INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 2, Number 2, Pages 163–178

A SLOPE LIMITING PROCEDURE IN DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR GASDYNAMICS APPLICATIONS

SHUANGZHANG TU AND SHAHROUZ ALIABADI

(Communicated by Peter Minev)

Abstract. In this paper we demonstrate the performance of a slope limiting procedure combined with a discontinuous Galerkin (DG) finite element solver for 2D compressible Euler equations. The slope limiter can be categorized into van Albada type and is differentiable. This slope limiter is modified from a similar limiter used in finite volume solvers to suit the needs of the DG solver. The gradient in an element is limited using the weighted average of the face gradients. The face gradients are obtained from the area-weighted average of the gradient on both sides of the faces. The slope limiting process is very suitable for meshes discretized by triangle elements. The HLLC (Harten, Lax and van Leer) or the local Lax-Friedrich (LLF) flux functions is used to compute the interface fluxes in the DG formulation. The second order TVD Runge-Kutta scheme is employed for the time integration. The numerical examples including transonic, supersonic and hypersonic flows show that the current slope limiting process together with the DG solver is able to remove overshoots and undershoots around high gradient regions while preserving the high accuracy of the DG method. The convergence histories of all examples demonstrate that the limiting process does not stall convergence to steady state as many other slope limiters do.

Key Words. Discontinuous Galerkin (DG), slope limiting and 2D inviscid compressible flows.

1. Introduction

Discontinuous Galerkin (DG) method has been gaining popularity in computational fluid dynamics (CFD) in recent years [8, 9, 27]. Indeed, the DG method can be considered as a mixture of classic finite element method (FEM) and finite volume method (FVM). In the DG method, the advantageous features of the FEM and FVM are combined resulting in a robust and accurate numerical scheme for

Received by the editors January 1, 2004 and, in revised form, April 22, 2004.
2000 *Mathematics Subject Classification.* 35L65, 76M10, 76N15.
This work is funded in part by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory contract number DAAD19-01-2-0014. Partial support for this publication made possible through support provided by DoD High Performance Computing Modernization Program (HPCMP) Programming Environment & Training (PET) activities through Mississippi State University under the terms of Agreement No. # GS04T01BFC0060. Views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army and Department of Defense position, policy, or decision unless so designated by other official documentation and no official endorsement should be inferred.

solution of problems involving shocks and other discontinuities. The variational form for each element is obtained by multiplying the governing equations with a test function and integrating conservative fluxes by parts. This results in boundary fluxes normal to the element interfaces. The inviscid flux is upwinded using an approximate Riemann solver.

Compared to the stabilized FEM, such as the streamline-upwinding/Petrov-Galerkin (SUPG) [1, 2, 3], the DG method is capable of sharper representation of the discontinuities in the solutions. In the DG method, the solution across each element can be discontinuous, therefore the DG method is naturally a better solution strategy for problems involving shocks and discontinuities. The DG method also eliminates the need for SUPG stabilization in advection dominant flows. In the DG method, the upwind fluxes provide the necessary stabilization. In addition, the *hp* refinement can easily be implemented in this method [9, 20] because hanging nodes are allowed in the DG method. The DG method is more compact than the FVM. In the finite volume method, the reconstruction within a cell relies on a cluster of neighboring cells using the path integral method or the least square method. If higher spatial accuracy order is desired in FVM, the number of supporting cells has to be increased. In contrast, in the DG method, linear or higher order interpolation functions can be employed to obtain the solutions at any points inside the element. The supporting elements are the same regardless of the spatial accuracy. This compactness makes the DG method more stable and easier to implement than the finite volume method. However, compared to the stabilized Galerkin finite element formulations, the DG methods require the solutions of systems of equations with more unknowns. However, if high order elements are used in the DG method, a very coarse mesh can be used to attain sufficient solution resolution [6]. Therefore the disadvantage of the DG method can be outweighed by its outstanding advantages.

It is well known that the nonphysical oscillations around high gradient discontinuities exist for linear stabilization techniques [1]. The oscillations are sometimes severe enough to cause stability problem. A discontinuity capturing [1] or an appropriate limiter is a common cure for this problem. Aliabadi and Tu [4] use discontinuity capturing method commonly used in SUPG/GLS finite element solvers as an alternative to slope limiters in their DG solver for 2D transient Euler equations. Sun and Takayama [22] employ a smoothing step in addition to the advection step to smooth the solution around high gradient regions. Their method is suitable for quadrilateral grids. The main defect of the methods mentioned above is that they usually require some user-defined parameters making them problem dependent. Many slope limiters used in the finite volume methods can be modified to meet the needs of the DG method. Slope limiters are usually parameter free. In [8], a limiter using maxmod functions is presented. Their limiter has the advantage over usual minmod based limiters [21], since it would not flatten smooth extrema. Hoteit et al. [17] introduce an extension of van Leer's slope limiter for two-dimensional DG method using unstructured quadrangular or triangular meshes. Unfortunately, there are two drawbacks with the use of slope limiters. One is the possible accuracy degradation in smooth regions; the other is that the convergence may be severely hampered. The main reason is that the limiters presented above are non-differentiable. It could be active in near uniform region. To overcome these drawbacks, some techniques have been adopted in practice. Venkatakrishnan [28] devised a new limiter that may accelerate the convergence rate. But it is not monotone. In addition, the constant in Venkatakrishnan's limiter is case independent.

De Zeeuw [30] freezes the limiter after some iterations. Instead of doing this, Delanaye [11] suggest using the so-called *historic modification* of the limiter to improve the convergence. The limiter presented in [9] takes into account the improvement at smooth regions, however we are unaware of its performance in steady-state simulations. Jawahar and Kamath [19] present a van Albada-typed limiting procedure for their finite volume solver. The van Albada-typed limiter is differentiable. Their numerical results show excellent performance in removing oscillations while not stalling convergence and not clipping the smooth extremum. In the present paper, the authors will slightly modify the limiter of Jawahar and Kamath to suit the needs of current DG solver for compressible flows. The whole limiting procedure contains five steps. Also, one step in the original limiter of Jawahar and Kamath is found unnecessary at least in current DG solver. The numerical examples presented in this paper will demonstrate this limiting process is very robust for 2D inviscid Euler flows ranging from subsonic to hypersonic.

The next section will briefly describe our DG method formulation where the focus will be put on the spatial integration scheme. Following that section is the brief description of the HLLC (Harten, Lax and van Leer) and local Lax-Friedrich (LLF) flux functions. The Runge-Kutta time integration schemes, time step determination and residual computation will be given in a new section. And then the detailed explanation of present limiting procedure will be given. Finally, a few numerical examples will be presented to verify the present limiting process incorporated into current DG solver. These examples include steady-state transonic, supersonic and hypersonic flows.

2. Formulation of discontinuous Galerkin method

The unsteady, compressible Euler equations can be written in the following conservative form as:

$$\frac{\partial \mathbf{U}(\mathbf{x},t)}{\partial t} + \frac{\partial \mathbf{F}_j(\mathbf{U}(\mathbf{x},t))}{\partial x_j} = 0, (\mathbf{x},t) \in \Omega \times (0,T) \tag{1}$$

where the repeated indices represent summation convention. Here $\mathbf{x} \in \Omega$ is the triangulated spatial domain with boundary $\partial\Omega$ and $t \in (0,T)$ is the time domain. $\mathbf{U}$ represents the conservative state vector and $\mathbf{F}_j$ the inviscid flux vectors. They can be expressed as:

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho E \end{bmatrix}, \mathbf{F}_j = \begin{bmatrix} \rho u_j \\ \rho u_i u_j + \delta_{ij} p \\ (\rho E + p) u_j \end{bmatrix}. \tag{2}$$

An initial condition, $\mathbf{U}_0(\mathbf{x}) = \mathbf{U}(\mathbf{x}, t=0)$, and appropriate boundary conditions on $\partial\Omega$ must also be given. All quantities appearing in Equations (1) and (2) have been nondimensionalized using a set of appropriate reference values. In the present DG implementation, the same iso-parametric shape functions as in continuous finite element method are used. The degrees of freedom are the unknown conservative variables $\mathbf{U}$ placed on the vertices of each element (see Figure 1). Since arbitrary number of elements can share a single vertex, the solution on each vertex can thus have multiple solutions. In this sense, the DG solution is allowed to be discontinuous across element interfaces. As will be explained in the section on limiting procedure, common nodes shared by neighboring triangles (the filled circle in Figure 1) and triangle centroids (the empty circle in Figure 1) stores primitive variables $\mathbf{V} = (\rho, u_i, e)$ which can be derived from the conservative state vector.

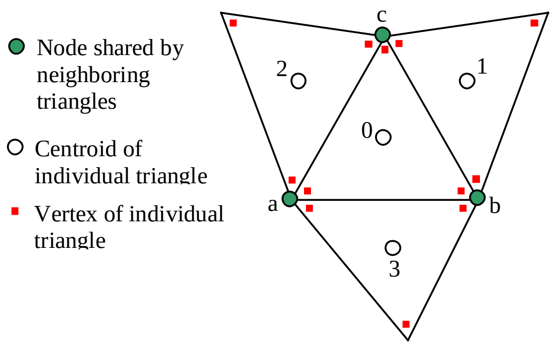

Node shared by neighboring triangles
Centroid of individual triangle
Vertex of individual triangle
c
a
b
0
1
2
3

FIGURE 1. Storing locations in the present DG implementation.

The DG finite element formulation is written for each element Ω^e. Equation (1) is first multiplied with test functions $\mathbf{W}^h$, and the flux integrals are further integrated in parts. The whole process yields

$$\int_{\Omega^e} \mathbf{W}^h \cdot \frac{\partial \mathbf{U}^h}{\partial t} d\Omega - \int_{\Omega^e} \frac{\partial \mathbf{W}^h}{\partial x_j} \cdot \mathbf{F}_j^h d\Omega + \int_{\partial\Omega^e} \mathbf{W}^h \cdot \hat{\mathbf{F}}_{\mathbf{n}}^h d\Gamma = 0 \tag{3}$$

where $\hat{\mathbf{F}}_{\mathbf{n}}$ is the upwind boundary flux across the element interface and $\mathbf{n}$ is the outward unit normal of the element boundary surface.

In this paper, linear triangular elements are used to discretize the 2D spatial domain. Hence, the solution at any point in each element can be represented in terms of the vertex solutions and the shape functions at each vertex:

$$\mathbf{U}^h(\mathbf{x}, t)|_{\Omega^e} = \sum_{m=1}^{3} \mathbf{U}_m^h(t) w_m(\mathbf{x})|_{\Omega^e} \tag{4}$$

where the subscript m is the index for each vertex of the element. By replacing $\mathbf{W}^h$ with interpolation functions at each vertex n, $w_n(\mathbf{x})$ $(n = 1, 2, 3)$ and $\mathbf{U}^h$ with Equation (4), we can obtain the following formulation for each component of $\mathbf{U}^h$ on the vertices of the element:

$$\frac{dU_m^h}{dt} \int_{\Omega^e} w_n w_m d\Omega = \int_{\Omega^e} \frac{\partial w_n}{\partial x_j} F_j^h d\Omega - \int_{\partial\Omega^e} w_n \hat{F}_{\mathbf{n}}^h d\Gamma \tag{5}$$

where $m, n = 1, 2, 3$. U and F now become the component of vectors $\mathbf{U}$ and $\mathbf{F}$. The integral on the left hand side of Equation (5) constitutes the components of the 3×3 mass matrix $\mathbf{M}$. Obviously, $\mathbf{M}$ is symmetric. The right hand side is noted as $\mathbf{R}$. Therefore, for straight-sided triangular elements, Equation (5) can be rewritten as:

$$\frac{d}{dt} \begin{pmatrix} U_1^h \\ U_2^h \\ U_3^h \end{pmatrix} = \mathbf{M}^{-1} \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix}. \tag{6}$$

If we transformed the physical triangles of arbitrary shape into the local regular right triangle, the mass matrix has the unique analytic form scaled by the determinant of the transformation Jacobian. The inverse of the mass matrix can be computed by hand in advance and stored. The present explicit formulation is efficient in this sense.

3. Flux functions

The boundary flux $\hat{\mathbf{F}}_\mathbf{n}$ in Equation (3) can be evaluated using any kind of upwind numerical flux functions. Since $\hat{\mathbf{F}}_\mathbf{n}$ is normal to the element interface and discontinuities are allowed across the interface, a local Riemann problem can be solved for the fluxes, given the left and the right states $\mathbf{U}_l$ and $\mathbf{U}_r$. Therefore, various approximate Riemann solvers can be used to compute $\hat{\mathbf{F}}_\mathbf{n}$. This is exactly the same as in a finite volume solver.

3.1. HLLC flux. In this paper, two flux functions have been implemented. One is the HLLC Riemann solver by Toro [23] which is an improved two-state version of the original single-state HLL Riemann solver by Harten, Lax and van Leer [16]. The original HLL numerical flux automatically satisfies entropy inequality [10] resolves isolated shocks [16] and preserves positivity [13]. Batten et al. [7] have shown that the HLLC flux not only preserves the advantageous features of the HLL flux, but also resolves isolated contact discontinuities exactly. This explains 'C' in HLLC stands for *contact discontinuity.* Besides, the implementation of the HLLC Riemann solver is relatively easier and the computational cost is lower compared with many other available Riemann solvers. In addition to the original authors, many other authors [25, 26, 29] successfully applied this Riemann solver to their problems.

According to References. [7, 23], the HLLC flux can be expressed as:

$$\mathbf{F}_{HLLC} = \begin{cases} \mathbf{F}_l & S_L < 0 \\ \mathbf{F}_l^* & S_L \le 0 < S_M \\ \mathbf{F}_r^* & S_M \le 0 \le S_R \\ \mathbf{F}_r & S_R < 0 \end{cases} \tag{7}$$

where the wave speeds are estimated as:

$$S_L = \min[q_l - a_l, \tilde{q} - \tilde{a}], \tag{8a}$$

$$S_R = \max[q_r + a_r, \tilde{q} + \tilde{a}], \tag{8b}$$

$$S_M = \frac{\rho_r q_r (S_R - q_r) - \rho_l q_l (S_L - q_l) + p_l - p_r}{\rho_r (S_R - q_r) - \rho_l (S_L - q_l)}. \tag{8c}$$

and q and a are the speed normal to the element interface and speed of sound, respectively. The tilded values are evaluated using Roe-averaged quantities as in the Roe's approximate Riemann solver. Here $\mathbf{F}_l^*$ and $\mathbf{F}_r^*$ in Equation (7) are computed according to:

$$\mathbf{F}_l^* = \mathbf{F}_l + S_L(\mathbf{U}_l^* - \mathbf{U}_l) \tag{9a}$$

$$\mathbf{F}_r^* = \mathbf{F}_r + S_R(\mathbf{U}_r^* - \mathbf{U}_r) \tag{9b}$$

where $\mathbf{F}_l = \mathbf{F}(\mathbf{U}_l)$ and $\mathbf{F}_r = \mathbf{F}(\mathbf{U}_r)$. Substituting $\mathbf{F}_l^* = \mathbf{F}(\mathbf{U}_l^*)$ and $\mathbf{F}_r^* = \mathbf{F}(\mathbf{U}_r^*)$ into Equations (9a) and (9b) to first compute $\mathbf{U}_l^*$ and $\mathbf{U}_r^*$, respectively, before $\mathbf{F}_l^*$ and $\mathbf{F}_r^*$ can be computed. For more details, refer to [7].

3.2. Local Lax-Friedrich (LLF) flux. Another numerical flux function employed in the present paper is the simple local Lax-Friedrich (LLF) flux [24] also known as Rusanov flux. The LLF flux is more dissipative than the HLLC flux but is more robust, especially for problems involving low velocities regions such as the region near the stagnation point of supersonic/hypersonic flows around blunted bodies. The convergence can be improved using the LLF flux. The LLF flux can be expressed as

$$\mathbf{F}_{LLF} = \frac{1}{2}[\mathbf{F}_l + \mathbf{F}_r - (|q^*| + a^*)(\mathbf{U}_r - \mathbf{U}_l)] \tag{10}$$

where q^* is the velocity normal to the interface and a^* the speed of sound at the interface. q^* and a^* are determined from the arithmetic average of the left and the right states $(\mathbf{U}_l+\mathbf{U}_r)/2$. $|q^*|+a^*$ represents the largest wave speed in the direction normal to the interface.

4. Time integration

The second order accurate two-stage TVD Runge-Kutta schemes of Cockburn and Shu [9] is employed in this work. According to [9], the optimal two-stage, second order accurate TVD Runge-Kutta scheme is expressed as:

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \Delta t\mathbf{R}(\mathbf{U}^n) \tag{11a}$$

$$\mathbf{U}^{n+1} = \frac{1}{2}\mathbf{U}^n + \frac{1}{2}\mathbf{U}^{(1)} + \frac{1}{2}\Delta t\mathbf{R}(\mathbf{U}^{(1)}) \tag{11b}$$

where $\mathbf{R}(\mathbf{U})$ is composed of the right hand side of Equation (6).

The time step for a two dimensional element j is determined according to [11], namely,

$$\Delta t_j \leq \text{CFL}\frac{\Omega_j}{\sum\limits_k l_k \left|\min(\mathbf{u}\cdot\mathbf{n} - a, 0)\right|} \tag{12}$$

where $\mathbf{u}$ is the velocity vector. Here $\mathbf{u}$ and a are evaluated at the centroid of the element for simplicity, k is the index of the faces surrounding the element, l_k is the length of the face and Ω_j is the area of the element. Numerical experience demonstrates that the time step determined by Equation (12) is stable for all cases considered in this paper. In practice, the CFL number is initially taken to be a relatively small value, say 0.25. If no stability problem is encountered in consecutive 100 time steps, it will be divided by 0.9 to obtain a larger value. CFL number can be increased continually until the user-defined maximum CFL number is reached. This technique can stabilize the initial time steps and accelerate the convergence when the solution is close to steady-state. To accelerate the convergence toward steady solutions, local time stepping [5] is also used.

Since the present paper is focused on the convergence performance of the proposed limiting procedure in a DG finite element solver, the computation of the numerical residual must be given. The residual is computed according to

$$\text{Residual} = \sqrt{\frac{\sum\limits_{i=1}^{ndf}\sum\limits_{j=1}^{nen}\sum\limits_{k=1}^{ne}\left(\frac{U_{ijk}^{n+1}-U_{ijk}^{n}}{\Delta t}\right)^2}{ndf\times nen\times ne}} \tag{13}$$

where U is the component of the conservative state vector $\mathbf{U}$, ndf is the number of components in $\mathbf{U}$ (=4 for 2D Euler equations), nen is the number of vertices in an element (=3 for triangles) and ne is the total number of elements. When the residual drops to 10^{-6}, the steady-state solution is assumed to have been reached.

5. Slope limiting procedure

Before we proceed to explain the limiting process in present DG solver, the following must be stressed first:

- The limiting process is based on primitive variables $\mathbf{V} = (\rho, u, v, e)$ where ρ, u, v, and e are density, x-component velocity, y-component velocity and specific internal energy, respectively.

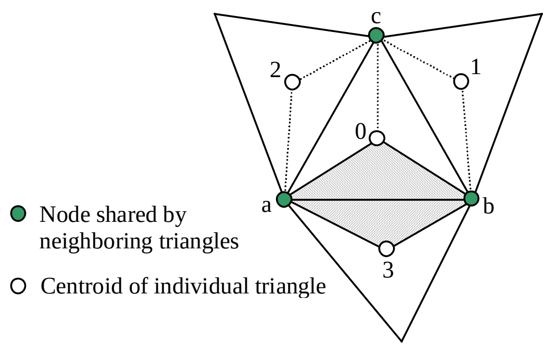

c
2
1
0
a
b
3
● Node shared by neighboring triangles
○ Centroid of individual triangle

FIGURE 2. Stencil for slope limiting procedure.

- The limiting process is conservative. The average solution for each element is kept unmodified after the limiting process. We denote each component of $\mathbf{U}$ by U and the average of the finite element solution U^h over an element Ω^e by $\bar{U}_e$, that is,
$$\bar{U}_e = \frac{1}{|\Omega^e|}\int_{\Omega^e} U^h d\Omega$$
For example, considering the linear triangular element Δabc with centroid 0 and vertices a, b and c shown in Figure 2, we have
$$\bar{U}_0 = \frac{1}{3}\sum_{i=1}^{3} U_{i,0} \tag{14}$$
where $U_{i,0}$ is the conservative solution at local ith vertex of element 0. The limiting process may change $U_{i,0}$ but must not change $\bar{U}_0$.
- The limiting process is modified from the limiter of Jawahar and Kamath [19]. The original limiter of Jawahar and Kamath was designed for the unstructured finite volume solver. In this paper, the limiter is modified slightly to suit the needs of the DG solver. Also, the authors of present paper found that a step in the original limiting procedure of Jawahar and Kamath is unnecessary for present DG method.

Our goal is to limit the solutions at vertices of each element. Since three non-collinear points determine a gradient plane uniquely, we can first limit the solution gradients in the element and reconstruct the vertex solutions using the limited element gradients. The limiting procedure contains the following five steps.

Step 1: Compute the solution at shared common nodes and element centroids.

In cell-centered finite volume solvers, since the degree of freedoms are placed at centroids of elements, an inverse-distance or a pseudo-Laplacian procedure [19] is usually employed to obtain vertex values from the corresponding cell-centered values. In current DG solver, however, the degrees of freedom are put on the vertices of each element, the solution on shared nodes can be easily obtained by taking the arithmetic average of those vertex values belonging to neighboring elements sharing the node. The solution at the element centroid can be obtained using Equation (14) for linear triangular elements. Since the limiting process is performed on primitive variables $\mathbf{V}$, the node solution and the centroid solution obtained in this step must be transformed to primitive variables.

Step 2: Compute the gradient across each element interface.

Consider the computation of the gradient across the element interface $\overline{ab}$ in Figure 2. The gradient across $\overline{ab}$ can be obtained in an averaging way by solving the following linear system:

$$\begin{bmatrix} x_3 - x_0 & y_3 - y_0 \\ x_a - x_b & y_a - y_b \end{bmatrix} \begin{bmatrix} (V_x)_{\overline{ab}} \\ (V_y)_{\overline{ab}} \end{bmatrix} = \begin{bmatrix} V_3 - V_0 \\ V_a - V_b \end{bmatrix}. \tag{15}$$

This method was also used by Frink [14] to evaluate the viscous flux. We can prove that this averaging is equivalent to the area-weighted averaging explained in [19]. Both averaging methods lead to the same expressions for the face gradients, namely,

$$(V_x)_{\overline{ab}} = \frac{1}{2A_{a3b0}}[(V_3 - V_0)(y_b - y_a) + (V_a - V_b)(y_3 - y_0)] \tag{16a}$$

$$(V_y)_{\overline{ab}} = -\frac{1}{2A_{a3b0}}[(V_3 - V_0)(x_b - x_a) + (V_a - V_b)(x_3 - x_0)] \tag{16b}$$

where $A_{a3b0} = A_{a3b} + A_{b0a}$. The gradients across $\overline{bc}$ and $\overline{ca}$ are computed in the same way. For faces located on physical boundaries, the solution at ghost elements is used to provide the same stencil to compute the face gradients.

Step 3: Compute the unlimited gradient in each element.

The unlimited gradients in element 0 is constructed using the area-weighted average of the corresponding face gradients as:

$$(\nabla V)_0 = \frac{A_{a3b0}(\nabla V)_{\overline{ab}} + A_{b1c0}(\nabla V)_{\overline{bc}} + A_{c2a0}(\nabla V)_{\overline{ca}}}{A_{a3b0} + A_{b1c0} + A_{c2a0}} \tag{17}$$

where $\nabla V = (V_x, V_y)$.

Step 4: Compute the limited gradient in each element.

After the unlimited gradients in all elements have been computed, the limited gradients in an element can be computed by taking the weighted average of the unlimited gradients in elements surrounding the element considered. For example, the limited gradients in element 0 in Figure 2 can be computed according to

$$(\nabla V)_0^l = w_1(\nabla V)_1 + w_2(\nabla V)_2 + w_3(\nabla V)_3 \tag{18}$$

where the weights are

$$w_1 = \frac{g_2 g_3 + \epsilon}{g_1^2 + g_2^2 + g_3^2 + 3\epsilon} \tag{19a}$$

$$w_2 = \frac{g_1 g_3 + \epsilon}{g_1^2 + g_2^2 + g_3^2 + 3\epsilon} \tag{19b}$$

$$w_3 = \frac{g_1 g_2 + \epsilon}{g_1^2 + g_2^2 + g_3^2 + 3\epsilon} \tag{19c}$$

and g_1, g_2 and g_3 are chosen as the square of the L_2 norm of the unlimited element gradients, i.e., $g_1 = \|(\nabla V)_1\|^2$, $g_2 = \|(\nabla V)_2\|^2$ and $g_3 = \|(\nabla V)_3\|^2$. ϵ is a small number introduced to prevent indeterminacy and set to be 10^{-10} throughout current work.

The weights in Equation (19) will become 1/3 when the three element gradients are equal. This limiting process formulated above is similar to the 1D van Albada limiter. The limiter is differentiable, which is a desirable property for limiters for convergence purpose. In the original limiting process by Jawahar and Kamath [19], an additional step is required to compute the unlimited gradients in an element, i.e., the unlimited gradients of element 0 are taken to be the area-weighted average of the gradients corresponding to its three neighboring elements which are computed from Equation (17). The authors of the present paper do not think this step is necessary at least for the current DG solver. Care must be taken when computing the limited gradient in boundary elements where a face gradient must be substituted for an element gradient in Equation (18).

Step 5: Compute the limited conservative variable at vertices of each element.

After the gradients in an element have been limited according to Equation (18), the local vertex solution can be reconstructed to satisfy each component of the limited gradient vector and the unchanged element centroid value. Since in the present DG solver, the local vertices of an element store conservative variables, the limited gradients of primitive variables, $(\nabla V)_0^l$, must be transformed to gradients of conservative variables, $(\nabla U)_0^l$, using the solution at the centroid. The reconstruction is unique for linear triangular elements. For example, for element 0 in Figure 2, we have

$$\begin{cases} \sum_{i=1}^{3} \hat{U}_{i,0}(w_{i,0})_x = (U_x)_0^l \\ \sum_{i=1}^{3} \hat{U}_{i,0}(w_{i,0})_y = (U_y)_0^l \\ \sum_{i=1}^{3} \hat{U}_{i,0} = 3\bar{U}_0 \end{cases} \tag{20}$$

where $\hat{U}_{i,k}$ represents the reconstructed solution at the ith vertex of the kth element and $(w_{i,k})_x$ and $(w_{i,k})_y$ are the derivatives of shape functions.

6. Numerical results

In this section, a few 2D Euler numerical examples are presented to demonstrate the effectiveness of the limiting procedure combined with current DG finite element solver for compressible flows. The following numerical examples will show that the present limiting process is able to remove the overshoots and undershoots across high gradient regions. Also will be shown is the limiting process together with current DG solver do not hamper the convergence to steady state as many other limiters do.

6.1. Supersonic inviscid flow passing a ramp. This case is about the supersonic flow with Mach number 2 around a ramp with 10° slope. The steady-state solution contains an oblique shock. Exact solution exists for this simple case under the inviscid assumptions. The HLLC flux is used in this example. We use an unstructured mesh consisting of 1615 triangles. Figure 3 shows the mesh (left picture) and the pressure contours at the steady state (right picture). The oblique shock has been captured with no numerical oscillations. The left picture in Figure 4 shows the pressure distributions at the bottom wall compared with the analytic solution. Excellent agreement can be observed. No overshoots and undershoots appear around regions of high gradients. The convergence history of the proposed limiting process is shown in the right picture in Figure 4. The residual is computed according to Equation (13).

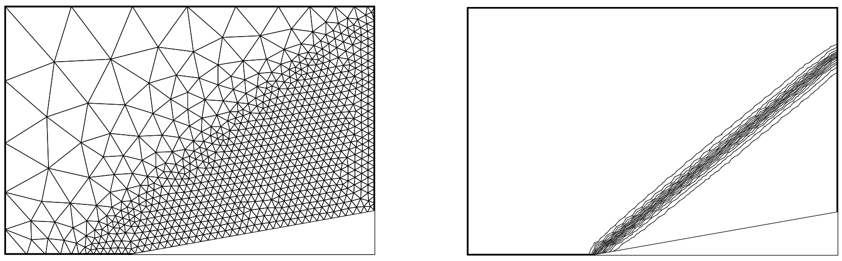


FIGURE 3. Supersonic flow (M = 2) passing a 10° ramp. Left: mesh; right: pressure contours.

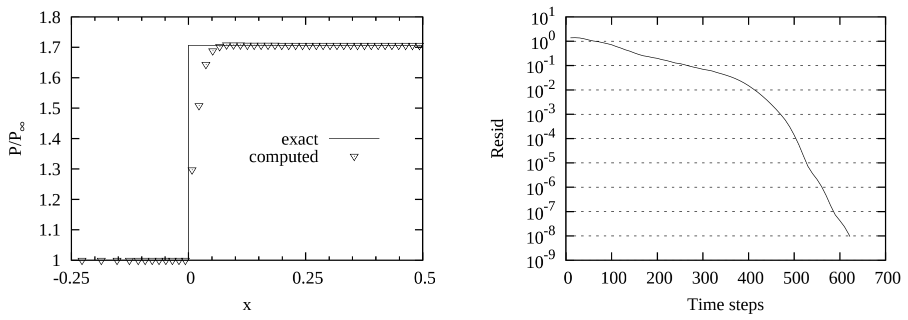


FIGURE 4. Supersonic flow (M = 2) passing a 10° ramp. Left: pressure distribution; right: convergence history.

6.2. Transonic flow around NACA0012 airfoil. This case is about a subsonic flow at Mach number 0.85 around the NACA0012 airfoil with 1° angle of attack. Due to the acceleration of the subsonic flow on curved airfoil surface, the flow will reach supersonic on part of the surface. Hence, supersonic regions will be pocketed in a subsonic region, making the overall flow transonic. This case was tested by Jawahar and Kamath in [19] to verify the performance of their slope limiters in a finite volume solver.

The computational domain shown in Figure 5 is about 25 chord lengths away from the airfoil. The unstructured mesh consists of 7171 triangles. There are 128 points put on the airfoil surface. Figure 6 shows the pressure and Mach number contours near the airfoil surface. As can be clearly seen, two shocks appear on the surface. The shock at the upper surface is stronger and located at a more downstream place than the shock on the lower surface due to the angle of attack. The pressure coefficient shown on Figure 7 (left) compares the current solution with that in [19]. Good agreement can be seen in terms of the location and strength of shocks. The convergence history in Figure 7 (right) demonstrates a steady drop of the residual below 10^{-3}. The initial oscillation is typical of transonic flow simulations in which the shocks oscillate about their steady-state location before settling down. The convergence performance is similar to that in [19].

6.3. Supersonic inlet flow. Supersonic inlet flows are typical of scramjet engines. The configuration shown in the top picture in Figure 8 is taken from [18]. A supersonic inflow at Mach number 3 enters the engine inlet from the left hand side. Due to the special configuration inside the inlet, complex flow features are expected to appear. The unstructured mesh consists of 8358 triangles. The HLLC

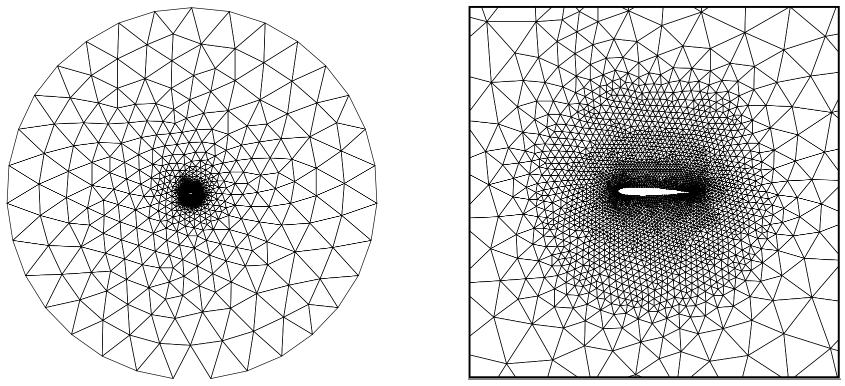


FIGURE 5. Mesh around NACA0012 airfoil. Left: whole; right: close-up near the airfoil.

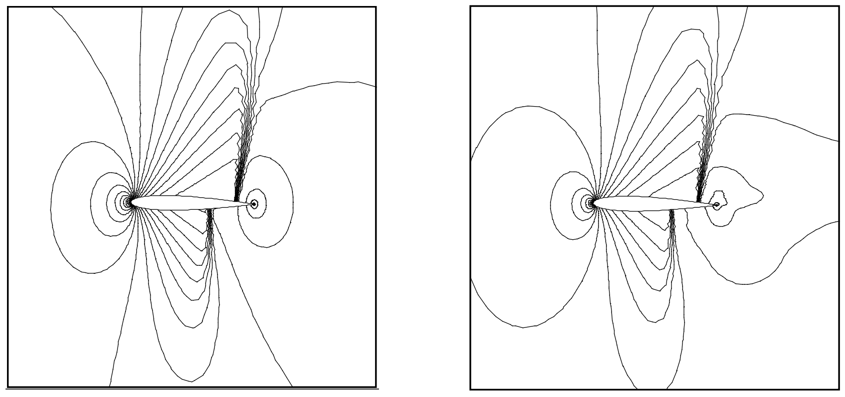


FIGURE 6. Transonic flow (M = 0.85, angle of attack $\alpha = 1°$) around NACA0012 airfoil. Left: pressure contours; right: Mach number contours.

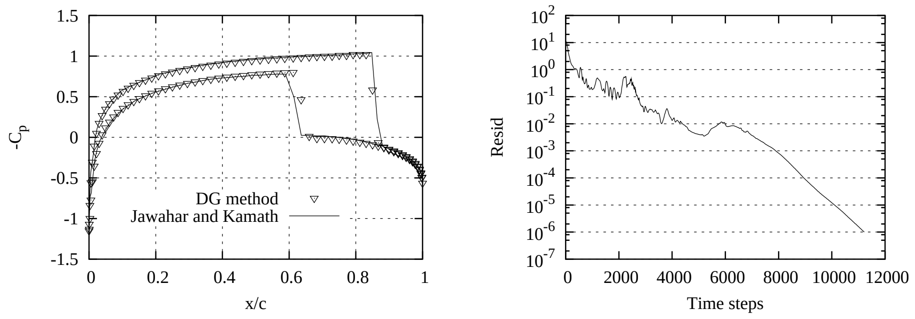


FIGURE 7. Transonic flow (M = 0.85, angle of attack =1°) around NACA0012 airfoil. Left: pressure coefficient distribution; right: convergence history.

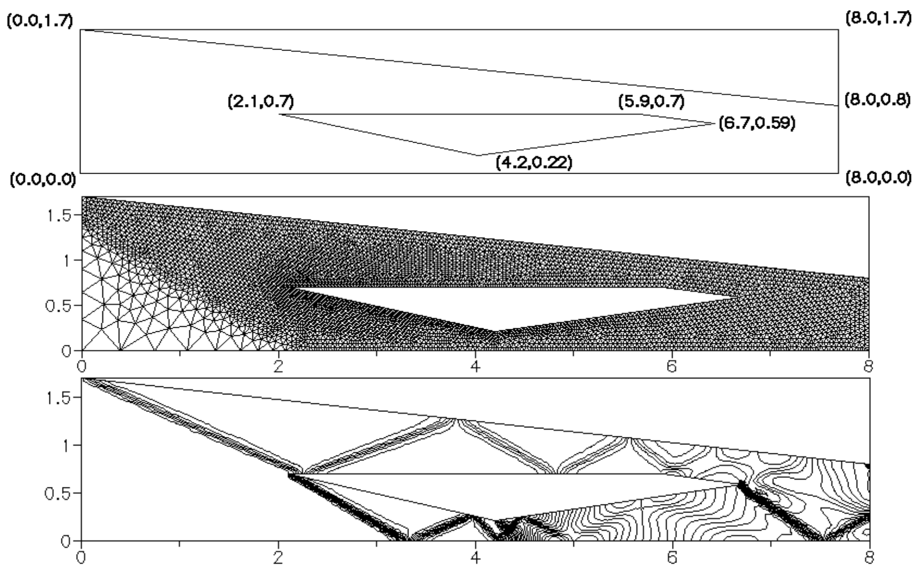


FIGURE 8. Supersonic inlet flow (M = 3). Top: geometry; middle: mesh; bottom: Mach number contours.

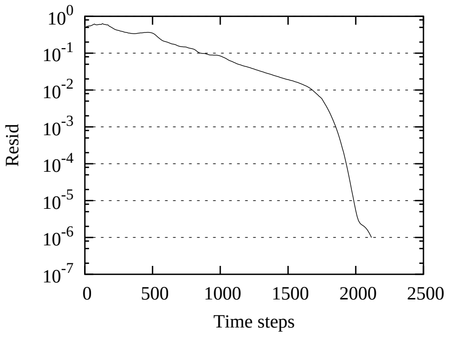


FIGURE 9. Convergence history of supersonic inlet flow (M = 3).

Riemann solver is used to calculate the interface flux. The Mach number contours shown in Figure 8 shows the captured complex flow structure. The flow features are qualitatively similar to those in [18]. Figure 9 shows the convergence history where a steep drop can be seen after the residual drops below 10^{-2}.

6.4. Hypersonic flow around a blunted body. Blunted body is essential in hypersonic flow conditions to reduce the heat transfer. Current case is a hypersonic flow around a 2D sphere-cone with the flight conditions M = 15, $P_\infty = 170\ N/m^2$, $\rho_\infty = 0.002\ kg/m^3$ at zero angle of attack. The flight conditions correspond with atmospheric conditions at an altitude of 45 km. This case has been tested for both perfect gas and equilibrium situations in Referencs [12, 15, 25]. In this case, the local Lax-Friedrich (LLF) flux is used to compute the interface flux due to the existence of the low velocity stagnation region. The LLF flux is very robust for these cases with the price of being more dissipative than the HLLC flux. Due to symmetry, only half of the domain is simulated. Figure 10 shows the unstructured

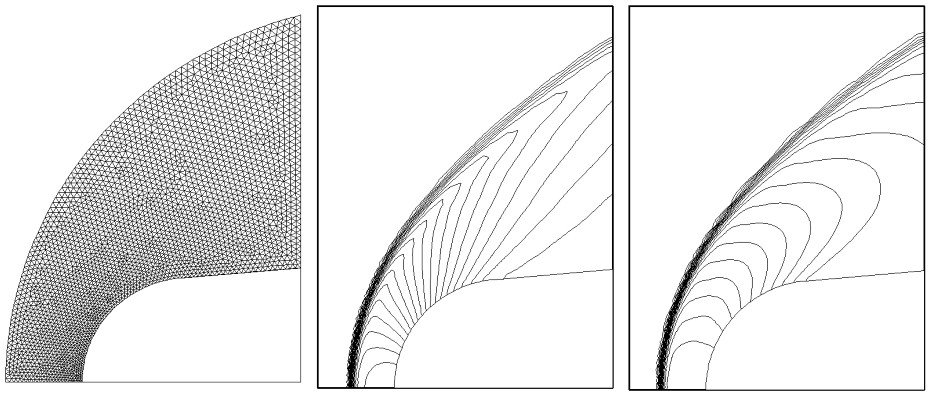
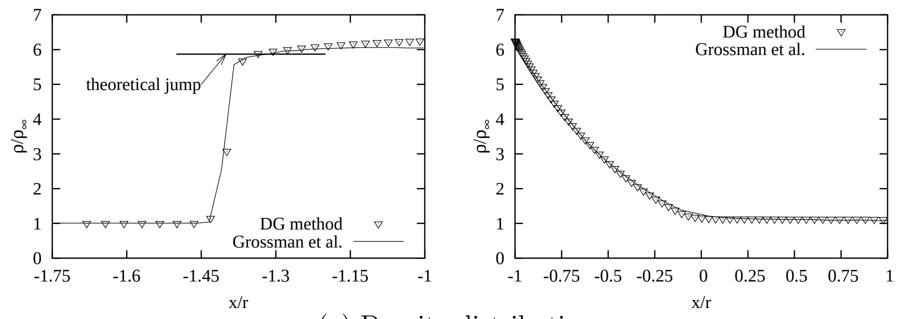


FIGURE 10. Hypersonic flow (M = 15) around a 2D sphere-cone body. Left: mesh; middle: pressure contours; right: temperature contours.

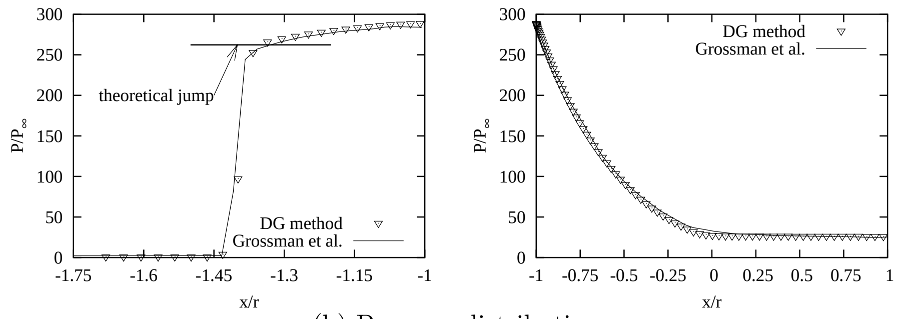
triangular grid, pressure and temperature contours. The detached shock has been captured very nicely and no wiggles are seen on contour plots. Figure 11 shows the solutions along the stagnation streamline and on the surface. The shock intersecting with the stagnation streamline can be considered as a normal shock. Theoretical jumps across the normal shock are also plotted on the corresponding figures. As can be seen, the current DG solver combined with the proposed limiting process is able to predict the jumps accurately without overshoots and undershoots. For comparison purpose, the solutions in [15] are also plotted on the same figures. The present solution shows good agreement with the solution from reference. The convergence history shown in Figure 12 shows a steady drop of residual almost from the beginning.

7. Conclusion

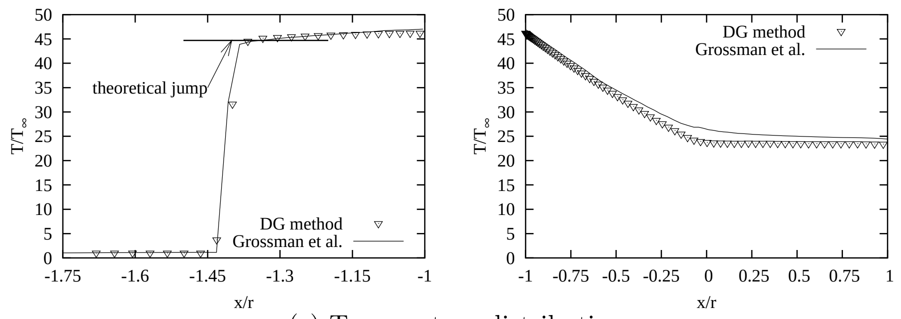
In this paper, a limiting process is incorporated into a discontinuous Galerkin (DG) finite element solver for compressible flows. The DG solver is described in detail in the first section. The HLLC (Harten, Lax and van Leer) and local Lax-Friedrich (LLF) flux functions are implemented in the current solver. The second order two-stage TVD Runge-Kutta time integration schemes are employed. A detailed explanation about the limiting process is given. The limiting process contains five steps and aimed to remove the possible oscillations across high gradient regions in solutions obtained from the DG solver. The gradient in an element is limited using the weighted average of face gradients. The face gradients are obtained from the area-weighted average of the gradient on both sides of faces. The limiter can be categorized into van Albada type. The limiting process is very suitable for comforming triangle meshes for 2D cases. Finally, a few numerical examples are presented to demonstrate the excellent performance of the present limiting process. These examples cover flows ranging from subsonic to hypersonic. As can be concluded from these examples, the present limiting process is able to suppress overshoots and undershoots around high gradient regions. In addition, the limiting process does not stall the convergence as many other slope limiters do. Combined with the present limiting process, current DG solver is very robust with high accuracy for inviscid compressible flows.



(a) Density distribution.



(b) Pressure distribution.



(c) Temperature distribution.

FIGURE 11. Hypersonic flow (M = 15) around a 2D sphere-cone body. Left column: solutions along the stagnation line; right column: solutions on the surface.

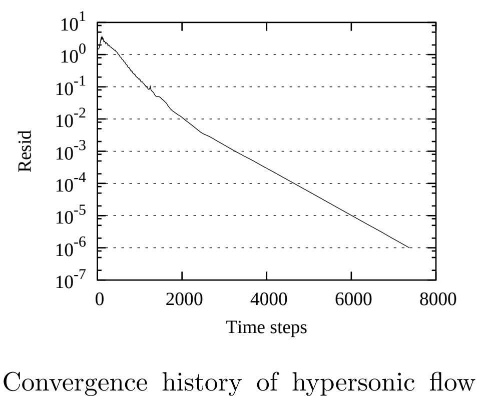


FIGURE 12. Convergence history of hypersonic flow (M = 15) around a 2D sphere-cone body.

Since the beauty of the DG method resides in its easy implementation of arbitrary spatial order of accuracy, it is desirable to extend the present limiting process to high order elements. Take the quadratic elements as an example, we might be able to limit the Hessian matrix which contains the constant second-order derivatives of the solutions in quadratic elements in a similar way to limiting the constant gradient vector in linear elements. As for non-comforming meshes encountered in h-typed mesh adaptation, we might need to perform the limiting on the elements of the same refinement level. All the extensions mentioned above are our ongoing work. Also, the extension to 3D tetrahedra meshes is of great interest too. We have gained some success in the extension of present limiting process to tetrahedra meshes and will report it in another paper.

References

[1] S. Aliabadi and T. Tezduyar, Space-time Finite Element Computation of Compressible Flows Involving Moving Boundaries and Interfaces, *Comput. Methods Appl. Mech. Eng.*, 107, (1993) 209-223.

[2] S. Aliabadi, J. Abedi, B. Zellars. and K. Bota, New Finite Element Technique for Simulation of Wave-Object Interaction, AIAA Paper 2002-0876.

[3] S. Aliabadi, A. Johnson, A. Abatan, J. Abedi, Y. Yeboah and K. Bota, Stabilized Finite Element Formulation of Buoyancy Driven Incompressible Flows, *Commun. Numer. Methods Eng. (CNME)*, 18, Issue 5, (2002) 315-324.

[4] S. Aliabadi and S. Tu, An Alternative To Slope Limiter In Discontinuous Galerkin Finite Element Method, AIAA paper 2004-0076.

[5] S. Aliabadi, *Parallel Finite Element Computations in Aerospace Applications*, PhD Thesis, The University of Minnesota, June 1994.

[6] F. Bassi and S. Rebay, A High-order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations, *J. Comput. Phys.*, 131 (1997), pp. 267-279.

[7] P. Batten, N. Clarke, C. Lambert and D. M. Causon, On the Choice of Wavespeeds for the HLLC Riemann Solver, *SIAM J. Sci. Comput.*, 18, No. 6, (1997) 1553-1570.

[8] A. Burbeau, P. Sagaut, and Ch.-H. Bruneau, A Problem-Independent Limiter for High-Order Runge-Kutta Discontinuous Galerkin Methods, *J. Comput. Phys.*, 169, (2001) 111-150.

[9] B. Cockburn and C.-W. Shu, The Runge-Kutta Discontinuous Galerkin Finite Element Method for Conservation Laws V: Multidimensional Systems, *J. Comput. Phys.*, 141, (1998) 199-224.

[10] S. F. Davis, Simplified Second-Order Godunov-type Methods, *SIAM J. Sci. Statist. Comput.*, 9, (1988) 445-473.

[11] M. Delanaye, *Polynomial Reconstruction Finite Volume Schemes for the Compressible Euler and Navier-Stokes Equations on Unstructured Adaptive Grids*, PhD thesis, Universite de Liege, 1996.

[12] D. Drikakis and S. Tsangaris, On the Accuracy and Efficiency of CFD Methods in Real Gas Hypersonics, *AGARD Conference Proceedings*, 514, (1993).

[13] B. Einfeldt, C. D. Munz, P. L. Roe and B. Sjogreen, On Godunov-type Methods Near Low Densities, *J. Comput. Phys.*, 92, (1991) 273-295.

[14] N. T. Frink, Assessment of an Unstructured grid Method for Predicting 3D Turbulent Viscous Flow, AIAA paper 96-0292.

[15] B. Grossman and W. Walters, Analysis of Flux-Split Algorithms for Euler's Equations with Real Gases, *AIAA J.*, 27, no. 5, (1989) 524-531.

[16] A. Harten, P. D. Lax and B. van Leer, On Upstream Differencing and Godunov-type Schemes for Hyperbolic Conservation Laws, *SIAM Rev.*, 25, (1983) 35-61.

[17] H. Hoteit, P. Ackerer, R. Mose, J. Erhel and B. Philippe, New Two-Dimensional Slope Limiters for Discontinuous Galerkin Methods on Arbitrary Meshes, Institut National de Recherche en Informatique et en Automatique (France), Rapport de Recherche, n 4491, July 2002.

[18] http://www.lcp.nrl.navy.mil/cfd-cta/CFD3.

[19] P. Jawahar and H. Kamath, A High-Resolution Procedure for Euler and Navier-Stokes Computations on Unstructured Grids, *J. Comput. Phys.*, 164, (2000) 165-203.

[20] J. T. Oden, I. Babuska and C. E. Baumann, A Discontinuous hp Finite Element Method for Diffusion Problems, *J. Comput. Phys.*, 146, (1998) 491-519.

[21] P. L. Roe, Some Contributions to the Modeling of Discontinuous Flows, *Lec. Notes Appl. Math.*, 22, (1985) 163-193.

[22] M. Sun and K. Takayama, Conservative Smoothing on an Adaptive Quadrilateral Grid, *J. Comput. Phys.*, 150, (1999) 143-180.

[23] E. F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics*, Springer, New York, 1999.

[24] G. Toth and D. Odstrcil, Comparison of some Flux Corrected Transport and Total Variation Diminishing Numerical Schemes for Hydrodynamic and Magnetohydrodynamic Problems, *J. Comput. Phy.*, 128:82, (1996).

[25] S. Tu and S. M. Ruffin, Solution Adaptive, Unstructured Cartesian-Grid Methodology for Chemically Reacting Flow, AIAA Paper 2002-3097.

[26] S. Tu and S. M. Ruffin, Calculation of Nonequilibrium Flows Using a Solution Adaptive, Unstructured Cartesian-Grid Solver, AIAA Paper 2002-3098.

[27] J. J. W. van der Vegt and H. van der Ven, Discontinuous Galerkin Finite Element Method with Anisotropic Local Grid Refinement for Inviscid Compressible Flows, *J. Comput Phys.*, 141, No. 1, (1998) 046-077.

[28] V. Venkatakrishnan, Convergence to Steady Solutions of the Euler Equations on Unstructured Grids with Limiters, *J. Comput. Phys.*, 118, (1995) 120-130.

[29] G. Yang, D. M. Causon, D.M. Ingram, R. Saunders and P. Batten, A Cartesian Cut Cell Method for Compressible Flows Part A: Static Body Problems, *The Aeronautical Journal*, pp.47-56, February 1997.

[30] D. L. De Zeeuw, *A Quadtree-Based Adaptively-Refined Cartesian-Grid Algorithm for Solution of the Euler Equations*, PhD thesis, The University of Michigan, 1993.

223 James P. Brawley Dr. SW, Clark Atlanta University, Department of Engineering, Atlanta, GA 30314, USA

E-mail: `stu@cau.edu and aliabadi@cau.edu`