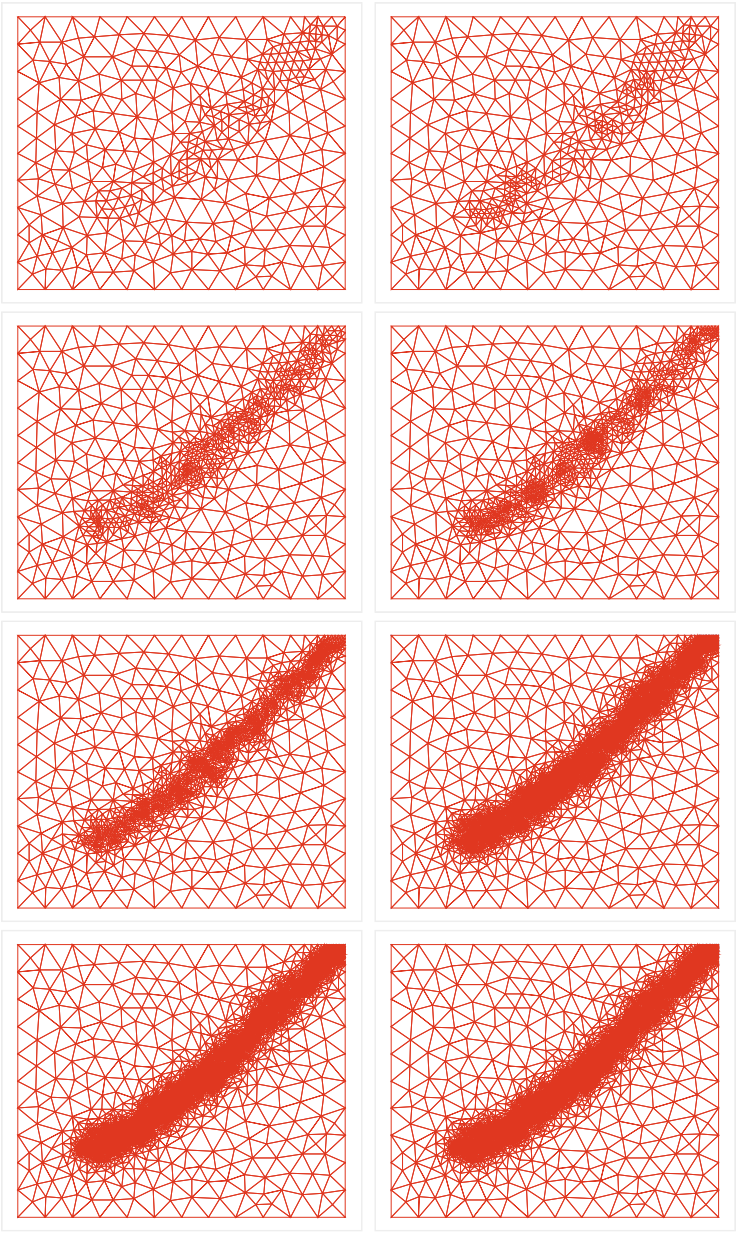

FIG. 10. Meshes obtained by Algorithm 2d for problem (4.1) with $\omega = 2,\ 1.75,\ 1.5,\ 1.25,\ 1,\ 0.75,\ 0.5,\ 0.25$ (*upper left to lower right*) and $p = 1$

TABLE 9
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 3 with $p = 1, 2$

p	$p = 1$			$p = 2$		
δ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
0.05	759	9.8076e−3	0.9541	1,167	9.7476e−4	0.9569
0.01	904	8.3650e−3	0.9628	2,092	6.2409e−4	0.9644
0.005	1,532	7.2128e−4	0.9718	3,945	7.0488e−5	0.9718
0.001	2,267	3.8453e−4	0.9737	5,444	2.4167e−5	0.9754
0.0005	5,890	1.4675e−4	0.9813	8,412	1.1247e−5	0.9841

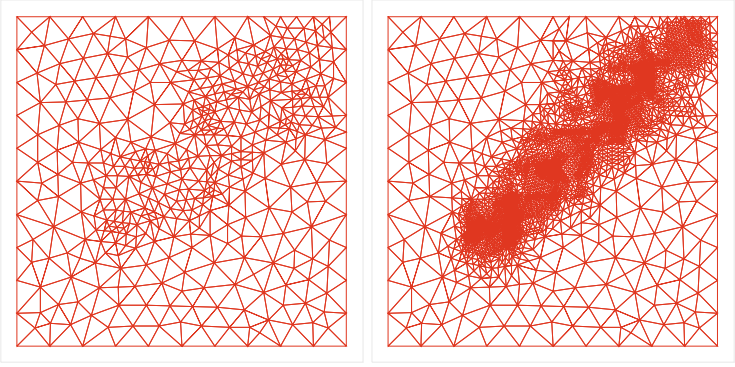

FIG. 11. Meshes obtained by Algorithm 3 for problem (4.1) with tolerance $\delta = 0.01$ and $p = 1$ (*left*), $p = 2$ (*right*)

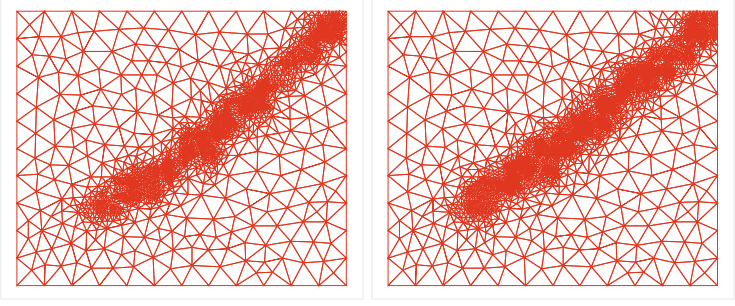

FIG. 12. Meshes obtained by Algorithm 3 for problem (4.1) with tolerance $\delta = 0.001$ and $p = 1$ (*left*), $p = 2$ (*right*)

TABLE 10
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 4 with $p = 1, 2$

	$p = 1$			$p = 2$		
δ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
0.0500	732	9.3422e−3	0.6447	1,087	8.8284e−4	0.5558
0.0100	1,023	6.3276e−3	0.7864	1,983	5.2330e−4	0.6574
0.0050	1,465	6.6306e−4	0.8771	3,829	5.5168e−5	0.7681
0.0010	2,105	3.0643e−4	0.9372	5,356	2.1162e−5	0.8882
0.0005	5,459	1.2974e−4	0.9800	8,221	1.0933e−5	0.9588

mesh-refinement strategy of Algorithm 3 yields a more efficient adaptive algorithm for the modified DG method applied to hyperbolic problems on general unstructured triangular meshes.

Now, we use Algorithm 4 and the modified DG method (2.9) and (2.6b) to solve (4.1). The final mesh is constructed through a sequence of successively refined meshes where triangles for which the $\mathcal{L}^2$ norm of the residual on each element $||r_i||$ is larger than $\delta = 0.001$ are refined. In Fig. 13 we plot the final meshes obtained by applying Algorithm 4 for $p = 1, 2$. In Table 10 we present the number of elements, $\mathcal{L}^2$ errors, and the global $\mathcal{L}^2$ effectivity indices for the tolerances $\delta = 0.05, 0.01, 0.005, 0.001, 0.0005$, $p = 1, 2$ which suggest that the error estimates converge to the true error during under adaptive mesh refinement.

As a final test, we solve (4.1) using Algorithm 5 with modified DG method (2.9) and (2.6b). The final mesh is constructed through a sequence of successively refined meshes where triangles for which the average residual exceeds the specified threshold δ are refined. In Fig. 14 we plot the final meshes from Algorithm 5 with $\delta = 0.001$ and $p = 1, 2$. In Table 11 we present the number of elements, $\mathcal{L}^2$ errors, and the global $\mathcal{L}^2$ effectivity indices for tolerances $\delta = 0.05, 0.01, 0.005, 0.001, 0.0005$, and degrees $p = 1, 2$. These results suggest that our a posteriori error estimates converge to the true error under adaptive mesh refinement.

TABLE 11
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.1) on unstructured meshes having N elements using Algorithm 5 with $p = 1, 2$

	$p = 1$			$p = 2$		
δ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
0.0500	712	9.9638e−3	0.6257	1,054	8.9863e−4	0.5578
0.0100	914	3.2468e−3	0.7474	1,623	4.8685e−4	0.7594
0.0050	1,296	5.4606e−4	0.8881	3,298	5.2624e−5	0.8601
0.0010	1,949	2.9263e−4	0.9482	4,846	2.0400e−5	0.9202
0.0005	4,787	1.3041e−4	0.9788	6,759	1.0845e−5	0.9608

We observe that Algorithms 2–5 generate adaptive meshes with fine elements only in the vicinity of the shock. The results from Algorithms 2a–d of Tables 12 and 13 suggest that Algorithms 2a–d have comparable efficiency. In Fig. 15 we plot the true $\mathcal{L}^2$ errors versus the number of elements needed to satisfy the specified tolerance for all Algorithms 1–5 applied to the Burger's equation with a shock (4.1). Our computations suggest that Algorithm 1 is the least efficient adaptive method, while Algorithms 3, 4, and 5 are the most efficient adaptive procedures. We further note that using the local residuals to refine elements in Algorithm 5 yields a slightly more efficient algorithm. Thus, we recommend the space-time DG method that uses local residuals to select elements for refinement and the a posteriori error estimates to assess the solution accuracy and terminate the adaptive process.

TABLE 12
$\mathcal{L}^2$ errors and number of elements for Algorithms 2a and 2c applied to problem (4.1)

$p=1$					$p=2$			
Algorithm 2a			Algorithm 2c		Algorithm 2a		Algorithm 2c	
ω	N	$\|\|e\|\|_{\mathcal{L}^2}$	N	$\|\|e\|\|_{\mathcal{L}^2}$	N	$\|\|e\|\|_{\mathcal{L}^2}$	N	$\|\|e\|\|_{\mathcal{L}^2}$
0.85	627	2.3903e−2	677	2.3062e−2	842	2.9155e−3	817	3.0276e−3
0.75	917	1.3549e−2	937	1.3072e−2	1,262	6.7175e−4	1,362	1.3199e−3
0.50	1,433	8.8076e−3	1,505	8.4978e−3	2,156	3.9337e−4	2,130	4.0850e−4
0.25	2,094	4.0488e−4	2,543	3.9064e−4	5,011	4.5249e−5	3,385	3.5191e−5
0.15	3,450	1.9546e−4	3,143	2.1354e−4	7,213	2.2231e−5	4,328	1.6104e−5
0.10	4,343	1.5432e−4	3,951	1.4983e−4	8,982	1.1856e−5	5,235	1.1467e−5

EXAMPLE 2 (Transient Burger's equation). Let us consider the initial-boundary value problem for the inviscid Burgers' equation

$$\epsilon u_t + u_y + \left(\frac{u^2}{2}\right)_x = 0, \ (x,y,t) \in [0,1] \times [0,0.999] \times [0,T], \tag{4.2a}$$

subject to the boundary conditions

$$u(x,0,t) = u_2(x) = 1 + \frac{1}{2}\sin(2\pi x), \qquad u(0,y,t) = u(1,y,t) = u_1(y). \tag{4.2b}$$

The initial conditions $u(x,y,0) = u_0(x,y)$ are selected such that $u_0(x,0) = u_2(x)$ and $u_0(0,y) = u_1(y)$ as follows

$$\begin{aligned} u_0(x,y) &= \overline{N}_1(x)u_1(y) + \overline{N}_2(x)\tilde{u}_1(y) + \overline{N}_1(y)u_2(x) + \overline{N}_2(y)\tilde{u}_2(x) \\ &\quad - \overline{N}_1(x)\overline{N}_1(y)u_1(0) - \overline{N}_2(x)\overline{N}_2(y)\tilde{u}_1(1) - \overline{N}_1(x)\overline{N}_2(y)u_1(1) \\ &\quad - \overline{N}_2(x)\overline{N}_1(y)u_2(1). \end{aligned} \tag{4.2c}$$

where $\overline{N}_1(x) = 1 - x$, $\overline{N}_2(x) = x$, $\tilde{u}_1(y) = (1-y)u_2(1)$ and $\tilde{u}_2(x) = (1-x)u_1(1)$.

We apply the modified DG method (2.10) to solve this problem on $[0,1] \times [0,0.999] \times [0,T]$ with a smooth solution on initial unstructured

TABLE 13
$\mathcal{L}^2$ errors and number of elements for Algorithms 2b and 2d applied to problem (4.1)

Algorithm 2b			Algorithm 2d	
ω	N	$\|\|e\|\|_{\mathcal{L}^2}$	N	$\|\|e\|\|_{\mathcal{L}^2}$
$p=1$				
2.00	580	2.0179e−2	705	1.8498e−2
1.75	749	2.7182e−2	865	1.0485e−2
1.5	1,399	1.1438e−2	1,426	6.8160e−3
1.25	1,836	7.4356e−3	1,941	5.7147e−3
1.00	3,094	5.2246e−3	3,140	4.7892e−3
0.75	5,430	3.4181e−4	5,462	3.1332e−4
0.50	10,272	2.2616e−4	11,142	2.0637e−4
0.25	17,841	1.4217e−4	19,280	1.2973e−4
$p=2$				
2.00	678	2.0660e−3	803	1.7897e−3
1.75	957	1.1710e−3	967	1.0145e−3
1.5	1,832	7.6126e−4	1,756	6.5947e−4
1.25	2,450	5.3490e−4	2,305	4.6337e−4
1.00	3,985	3.6675e−4	3,922	3.3465e−4
0.75	5,043	8.3722e−5	6,431	3.0315e−5
0.50	8,086	3.1995e−5	8,670	2.0813e−5
0.25	15,479	1.1250e−5	14,548	1.0637e−5

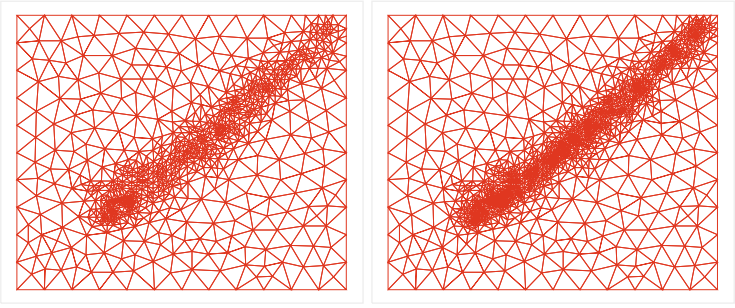

FIG. 13. Meshes generated by Algorithm 4 for problem (4.1) with tolerance $\delta = 0.001$ and $p = 1$ (*left*), $p = 2$ (*right*)

meshes having $N = 500$ triangular elements of type I, II, and III with $\mathcal{U}_p$, $p = 1, 2$, and $\epsilon = 10^{-2}$.

We use the adaptive mesh-refinement procedure given in Algorithm 6. The final mesh at $t = T = 1$ is constructed through a sequence of successively refined meshes where triangles for which the $\mathcal{L}^2$ error exceeds δ are refined. In Figs. 16 and 17 we plot the sequence of meshes obtained by applying the adaptive method with $\delta = 0.001$ and $t = 0, 0.25, 0.5, 0.75, 1$ to the problem (4.2).

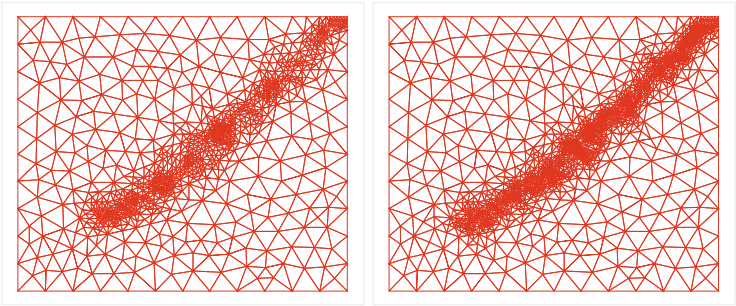

FIG. 14. Meshes generated by Algorithm 5 for problem (4.1) with $\delta = 0.001$ and $p = 1$ (*left*), $p = 2$ (*right*)

In Table 14 we present the number of elements, the true $\mathcal{L}^2$ errors, and the global $\mathcal{L}^2$ effectivity indices at $t = 0, 0.25, 0.5, 0.75, 1$ and $p = 1, 2$ which show that the error estimates converge to the true error during the simulation. These computational results indicate that our estimators are accurate on adaptively refined unstructured triangular meshes and further suggest that they converge to the true error under adaptive mesh refinement of Algorithm 6.

EXAMPLE 3. We consider the following linear problem

$$2u_x + u_y = 0, \qquad (x, y) \in [0, 1]^2, \tag{4.3a}$$

subject to the boundary conditions

$$u(x, 0) = e^{-x}, \qquad 0 \le x \le 1, \tag{4.3b}$$

$$u(0, y) = e^{2y} + .25, \qquad 0 < y \le 1, \tag{4.3c}$$

with the true solution having a contact discontinuity along $y = x/2$

$$u(x, y) = \begin{cases} e^{2y-x} & \text{if } x \ge y \\ e^{2y-x} + .25 & \text{if } x < y \end{cases}. \tag{4.3d}$$

We solve (4.3) on the unstructured mesh having 100 elements shown in Fig. 1 with the spaces $\mathcal{U}_p$, $p = 1, 2, 3, 4$ and apply the modified DG method (2.6) and (2.7) with the adaptive refinement strategy described in Algorithm 3 for $\delta = 0.01$, 0.005, and 0.001. We present the mesh in Fig. 18 for $\delta = 0.001$ and show in Table 15 the number of elements N, the global $\mathcal{L}^2$ norm of the error, and the global $\mathcal{L}^2$ effectivity indices with $\delta = 0.01$, 0.005, and 0.001 and $p = 1, 2, 3, 4$. These results suggest that the error estimates converge to the true error under local adaptive mesh refinement algorithm that refines elements near the discontinuity and whose errors are underestimated.

TABLE 14
$\mathcal{L}^2$ errors and global effectivity indices for problem (4.2) on unstructured meshes having N elements with $p = 1, 2$ at $t = 0, 0.25, 0.5, 0.75, 1$

	$p=1$			$p=2$		
t	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ	N	$\|\|e\|\|_{\mathcal{L}^2}$	θ
0.00	432	1.4402e−2	0.5860	432	2.7326e−3	0.4879
0.25	834	8.1633e−3	0.6743	984	1.1912e−3	0.5679
0.50	1,895	5.3067e−3	0.7654	2,348	6.2960e−4	0.6974
0.75	3,469	3.7288e−3	0.8904	4,467	3.6869e−4	0.8279
1.00	5,737	2.4394e−4	0.9747	7,404	1.4292e−5	0.9372

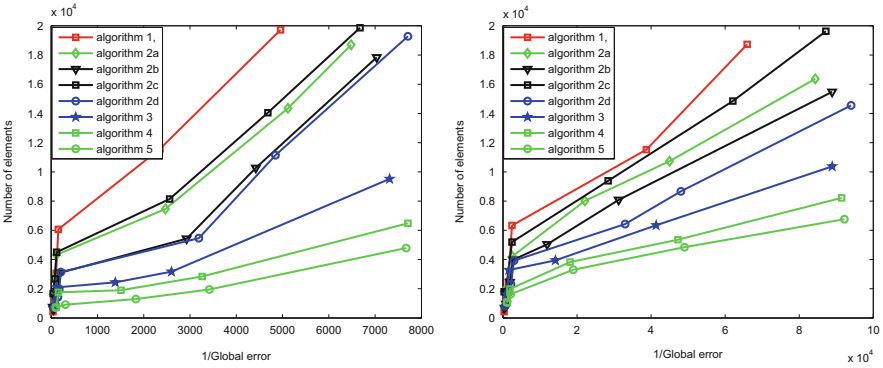

FIG. 15. True $\mathcal{L}^2$ errors versus the number of elements for Algorithms 1–6 with $p = 1$ (*left*) and $p = 2$ (*right*) for Problem (4.1)

TABLE 15
Global effectivity indices and final number of elements N for problem (4.3) with $p = 1, 2, 3, 4$ and error tolerances of $\delta = 0.01, 0.005, 0.001$

	$p=1$		$p=2$		$p=3$		$p=4$	
δ	θ	N	θ	N	θ	N	θ	N
0.01	0.9280	100	0.9255	100	0.9242	100	0.9348	100
0.005	0.9628	139	0.9456	121	0.9399	116	0.9482	116
0.001	0.9941	738	0.9928	523	0.9864	492	0.9858	426

5. Conclusions. We tested the residual-based a posteriori DG error estimates of Adjerid and Baccouch [3] for hyperbolic problems on unstructured triangular meshes. Several computational examples suggest that the Taylor's series residual-based error estimates proposed in this manuscript converge to the true error under local mesh refinement and in the presence of discontinuities. Similarly, we expect that the error estimates proposed by Adjerid and Mechai [6] on tetrahedral meshes will also converge to the true error under adaptive mesh refinement. Our future work will focus on

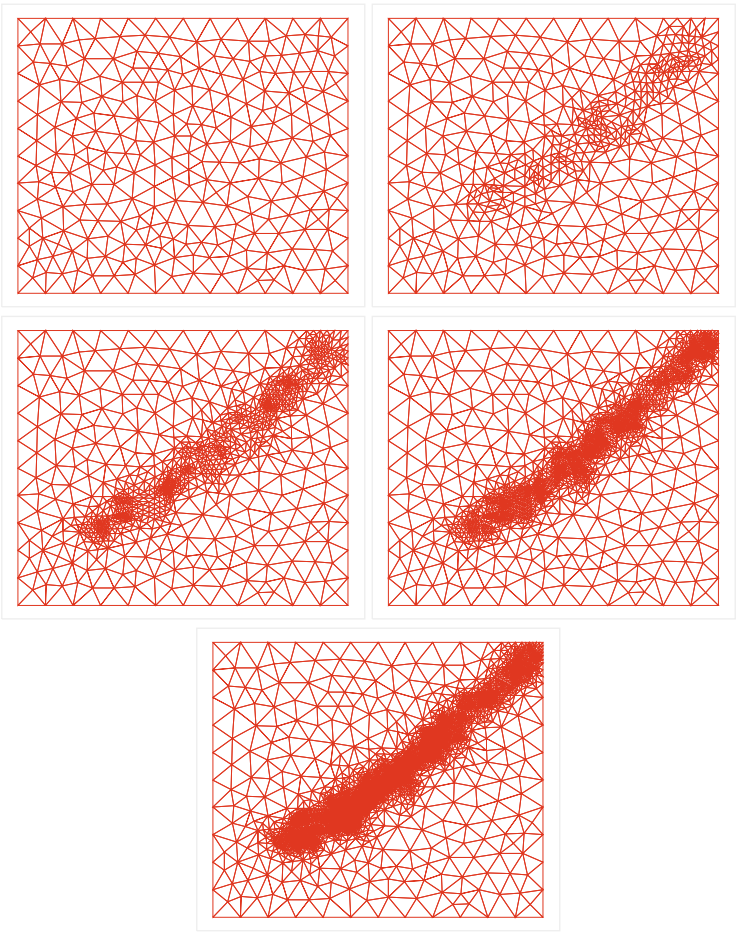

FIG. 16. Adaptive meshes for problem (4.2) with $p = 1$ at $t = 0.00, 0.25, 0.50, 0.75, 1.00$ (*upper left to lower right*)

applying adaptive refinement strategies to multi-dimensional hyperbolic systems of conservation laws and reach similar conclusions for general unstructured tetrahedral meshes.

Acknowledgments. The authors are grateful to Bryan Johnson (undergraduate student at the University of Nebraska at Omaha) for applying the adaptive algorithms to the contact problem to generate the results for Example 3.

The work of the Slimane Adjerid author was supported in part by NSF grant DMS-0809262. The work of the Mahboub Baccouch author was supported by the NASA Nebraska Space Grant Program and UCRCA at the University of Nebraska at Omaha.

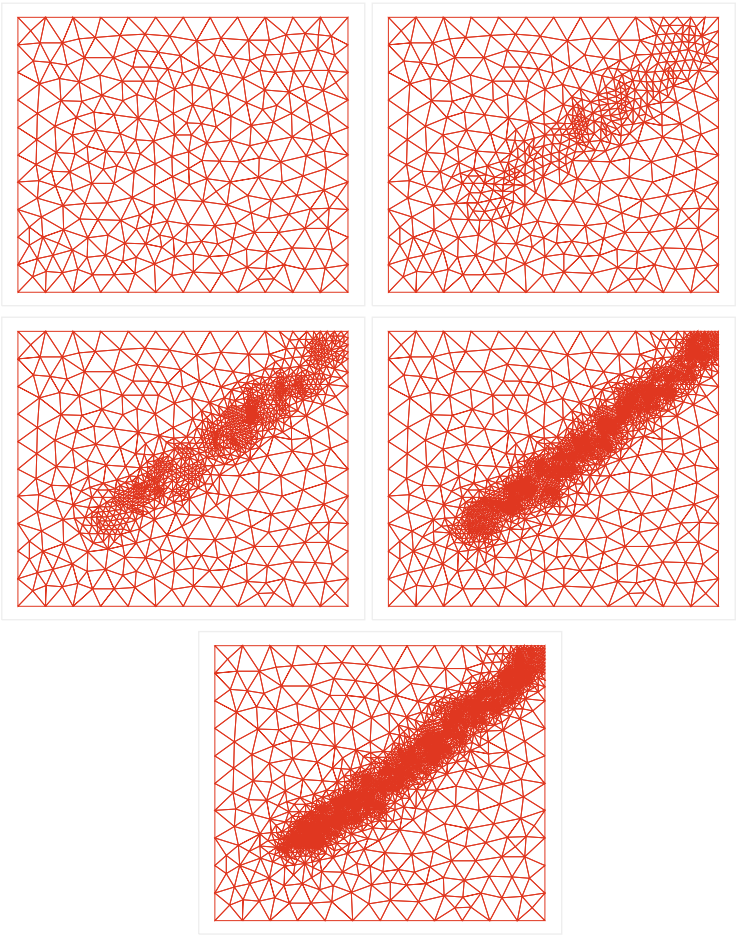

FIG. 17. Adaptive meshes for problem (4.2) with $p = 2$ at $t = 0.00, 0.25, 0.50, 0.75, 1.00$ (*upper left to lower right*)

REFERENCES

[1] S. Adjerid and M. Baccouch. The discontinuous Galerkin method for two-dimensional hyperbolic problems part I: Superconvergence error analysis. *J. Sci. Comput.*, 33(1):75–113, 2007.

[2] S. Adjerid and M. Baccouch. The discontinuous Galerkin method for two-dimensional hyperbolic problems part II: A posteriori error estimation. *J. Sci. Comput.*, 38(1):15–49, 2008.

[3] S. Adjerid and M. Baccouch. *A Posteriori* error analysis of the discontinuous Galerkin method for two-dimensional hyperbolic problems on unstructured meshes. *Computer Methods in Applied Mechanics and Engineering*, 200: 162–177, 2011.

[4] S. Adjerid, K. Devine, J. Flaherty, and L. Krivodonova. *A posteriori* error estimation for discontinuous Galerkin solutions of hyperbolic problems. *Computer Methods in Applied Mechanics and Engineering*, 191:1097–1112, 2002.

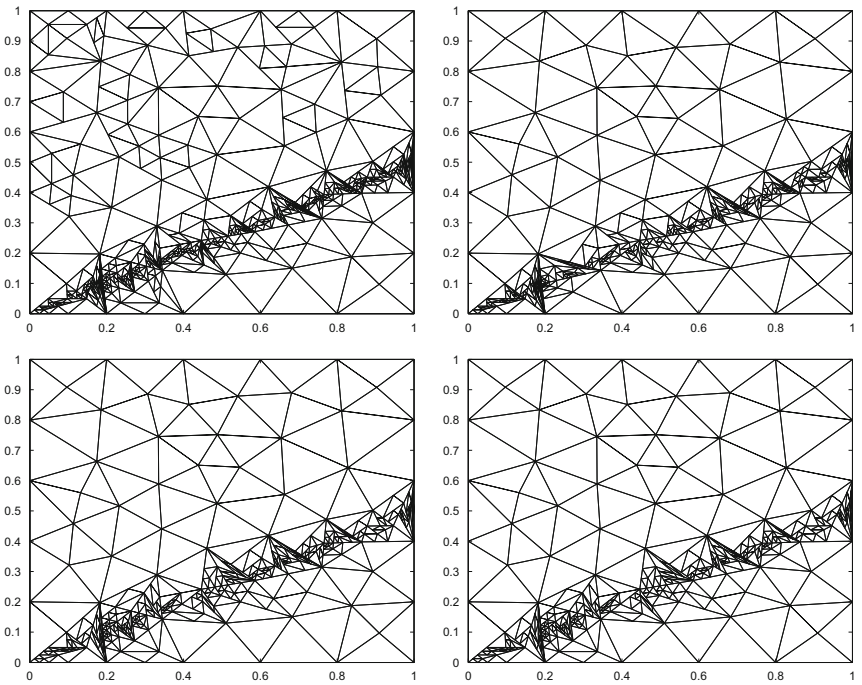

FIG. 18. Meshes generated by adaptive Algorithm 3 for problem (4.3) with $\delta = 0.001$ and $p = 1, 2, 3, 4$ (*upper left to lower right*)

[5] S. Adjerid and T. C. Massey. *A posteriori* discontinuous finite element error estimation for two-dimensional hyperbolic problems. *Computer Methods in Applied Mechanics and Engineering*, 191:5877–5897, 2002.

[6] S. Adjerid and I. Mechai. *A posteriori* discontinuous Galerkin error estimation on tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 201–204:157–178, 2012.

[7] S. Adjerid and T. Weinhart. Discontinuous Galerkin error estimation for linear symmetric hyperbolic systems. *Computer Methods in Applied Mechanics and Engineering*, 198:3113–3129, 2009.

[8] S. Adjerid and T. Weinhart. Asymptotically exact discontinuous Galerkin error estimates for linear symmetric hyperbolic systems. *Applied Numerical Mathematics*, in press, 2011.

[9] S. Adjerid and T. Weinhart. Discontinuous Galerkin error estimation for linear symmetrizable hyperbolic systems. *Mathematics of Computation*, 80: 1335–1367, 2011.

[10] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comp. Phys.*, 131:267–279, 1997.

[11] R. Biswas, K. Devine, and J. E. Flaherty. Parallel adaptive finite element methods for conservation laws. *Applied Numerical Mathematics*, 14:255–284, 1994.

[12] B. Cockburn, G. E. Karniadakis, and C. W. Shu, editors. *Discontinuous Galerkin Methods Theory, Computation and Applications, Lectures Notes in Computational Science and Engineering*, volume 11. Springer, Berlin, 2000.

[13] B. Cockburn, S. Y. Lin, and C. W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin methods of scalar conservation laws III: One dimensional systems. *Journal of Computational Physics*, 84:90–113, 1989.

[14] B. Cockburn and C. W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin methods for scalar conservation laws II: General framework. *Mathematics of Computation*, 52:411–435, 1989.

[15] K. D. Devine and J. E. Flaherty. Parallel adaptive hp-refinement techniques for conservation laws. *Computer Methods in Applied Mechanics and Engineering*, 20:367–386, 1996.

[16] K. Ericksson and C. Johnson. Adaptive finite element methods for parabolic problems I: A linear model problem. *SIAM Journal on Numerical Analysis*, 28: 12–23, 1991.

[17] J. E. Flaherty, R. Loy, M. S. Shephard, B. K. Szymanski, J. D. Teresco, and L. H. Ziantz. Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws. *Journal of Parallel and Distributed Computing*, 47:139–152, 1997.

[18] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for CFD*. Oxford University Press, New York, 1999.

[19] L. Krivodonova and J. E. Flaherty. Error estimation for discontinuous Galerkin solutions of two-dimensional hyperbolic problems. *Advances in Computational Mathematics*, 19:57–71, 2003.

[20] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, 1973.