EVALUATION OF DISCONTINUOUS GALERKIN AND SPECTRAL VOLUME MEDTHODS FOR 2D EULER EQUATIONS ON UNSTRUCTURED GRIDS

Yuzhi Sun[*] and Z.J. Wang[†]

Department of Mechanical Engineering
Michigan State University, East Lansing, MI 48824

ABSTRACT

The discontinuous Galerkin (DG) method and spectral volume (SV) method are two classes of robust and high-order accurate methods for hyperbolic conservation laws, capable of handling unstructured grids. In this paper, we evaluate their performance for the two-dimensional Euler equations in terms of accuracy, efficiency and memory requirement. We first review the basic features of the two methods, and compare their similarities and differences. Then we estimate the number of operations and storage requirement for each method. Finally both methods are used to solve a vortex propagation problem with an analytical solution to assess their accuracy and efficiency. They are also used to solve a double Mach reflection problem with both smooth features and discontinuities. Both the DG and SV methods are capable of achieving their formal order of accuracy while the DG method is slightly more accurate in terms of the error magnitude and takes more memory. The SV method appears to have a higher resolution for discontinuities because the data limiting can be done at the sub-element level.

1. INTRODUCTION

Problems of practical interest in which *convection* plays an important role arise in applications as diverse as meteorology, weather forecasting, oceanography, gas dynamics, aeroacoustics, turbo machinery, turbulent flows, transport of contaminant in porous media, semiconductor device simulation, and electro-magnetism, among many others. This is why devising robust, accurate, and efficient methods for numerically solving these problems is of considerable importance and, as expected, has attracted the interest of many researchers and practitioners. Many endeavors have been made to construct robust, accurate and efficient methods for convection-dominant problems. There have been significant progresses made in the last three decades.

Real world applications often are associated with complex geometries. Unstructured-grid based methods have shown tremendous promise in handling these geometries with relative ease. We therefore focus our attention on methods that can be applied on unstructured grids. During the last two decades, many successful high-order methods have been developed for unstructured grids, e.g., the spectral element method [13] or multi-domain spectral method [12], k-exact finite volume (FV) method [3,7], ENO/WENO method [1,9,11,15], the discontinuous Galerkin (DG) method [4-6], unstructured spectral method [10], fluctuation-splitting (FS) method [2], and recently the spectral volume (SV) method [17-19]. Among the higher-order accurate methods that are conservative, the DG and SV methods may be the most efficient.

The DG method is a finite element method using discontinuous solution and test spaces (usually piecewise polynomials of suitable degree), which means that the state variables usually are not continuous across element boundaries. The fluxes through the element boundaries are then computed using an approximate Riemann solver, mimicking the successful Godunov finite volume method [8]. Due to the use of Riemann fluxes across element boundaries, the DG method is fully conservative at the element level. The SV method [17-19] is ultimately a finite volume method. Each element (called a spectral volume) is partitioned into structured sub elements named control volumes (CVs). Mean state-variables at the CVs inside a SV are employed to construct a high-order polynomial, which is then utilized to update the means at the CVs. The reconstruction problem can be solved analytically, and is identical for all simplexes. Therefore a high-order SV method is much more efficient than a high-order k-exact FV, in which a reconstruction problem must be solved for each control volume. The SV method is fully conservative at the sub-cell control volume level.

A comparison of these two methods in terms of accuracy and CPU time has been made for scalar conservation laws in [16]. In this study, we further compare these two methods for the 2D Euler equations with both smooth and non-smooth problems. In the next

[*] Graduate Student (sunyuzhi@egr.msu.edu), AIAA Student Member
[†] Associate Professor (zjw@egr.msu.edu), AIAA Senior Member

two sections, we review the major features of these two methods for the 2D Euler equations. In Section 4, we estimate the number of operations and memory requirement for both methods. In Section 5, two numerical tests are presented. The numerical order of accuracy and CPU times are shown to verify the estimates. Both methods are also compared for their shock capturing abilities using a problem with both smooth features and discontinuities. Finally, several concluding remarks based on the current study are summarized in Section 6.

2. DISCONTINOUS GALERKIN METHOD

Consider the 2D Euler equations

$$\frac{\partial Q}{\partial t} + \nabla \cdot F = 0 \qquad (1)$$

equipped with proper initial and boundary conditions. In (1), Q is the vector of conserved variables, $F = (f, g)$, and f and g are the flux vectors in the x and y directions respectively

$$Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E \end{bmatrix}, \; f = \begin{bmatrix} \rho u \\ \rho uu + p \\ \rho uv \\ u(E+p) \end{bmatrix}, \; g = \begin{bmatrix} \rho v \\ \rho uv \\ \rho vv + p \\ v(E+p) \end{bmatrix}, \qquad (2)$$

where ρ is the density, u and v are the velocity components in x and y directions, p is the pressure, and E is the total energy. The pressure is related to the total energy by

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}(\rho u^2 + \rho v^2) \qquad (3)$$

with the ratio of specific heats γ being a constant.

By multiplying (1) by a scalar test function φ, integrating over the domain Ω, and performing an integration by parts we obtain the following weak statement for (1)

$$\int_{\Omega} \varphi Q_t dV + \oint_{\partial\Omega} \varphi F \bullet \boldsymbol{n} dS - \int_{\Omega} \nabla\varphi \bullet F dV = 0, \forall \varphi . \qquad (4)$$

The integrals in (4) are understood to be performed in a component-wise manner.

Space-Discretization

A discrete analogue of (4) can be obtained by subdividing Ω into N non-overlapping triangular elements $\{T_i\}$, and by applying (4) on each element. Assume the solution and test function be piece-wise polynomials in each element. Let the polynomial basis function be $\xi_j(x)$ (x is taken to represent the position vector from here on if there is no confusion). If the polynomial is of order k, then the dimension of the polynomial space in 2D is $n = (k+1)(k+2)/2$. The solution and the test function can be expressed as

$$Q_i(x,t) = \sum_{j=1}^{n} Q_i^j(t)\xi_j(x), \; \varphi_h = \sum_{j=1}^{n} \varphi_h^j \xi_j(x). \qquad (5)$$

The expansion coefficients Q_i^j denote the degrees of freedom (DOFs) of the numerical solution for element T_i. Note that there is no global continuity requirement for Q_i , which is generally discontinuous across the element boundaries. By splitting the integral over Ω in (4) into the sum of integrals over the elements and by replacing Q and φ with functions Q_i and φ_h , we obtain the following semi-discrete equation for element T_i

$$\frac{d}{dt}\int_{T_i} \varphi_h Q_i dV + \oint_{\partial T_i} \varphi_h F \bullet \boldsymbol{n} dS - \int_{T_i} \nabla\varphi_h \bullet F dV = 0. \qquad (6)$$

Equation (6) must be satisfied for any test function φ_h . Since ξ_j is the basis for φ_h , (6) is equivalent to the following system of n equations

$$\frac{d}{dt}\int_{T_i} \xi_j Q_i dV + \oint_{\partial T_i} \xi_j F \bullet \boldsymbol{n} dS - \int_{T_i} \nabla\xi_j \bullet F dV = 0.$$

$$1 \le j \le n \qquad (7)$$

Because the approximate solution is discontinuous at the element boundaries, the interface flux is not uniquely defined. It is at this stage the Riemann fluxes used in the Godunov finite volume method are borrowed. The interface flux function $F \bullet \boldsymbol{n}$ is replaced by a Riemann flux $\hat{F}(Q^L, Q^R, \boldsymbol{n})$, where Q^L and Q^R are the conserved variables at the left and right side of the element boundary. In order to guarantee consistency and conservation, the Riemann flux must satisfy

$$\hat{F}(Q,Q,\boldsymbol{n}) = F(Q) \bullet \boldsymbol{n}, \; \hat{F}(Q,R,\boldsymbol{n}) = -\hat{F}(R,Q,-\boldsymbol{n}) . \qquad (8)$$

The surface and volume integrals in (7) are computed with Gauss quadrature formulas of suitable orders of accuracy. For example, the surface integral must be exact for polynomials of degree *2k,* while the volume integral must be exact for polynomials of degree *2k-1*, i.e.,

$$\oint_{\partial T_i} \xi_j F \bullet \boldsymbol{n} dS = \sum_{r=1}^{K} \int_{A_r} \xi_j F \bullet \boldsymbol{n} dS.$$

$$\int_{A_r} \xi_j F \bullet \boldsymbol{n} dS \approx \sum_{s=1}^{ns} w_{rs} \xi_j(x_{rs}) \hat{F}(Q^L(x_{rs}), Q^R(x_{rs}), \boldsymbol{n}_r) A_r,$$

$$\int_{T_i} \nabla\xi_j \bullet F dV \approx \sum_{s=1}^{nv} w_s \nabla\xi_j(x_s) \bullet F(Q_i(x_s)) V_i. \qquad (9)$$

where K is the number of planar facets of T_i, ns is the number of quadrature points on a planar face for the surface integral, nv is the number of quadrature points in the element for the volume integral, w_{rs} and w_s are the Gauss quadrature weights, x_{rs} and x_s are the Gauss quadrature points. By assembling together all the elemental contributions, a system of ordinary differential equations that govern the evolution in terms of the discrete solution can be written as

$$\frac{dU}{dt} = R(U). \tag{10}$$

where U is the global vector of the degrees of freedom, and *R(U)* is the residual vector.

Time Integration

An explicit multi-stage third-order TVD (total variation diminishing) Runge-Kutta scheme is employed for time integration [14]. The Runge-Kutta scheme can be expressed in the following form:

$$\begin{aligned} U^{(1)} &= U^n + \Delta t R(U^n);\\ U^{(2)} &= \frac{3}{4}U^n + \frac{1}{4}[U^{(1)} + \Delta t R(U^{(1)})]; \\ U^{n+1} &= \frac{1}{3}U^n + \frac{2}{3}[U^{(2)} + \Delta t R(U^{(2)})]. \end{aligned} \tag{11}$$

Monotonicity Limiter

For the non-linear Euler equations, it is necessary to perform data limiting to maintain stability if the solution contains discontinuities. There are two different ways of applying limiters in the system setting. One way is to apply a limiter to each characteristic variable. The other is to apply a limiter to each component of the vector of the conservative variables. The former has the nice property of naturally degenerating to the scalar case if the hyperbolic system is linear, but the latter is much more efficient. In this paper, we choose the component-wise approach because of its efficiency. To this end, we first establish the following numerical monotonicity criterion for each element

$$\overline{Q}_i^{\min} \le Q_i(x_s) \le \overline{Q}_i^{\max}, \tag{12}$$

where $\overline{Q}_i^{\min}$ and $\overline{Q}_i^{\max}$ are the minimum and maximum cell-averaged solutions among all its neighboring elements sharing a face with T_i, and $Q_i(x_s)$ is the solution at any of the quadrature points. If (12) is strictly enforced, the resultant numerical scheme for the scalar case is TVD. If (12) is violated for any quadrature point, then it is assumed that the element is close to a discontinuity, and the solution in the element is locally linear, i.e.,

$$Q_i(x) = \overline{Q}_i + \nabla Q_i \bullet (\boldsymbol{x} - \boldsymbol{x}_i), \quad \forall \boldsymbol{x} \in T_i, \tag{13}$$

where $\boldsymbol{x}_i$ is the position vector of the centroid of T_i. The magnitude of the solution gradient is maximized subject to the monotonicity condition given in (12). The original high-order polynomial is used to compute an initial guess of the gradient, i.e.,

$$\nabla Q_i = \left(\frac{\partial Q_i}{\partial x}, \frac{\partial Q_i}{\partial y} \right)\Bigg|_{x_i}.$$

This gradient may not satisfy (12). Therefore it is limited by multiplying a scalar limiter $\varphi \in [0, 1]$ so that the following solution satisfies (12)

$$Q_i(x) = \overline{Q}_i + \varphi \nabla Q_i \bullet (\boldsymbol{x} - \boldsymbol{x}_i). \tag{14}$$

3. SPECTRAL VOLUME METHOD

In the SV method, the element T_i is named a *spectral volume*, which is further partitioned into subcells named *control volumes* (CVs), indicated by $C_{i,j}$, as shown in Figure 1. To represent the solution as a polynomial of degree *k* in two dimensions (2D), we need to partition the SV into *n = (k+1)(k+2)/2* sub-cells. The degrees of freedom (DOFs) in a SV are the volume-averaged mean variables $\overline{Q}_{i,j}$ at the *n* CVs. There are numerous ways of partitioning a SV, and not every partition is admissible in the sense that the partition may not be capable of producing a degree *k* polynomial. Once *n* mean solutions in the CVs of an admissible SV are given, a unique polynomial reconstruction can be obtained from

$$p_i(x) = \sum_{j=1}^{n} L_j(x)\overline{Q}_{i,j}, \tag{15}$$

where $L_j(x)$ are also degree *k* polynomials satisfying

$$\int_{C_{i,j}} L_m(x)dV = V_{i,j}\delta_{jm}, \tag{16}$$

and $V_{i,j}$ is the volume of $C_{i,j}$. This high-order polynomial reconstruction facilitates a high-order update for the mean solution of each CV. Integrating (1) in each *CV*, we obtain

$$\frac{d\overline{Q}_{i,j}}{dt}V_{i,j} + \sum_{r=1}^{K}\int_{S_r}(F \bullet \boldsymbol{n})dS = 0, \tag{17}$$

where *K* is the total number of faces in $C_{i,j}$. The flux integral in (17) is then replaced by a Gauss-quadrature formula that is exact for polynomials of degree *k*

$$\int_{S_r}(F \bullet \boldsymbol{n})dS \approx \sum_{s=1}^{ne} w_{rs}F(Q(x_{rs})) \bullet \boldsymbol{n}_r S_r, \tag{18}$$

where ne is the number of quadrature points on the *r-th* face, w_{rs} are the Gauss quadrature weights, x_{rs} are the Gauss quadrature points. Since the reconstructed polynomials are piece-wise continuous, the solution is usually discontinuous across the boundaries of a SV, although it is continuous across interior CV faces. The fluxes at the interior faces can be computed directly based on the reconstructed solutions at the quadrature points. The fluxes at the boundary faces of a SV are computed using approximate Riemann solvers given the left and right reconstructed solutions. We also use

Runge-kutta scheme expressed as (11) for time integration.

The TVD limiter in the SV method is applied for the sub-cells, rather than for the macro SVs. This is possible because of the availability of the sub-cell averages of the state variables. In order to make an objective comparison with the DG method, the limiters are implemented in a similar fashion.

4. NUMBER OF OPERATIONS AND MEMORY REQUIREMENT FOR DG AND SV

In order to provide a reasonable estimate of the number of operations for both methods on the two-dimensional Euler equations, we need to specify the approximate Riemann solver, which is the local Lax-Friedrichs solver, i.e.

$$\hat{F}(Q^L, Q^R, \mathbf{n}) = \tfrac{1}{2}\{[F(Q^L) + F(Q^R)] \bullet \mathbf{n} - \alpha(Q^R - Q^L)\}$$

where $\alpha = |\bar{v}_n| + \bar{c}$, $\bar{v}_n$ is the average normal velocity, and $\bar{c}$ the average speed of sound at the face. Since modern computers can execute multiplications as fast as additions, 1 operation is assumed to be one multiplication or one addition. Internal functions such as *sqrt* is assumed to cost 10 operations. Given the vector of conserved variables, it is estimated that an analytical flux evaluation costs M_a = 24 operations, and a Riemann computation takes M_R = 85 operations.

4.1 DG Method

We consider linear, quadratic and cubic elements, which yield second, third and fourth-order order spatial accuracy respectively. The DOFs for these elements are given in Figure 2. Over each element T_i , the numerical residual vector can be written as

$$R(T_i) = W^{-1}\begin{bmatrix} \int_{T_i}(F \bullet \nabla\xi_1)dV - \oint_{\partial T_i} F \bullet \boldsymbol{n}\xi_1 dS \\ \vdots \\ \int_{T_i}(F \bullet \nabla\xi_n)dV - \oint_{\partial T_i} F \bullet \boldsymbol{n}\xi_n dS \end{bmatrix}, \quad (19)$$

where W is the mass matrix. The total number of operations can be roughly divided into three main parts, corresponding to the cost for computing the conserved variables at all the Gauss quadrature points (N_1), the number of operations to compute the fluxes (N_2), and the cost to multiply the mass matrix (N_3).

There are a total of ($nv+3*ns$) quadrature points that are used for surface and volume integrals assuming that the element is a triangle. We need n multiplications and n-1 additions to compute one conserved variable given the DOFs. Since we have four conserved variables in total, therefore the total number of operations to compute the solutions at all the quadrature points is then

$$N_1 = 4*(2n-1)*(nv+3ns) .$$

To evaluate the volume integral, we need to compute the (analytical) fluxes at nv quadrature points relating to n shape functions, while $3*ns$ Riemann fluxes are necessary to evaluate the surface integral. However Riemann fluxes are shared between two neighboring elements. Therefore we need to halve the number of operations for the Riemann fluxes when evaluating the number of operations per element. We also include the number of operations to carry out the Gauss quadrature formula. Thus we obtain

$$N_2 = n*nv*M_a + 3*ns*M_R/2$$
$$+4*n*(2*nv-1)+4*n*(3*ns-1)$$

N_3 is simply the cost of a square matrix multiplying a vector, which is $n*n$ for one component. Since we have 4 components, N_3 is therefore $4*n*n$. The total cost to compute the residual vector for a single element is then

$$N_T = N_1 + N_2 + N_3$$

The numbers of operations for the DG schemes of second to fourth orders are listed in Table 1.

Table 1. Number of Operations for the DG Method

k	n	nv	ns	N_T
1	3	3	2	807
2	6	6	3	2507
3	10	12	4	6974

The memory requirement for the DG method is estimated as followed:

- Two solutions; one at current time step, and other at the last time step.
- Residual, transformation from a cell to a standard cell.
- Volume, centorid coordinates, face area, and face unit normal.
- Coordinates of quadrature points (face, cell)
- Gradient of shape function on quadrature points (face, cell).
- Value, and gradient of shape functions at the centorid of a cell.

The storage requirement is roughly 94 words per element for a second-order DG scheme, 226 words per element for a third-order DG scheme, and 519 words per element for a fourth-order DG scheme.

4.2 SV Method

The degrees of freedom in the SV method are the mean state variables at the sub-cell control volumes. Over each spectral volume T_i , the numerical residual can be expressed as

$$R(T_i) = \begin{bmatrix} -\oint_{\partial C_{i,1}} F \bullet \boldsymbol{n} dS \\ \vdots \\ -\oint_{\partial C_{i,n}} F \bullet \boldsymbol{n} dS \end{bmatrix}. \tag{20}$$

There are two kinds of faces in a spectral volume. The faces that lie on the SV boundaries are called *Riemann faces*, because the state variables are discontinuous across these faces. The other faces that lie inside a SV are named *continuous faces* because the state variables are continuous across these faces. Denote the total number of faces in a SV with *nf*, and the number of Riemann faces *nr*. Then the number of continuous faces is then (*nf* - *nr*). Let the number of quadrature points on each face (edge) be *ne*. Then the number of operations to compute the state variables at all the quadrature points is 4**nf***ne*(2n-1)*. In addition, a total of *(nf - nr)*ne* analytical fluxes need to be computed while *nr*ne* Riemann fluxes must be computed. Since the Riemann faces are shared between two neighboring SVs, the number of operations is again halved. We also include the number of operations to carry out the Gauss quadrature formula *(2*ne-1)*nf*4*. Since the mass matrix in the SV method is always the identity matrix, the total number of operations in the SV method to evaluate the residual can be written as

$$N_T = N_1 + N_2 + N_3$$

where

$$N_1 = 4 * nf * ne * (2n-1)$$

$$N_2 = (nf - nr) * ne * M_a + nr * ne * M_R / 2 + (2 * ne - 1) * nf * 4$$

$$N_3 = 0$$

The numbers of operations for the SV schemes of second to fourth orders are listed in Table 2.

Table 2. Number of Operations for the SV Method

k	*n*	*ne*	*nf*	*nr*	N_T
1	3	1	9	6	543
2	6	2	15	9	2553
3	10	2	27	12	6168

In the implementation of the SV method, the top priority has been to achieve the best efficiency. We therefore choose to store many geometric properties. The permanent memory requirement is estimated as follows:

- Two solutions, one at the latest time step, and the other at the last time step;
- The residuals, volumes and centroid coordinates for the CVs, the face unit normals and areas for the sub-cell grid;
- Face to cell and face to node connectivities for the sub-cell grid;
- Coordinates of the sub-cell grid;
- A connectivity linking each quadrature point on a face to a point of the local standard SV to reconstruct the solution at the quadrature point.

The storage requirement is roughly 99 words per element for a second-order SV scheme, 194 words per element for a third-order SV scheme, and 361 words per element for a fourth-order SV scheme. Note that the SV schemes take less memory than the DG schemes at 3rd and 4th orders of accuracy.

5. NUMERICAL TESTS

All of the computations were performed on a Pentium IV 2.0 GHz PC running the Redhat Linux 7.2 operating system. The code was written in C++, optimized and compiled with the default gcc compiler.

5.1 Vortex Propagation Problem

This is an idealized problem for the Euler equations in 2D, which was used by Shu [15]. The mean flow is $\{\rho, u, v, p\} = \{1, 1, 1, 1\}$. An isotropic vortex is then added to the mean flow, i.e., with perturbations in *u*, *v*, and temperature $T = p/\rho$, and no perturbation in entropy $S = p/\rho^\gamma$:

$$(\delta u, \delta v) = \frac{\varepsilon}{2\pi} e^{0.5(1-r^2)}(-\bar{y}, \bar{x}),$$

$$\delta T = -\frac{(\gamma-1)\varepsilon^2}{8\gamma\pi^2} e^{1-r^2},$$

$$\delta S = 0,$$

where $(\bar{x}, \bar{y}) = (x-5, y-5), r^2 = \bar{x}^2 + \bar{y}^2$, and the vortex strength $\varepsilon = 5$. In the numerical simulation, the computational domain is taken to be [0,10] x [0,10], with characteristic inflow and outflow boundary conditions imposed on the boundaries.

It can be readily verified that the Euler equations with the above initial conditions admit an exact solution that moves with the speed (1, 1) in the diagonal direction. Both the DG and SV methods were employed for this problem. The numerical simulation was carried out until t = 0.1 on two different grids, one regular and one irregular as shown in Figure 3.

The errors are computed based on the volume-averaged state variable on the element or the SV. Table 1 and Table 2 present the errors and recorded CPU times of both methods on the regular mesh, while Table 3 and Table 4 display the errors and CPU times on the irregular mesh. Note that both methods achieved the expected numerical order of accuracy. The DG schemes appear to have slightly smaller error magnitude than the corresponding SV schemes.

Based on the CPU times for regular mesh, we note that the DG method takes 43.47, 104.14, and 212.35 μs for linear, quadratic, and cubic elements respectively, while SV method spends 27.42, 99.88, and 237.69 for 2nd, 3rd, and 4th order schemes. The SV method is faster than the DG method at 2nd and 3rd order, but is slower at 4th order. We will further investigate the slight discrepancy from the estimates.

5.2 Double Mach Reflection

This problem is also a standard test case [20] for high-resolution schemes, and has been studied extensively by many researchers. The computational domain for this problem is chosen to be [0, 4] x [0, 1]. The reflecting wall lies at the bottom of the computational domain starting from x=1/6. Initially a right-moving Mach 10 shock is positioned at x=1/6, y=0 and makes a 60° angle with the x-axis. For the bottom boundary, the exact post-shock condition is imposed for the region from x=0 to x=1/6 and a solid wall boundary condition is used for the rest. For the top boundary of the computational domain, the solution is set to describe the exact motion of the Mach 10 shock. The left boundary is set at the exact post-shock condition, while the right boundary is set as an outflow boundary.

The numerical simulation was carried out until t = 0.2. A mesh refinement study was carried out on three different grids. The grids are generated from regular Cartesian meshes by subdividing each Cartesian cell into two triangles. The coarse grid has 25*100*2 triangles, the medium grid 50*188*2 triangles, and the fine grid consists of 120*480*2 triangles. The density contours with 30 equally spaced contour lines from $\rho = 1.528$ to $\rho = 20.863$ are shown in Fig. 4 and Fig. 5 for the second order DG scheme and SV scheme. Note that the "blown–up" region was also shown in those figures.

Note that the SV method has a higher resolution than the DG method for the shock, slip line and the other finer features near the triple point. The main reason is that the TVD limiter in the SV method is applied for the sub-cells, but the limiter in the DG method is applied for the elements (macro SVs).

6. CONCLUDING REMARKS

We have presented a comparison of the DG and SV methods for the 2D Euler equations. Similar to the 2D scalar conservation laws, the DG method has a lower error magnitude than the SV method at 3rd and 4th orders. The 2nd-order SV scheme is faster than the 2nd-order DG scheme. However, 3rd and 4th order SV schemes are quite similar to the corresponding DG schemes in terms of efficiency (<12 % in difference). It is also clear that the SV method has a higher resolution for discontinuities than the DG method because of the sub-cell average based data limiting. We also confirm that the SV method takes less memory and allows larger time steps than the DG method for the 2D Euler equations.

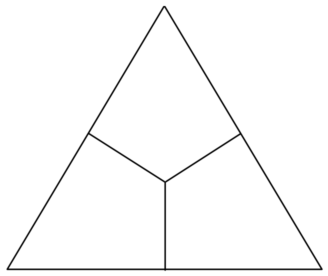
ACKNOWLEDGEMENT

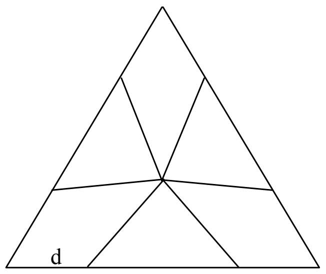
The authors gratefully acknowledge an Intramural Research Grant Project from Michigan State University, and a start-up funding provided by the Department of Mechanical Engineering, College of Engineering of Michigan State University.

REFERENCES

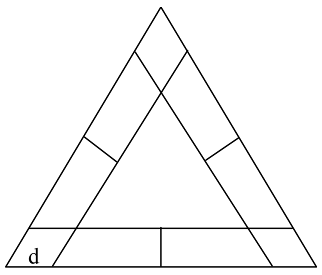
1. R. Abgrall, On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation, *J. Comput. Phys*. **114**, 45-58 (1994).
2. R. Abgrall and P.L. Roe, High order fluctuation splitting schemes on triangular mesh, submitted.
3. T.J. Barth and P.O. Frederickson, High-order solution of the Euler equations on unstructured grids using quadratic reconstruction," *AIAA Paper* No. 90-0013, 1990.
4. F. Bassi and S. Rebay, High-order accurate discontinuous finite element solution of the 2D Euler equations, *J. Comput. Phys*. **138**, 251-285 (1997).
5. B. Cockburn, S. - Hou and C. -W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case, *Mathematics of Computation* 54,545-581 (1990).
6. B. Cockburn and C.-W. Shu, The Runge-Kutta discontinuous Garlerkin method for conservation laws V: multidimensional systems, *J. Comput. Phys.*, **141**, 199 - 224, (1998**).**
7. M. Delanaye and Y. Liu, Quadratic reconstruction finite volume schemes on 3D arbitrary unstructured polyhedral grids, *AIAA* Paper No. 99-3259-CP, 1999.
8. S.K. Godunov, A finite-difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics, *Mat. Sb*. **47**, 271 (1959).
9. A. Harten, B. Engquist, S. Osher and S. Chakravarthy, Uniformly high order essentially non-oscillatory schemes III, *J. Comput. Phys.* **71**, 231 (1987).

10. J.S. Hesthaven and C.H. Teng, Stable spectral methods on tetrahedral elements, *SIAM J. Sci. Comput*. **21**, 2352-2380 (2000).
11. C. Hu and C.-W. Shu, Weighted essentially non-oscillatory schemes on triangular meshes, *J. Comput. Phys.* **150**, 97-127 (1999).
12. D.A. Kopriva, Multidomain spectral solutions of the Euler gas-dynamics equations, *J. Comput. Phys.* **96**, 428 (1991).
13. A.T. Patera, A Spectral element method for fluid dynamics: laminar flow in a channel expansion, *J. Comput. Phys.* **54** 468-488 (1984).
14. C.-W. Shu, Total-Variation-Diminishing time discretization, *SIAM Journal on Scientific and Statistical Computing* **9**, 1073-1084 (1988).
15. C.-W. Shu, Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, B. Cockburn, C. Johnson, C.-W. Shu and E.Tadmor, *Lecture Notes in Mathematics ,* volume 1697, Springer, 325-432 (1998).
16. Yuzhi Sun and Z.J. Wang, "Evaluation of Discontinuous Galerkin and Spectral Volume Methods for Conservation Laws on Unstructured Grids*," AIAA Paper* No. 2003-0253.
17. Z.J. Wang, Spectral (finite) volume method for conservation laws on unstructured grids: basic formulation, *J. Comput. Phys.*178, 210-251(2002).
18. Z.J. Wang and Yen Liu, Spectral (finite) volume method for conservation laws on unstructured grids II: extension to two-dimensional scalar equation, *J. Comput. Phys.* **179**, 665-697 (2002).
19. Z.J. Wang and Yen Liu, Spectral (finite) volume method for conservation laws on unstructured grids III: one-dimensional systems and partition optimization, J. of Scientific Computing, to appear.
20. P. Woodward and P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, *J. Comput. Phys.*, **54**, 115-173 (1984).


(a) Linear SV

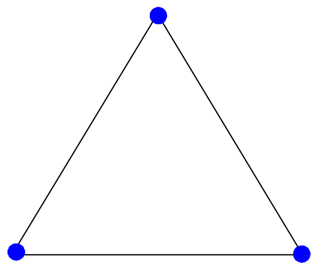


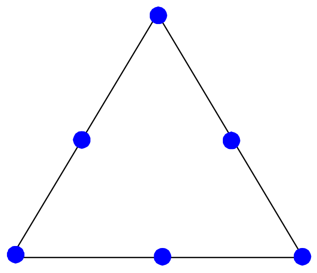
(b) Quadratic SV



(c) Cubic SV

Figure 1. SVs of various degrees


(a) Linear element


(b) Quadratic element

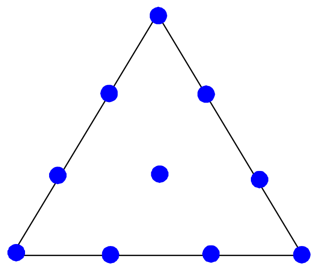
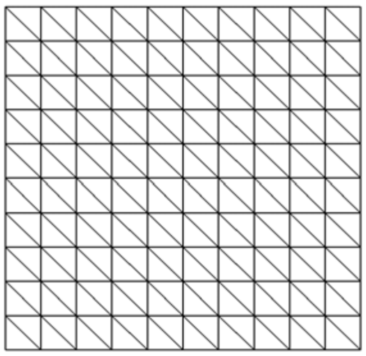
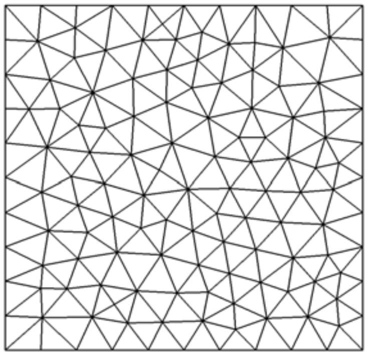

(c) Cubic element

Figure 2. The degrees of freedom in DG



(a) Regular (10x10x2)



(b) Irregular (10x10x2)

Figure 3. Regular and irregular grids for the vortex case

Table 1. Errors and CPU time on 2D Euler equations for vortex case at t = 0.1 using DG method (regular mesh)

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10x10x2	2.62e-03	--	2.63e-02	--	1.71e-01
	20x20x2	9.33e-04	1.49	1.19e-02	1.14	1.34e+00
	40x40x2	2.52e-04	1.89	3.41e-03	1.80	1.09e+01
	80x80x2	6.30e-05	2.00	8.75e-04	1.96	8.30e+01
	160x160x2	1.57e-05	2.00	2.20e-04	1.99	8.13e+02
3	10x10x2	3.75e-04	--	5.54e-03	--	4.05e-01
	20x20x2	6.67e-05	2.49	9.03e-04	2.62	3.14e+00
	40x40x2	1.02e-05	2.71	1.47e-04	2.62	2.47e+01
	80x80x2	1.72e-06	2.57	3.04e-05	2.27	2.03e+02
	160x160x2	2.78e-07	2.63	5.38e-06	2.50	2.06e+03
4	10x10x2	8.25e-05	--	6.51e-04	--	8.37e-01
	20x20x2	6.58e-06	3.65	8.17e-05	2.99	6.77e+00
	40x40x2	4.22e-07	3.96	5.27e-06	3.95	5.43e+01
	80x80x2	2.73e-08	3.95	3.56e-07	3.89	4.27e+02
	160x160x2	1.73e-09	3.98	1.93e-08	4.21	3.61e+03

Table 2. Errors and CPU time on 2D Euler equations for vortex case at t = 0.1 using SV method (regular mesh)

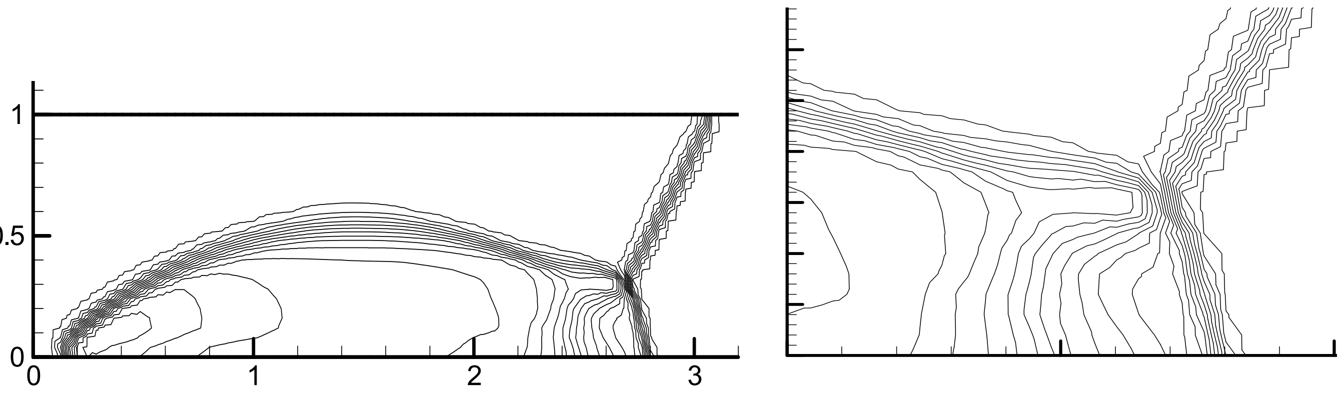
Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10x10x2	1.72e-03	--	1.52e-02	--	1.16e-01
	20x20x2	6.09e-04	1.50	6.28e-03	1.28	8.87e-01
	40x40x2	1.63e-04	1.90	1.79e-03	1.81	7.15e+00
	80x80x2	4.09e-05	1.99	4.66e-04	1.94	5.23e+01
	160x160x2	1.02e-05	2.00	1.19e-04	1.97	4.41e+02
3	10x10x2	3.79e-04	--	4.75e-03	--	4.37e-01
	20x20x2	7.59e-05	2.32	9.53e-04	2.32	3.19e+00
	40x40x2	1.34e-05	2.50	1.50e-04	2.67	2.42e+01
	80x80x2	2.25e-06	2.57	3.52e-05	2.09	1.90e+02
	160x160x2	3.73e-07	2.59	5.15e-06	2.77	1.69e+03
4	10x10x2	7.28e-05	--	6.09e-04	--	9.63e-01
	20x20x2	6.97e-06	3.38	1.22e-04	2.32	7.45e+00
	40x40x2	6.10e-07	3.51	1.39e-05	3.13	6.22e+01
	80x80x2	5.43e-08	3.49	1.17e-06	3.57	4.64e+02
	160x160x2	4.65e-09	3.55	8.05e-08	3.86	4.02e+03

Table 3. Errors and CPU time on 2D Euler equations for vortex case at t = 0.1 using DG method (irregular mesh)

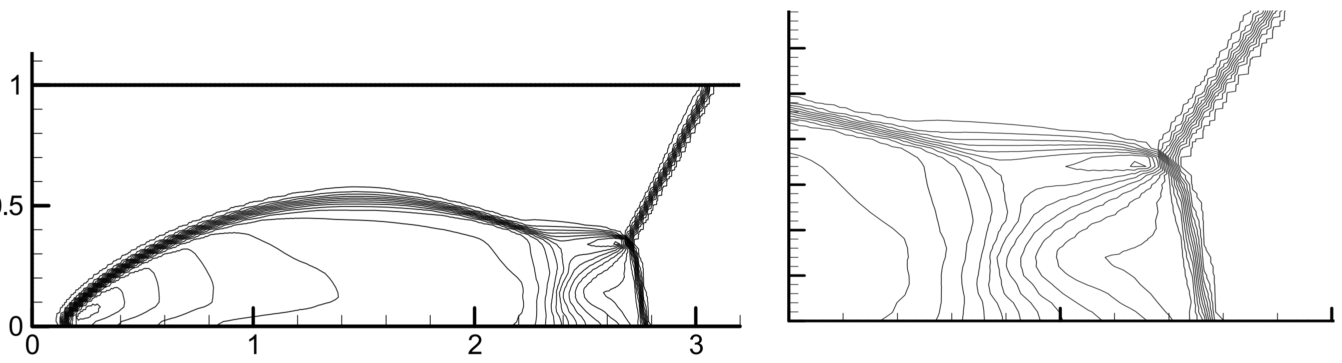
Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10x10	2.1938e-03	--	2.4772e-02	--	1.93e-01
	20x20	7.3981e-04	1.57	8.9395e-03	1.47	1.58e+00
	40x40	1.9561e-04	1.92	3.0051e-03	1.57	1.25e+01
	80x80	4.9088e-05	1.99	8.6666e-04	1.79	9.93e+01
	160x160	1.2246e-05	2.00	2.2906e-04	1.92	9.46e+02
3	10x10	2.7030e-04	--	4.2133e-03	--	4.56e-01
	20x20	4.5246e-05	2.58	7.2799e-04	2.53	3.76e+00
	40x40	7.3538e-06	2.62	1.1734e-04	2.63	2.91e+01
	80x80	1.2327e-06	2.58	2.5897e-05	2.18	2.41e+02
	160x160	1.8792e-07	2.71	4.4533e-06	2.54	2.25e+03
4	10x10	4.9035e-05	--	5.2184e-04	--	9.69e-01
	20x20	3.9102e-06	3.65	6.3063e-05	3.05	9.12e+00
	40x40	2.8644e-07	3.77	6.2253e-06	3.34	6.21e+01
	80x80	1.7762e-08	4.01	4.6664e-07	3.74	5.05e+02
	160x160	1.0828e-09	4.04	3.3610e-08	3.80	4.10e+03

Table 4. Errors and CPU time on 2D Euler equations for vortex case at t = 0.1 using SV method (irregular mesh)

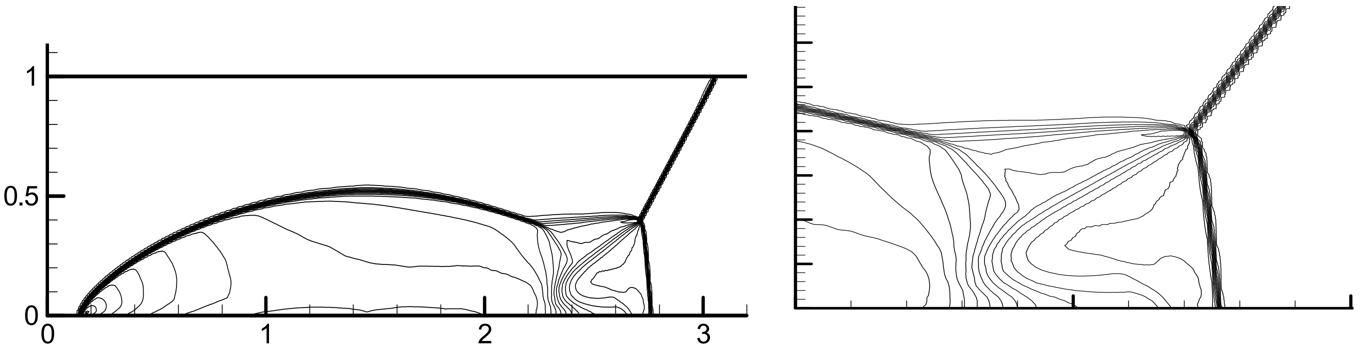
Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10x10	1.27e-03	--	1.43e-02		1.34e-01
	20x20	4.65e-04	1.45	5.68e-03	1.33	1.02e+00
	40x40	1.25e-04	1.90	1.45e-03	1.97	9.04e+00
	80x80	3.18e-05	1.97	5.36e-04	1.44	6.35e+01
	160x160	8.03e-06	1.99	1.68e-04	1.67	5.15e+02
3	10x10	2.68e-04	--	2.57e-03	--	4.97e-01
	20x20	4.97e-05	2.43	5.86e-04	2.13	3.69e+00
	40x40	8.80e-06	2.50	1.99e-04	1.56	2.86e+01
	80x80	1.56e-06	2.50	3.61e-05	2.46	2.21e+02
	160x160	2.67e-07	2.55	5.03e-06	2.84	2.03e+03
4	10x10	4.23e-05	--	5.07e-04	--	1.13e+00
	20x20	4.37e-06	3.27	8.69e-05	2.54	8.52e+00
	40x40	3.53e-07	3.63	1.31e-05	2.73	6.68e+01
	80x80	2.59e-08	3.77	9.30e-07	3.82	5.37e+02
	160x160	2.06e-09	3.65	9.63e-08	3.27	4.66e+03



(a) coarse grid

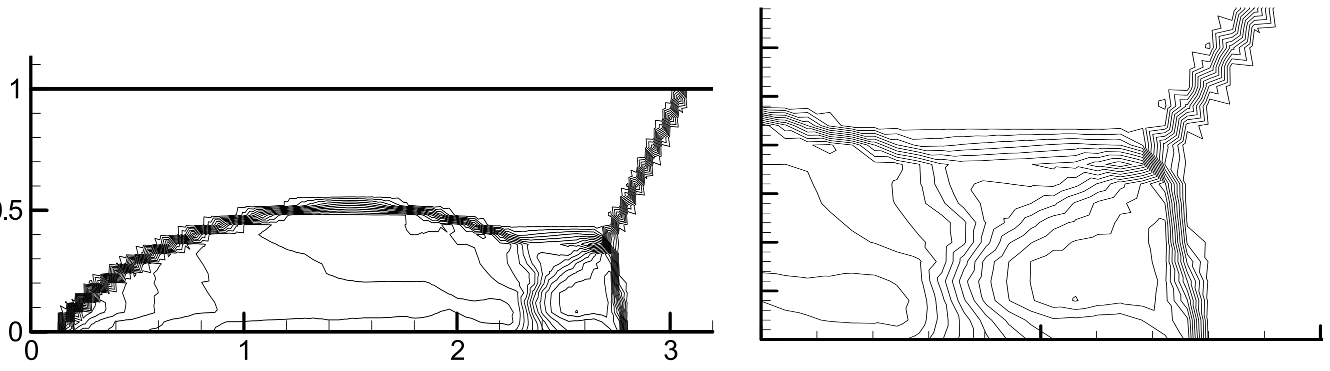


(b) medium grid

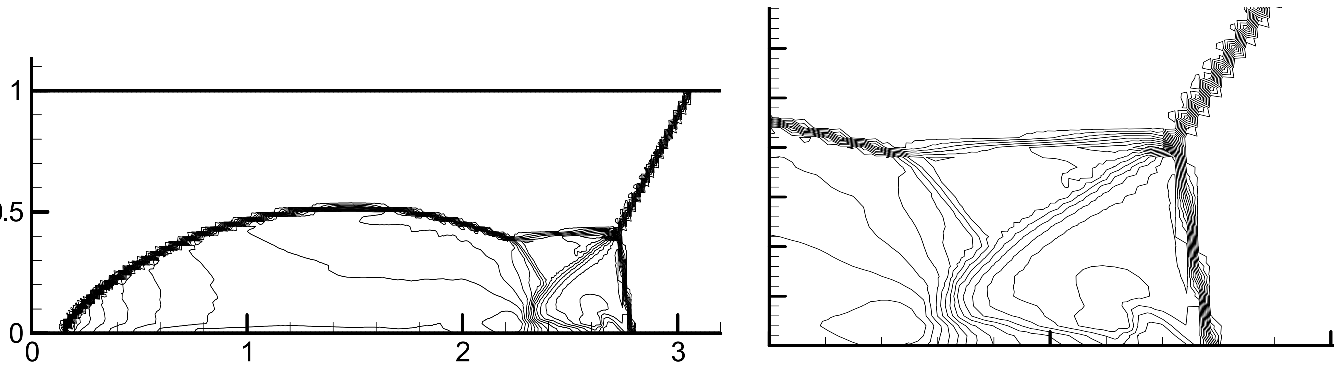


(c) fine grid

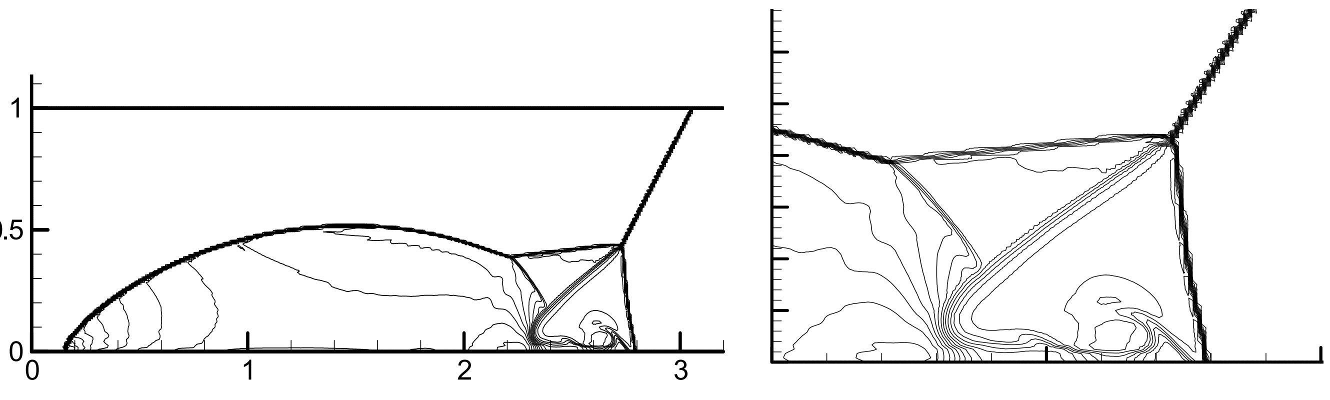
Figure 4: Density contours computed with second order DG scheme using a TVD limiter (30 equally spaced contour lines from $\rho = 1.528$ to $\rho = 20.863$).



(a) coarse grid



(a) medium grid



(c) fine grid

Figure 5. Density contours computed with second order SV scheme using a TVD limiter (30 equally spaced contour lines from $\rho = 1.528$ to $\rho = 20.863$).