INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **45**:819–838 (DOI: 10.1002/fld.726)

Evaluation of discontinuous Galerkin and spectral volume methods for scalar and system conservation laws on unstructured grids

Yuzhi Sun and Z. J. Wang*,†

Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824, U.S.A.

SUMMARY

The discontinuous Galerkin (DG) and spectral volume (SV) methods are two recently developed high-order methods for hyperbolic conservation laws capable of handling unstructured grids. In this paper, their overall performance in terms of efficiency, accuracy and memory requirement is evaluated using a 2D scalar conservation laws and the 2D Euler equations. To measure their accuracy, problems with analytical solutions are used. Both methods are also used to solve problems with strong discontinuities to test their ability in discontinuity capturing. Both the DG and SV methods are capable of achieving their formal order of accuracy while the DG method has a lower error magnitude and takes more memory. They are also similar in efficiency. The SV method appears to have a higher resolution for discontinuities because the data limiting can be done at the sub-element level. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: spectral finite volume; discontinuous Galerkin; unstructured grid; conservation laws

1. INTRODUCTION

It is well known that nature is governed by many conservation laws. The motion of fluids is one such phenomenon. The governing principles for fluids in motion are the conservation of mass, momentum and energy. Many other physical processes such as chemical reactions, combustion, explosions, and multi-phase flow problems can also be cast in conservation forms. Therefore, progresses made in computational methods for conservation laws can significantly impact the numerical simulation of numerous physical phenomena in nature. Real world applications often are associated with complex geometries. Unstructured-grid based methods

*Correspondence to: Z. J. Wang, Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824, U.S.A.
†E-mail: zjw@egr.msu.edu

Contract/grant sponsor: Intranural Research Grant Project
Contract/grant sponsor: Department of Mechanical Engineering

Published online 19 April 2004 | *Received 27 October 2003*
Copyright © 2004 John Wiley & Sons, Ltd. | *Revised 23 February 2004*

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 [1] or multi-domain spectral method [2], k-exact finite volume (FV) method [3, 4], essentially non-oscillatory (ENO) weighted ENO (WENO) method [5–8], the discontinuous Galerkin (DG) method [9–11], unstructured spectral method [12], fluctuation-splitting (FS) method [13], and recently the spectral volume (SV) method [14–17]. Since both the DG and SV methods are conservative at the element level, and can handle triangular unstructured grids and discontinuities, they will be the focus of this paper.

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 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 [18]. Due to the use of Riemann fluxes across element boundaries, the DG method is fully conservative at the element level. The SV method [14–17] is 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 within the element or SV, 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 method, 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.

In Reference [14], the SV method was shown to allow larger time steps than the DG method. On the other hand, the DG method was shown to have a lower error magnitude than the SV method [19] although both have the same order of accuracy. In this paper, the DG and SV methods are further evaluated in terms of accuracy, efficiency and memory requirement for both scalar and system conservation laws with both smooth and non-smooth problems [20, 21]. In the next two sections, the major features of these two methods for 2D conservation laws are reviewed. In Section 4, the number of operations and memory requirement for both methods are estimated. Section 5 presents several test cases. The numerical order of accuracy and CPU times are shown for both methods to verify the estimates. The methods are also compared for their shock capturing abilities using a problem with both smooth features and discontinuities. Finally, concluding remarks based on the current study are summarized in Section 6.

2. DISCONTINUOUS GALERKIN METHOD

Consider the following 2D conservation laws:

$$\frac{\partial Q}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0, \quad \mathbf{r} = (x, y) \in \Omega \tag{1}$$

where Q is the state variable, f and g are the fluxes in x and y directions, respectively. In (1), Q, f and g can be either scalars or column vectors, representing scalar or system conservation laws, respectively. For example, (1) represents the Euler equations if

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

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

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

with the ratio of specific heats γ being a constant. Let $\mathbf{F}=(f,g)$. Then (1) can be expressed in the following divergence form:

$$Q_t+\nabla\cdot\mathbf{F}=0 \tag{4}$$

The numerical solution of (4) is sought on the computational domain Ω subject to proper initial and boundary conditions.

Multiplying (4) by a scalar test function φ and integrating over the domain Ω, we obtain the following weighted residual formulation for Equation (4):

$$\begin{aligned}&\int_{\Omega}\varphi(Q_t+\nabla\cdot\mathbf{F})\,\mathrm{d}V\\ &=\int_{\Omega}\varphi Q_t\,\mathrm{d}V+\oint_{\partial\Omega}\varphi\mathbf{F}\cdot\boldsymbol{n}\,\mathrm{d}S-\int_{\Omega}\nabla\varphi\cdot\mathbf{F}\,\mathrm{d}V\\ &=0,\quad\forall\varphi\end{aligned} \tag{5}$$

Note that integrating by parts has been used in deriving (5), and that the integrals in (5) are understood to be performed in a component-wise manner if Q is a column vector.

2.1. Space-discretization

Assume that the computational domain Ω is subdivided into N non-overlapping triangular elements T_i. By applying (5) to each element T_i, we can obtain the discrete analogue of (5) on the computational grid. Let the solution and test function be piece-wise polynomials in each element. Denote the polynomial basis as $\boldsymbol{\xi}(\mathbf{r})=\{\xi_1(\mathbf{r}),\ldots,\xi_n(\mathbf{r})\}^{\mathrm{T}}$. If the polynomial is of order k, the dimension of the polynomial space in 2D is $n=(k+1)(k+2)/2$. The solution and the test function on element T_i can be expressed as

$$Q_i(\mathbf{r},t)=\sum_{j=1}^{n}Q_i^j(t)\xi_j(\mathbf{r}),\quad \varphi_h=\sum_{j=1}^{n}\varphi_h^j\xi_j(\mathbf{r}) \tag{6}$$

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. Using the solution and test function, (5) on element T_i becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{T_i}\varphi_h Q_i\,\mathrm{d}V+\oint_{\partial T_i}\varphi_h\mathbf{F}\cdot\boldsymbol{n}\,\mathrm{d}S-\int_{T_i}\nabla\varphi_h\cdot\mathbf{F}\,\mathrm{d}V=0 \tag{7}$$

Equation (7) must be satisfied for any test function φ_h. Since ξ_j is the basis function for φ_h, (7) is equivalent to the following system of n equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{T_i}\xi_j Q_i\,\mathrm{d}V+\oint_{\partial T_i}\xi_j\mathbf{F}\cdot\boldsymbol{n}\,\mathrm{d}S-\int_{T_i}\nabla\xi_j\cdot\mathbf{F}\,\mathrm{d}V=0,\quad 1\leqslant j\leqslant n \tag{8}$$

Because the approximate solution is discontinuous at the element boundaries, the interface flux is not uniquely defined. It is at this stage the Riemann flux used in the Godunov finite volume method [18] is borrowed. The interface flux function $\mathbf{F}\cdot\boldsymbol{n}$ is replaced by a Riemann flux $\hat{F}(Q^{\mathrm{L}},Q_{\mathrm{R}},\boldsymbol{n})$, where Q^{L} and Q^{R} are the state variables at the left and right side of the interface. In order to guarantee consistency and conservation, the Riemann flux must satisfy

$$\hat{F}(Q,Q,\boldsymbol{n})=\mathbf{F}(Q)\cdot\boldsymbol{n},\quad \hat{F}(Q^{\mathrm{L}},Q^{\mathrm{R}},\boldsymbol{n})=-\hat{F}(Q^{\mathrm{R}},Q^{\mathrm{L}},-\boldsymbol{n}) \tag{9}$$

The surface and volume integrals in (8) can be computed with Gauss quadrature formulas of suitable orders of accuracy. Following the arguments given in Reference [10], 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.

$$\begin{aligned}&\oint_{\partial T_i}\xi_j\mathbf{F}\cdot\boldsymbol{n}\,\mathrm{d}S=\sum_{r=1}^{K}\int_{A_r}\xi_j\mathbf{F}\cdot\boldsymbol{n}\,\mathrm{dS}\\&\int_{A_r}\xi_j\mathbf{F}\cdot\boldsymbol{n}\,\mathrm{d}S\approx\sum_{s=1}^{ns}w_{rs}\xi_j(\mathbf{r}_{rs})\hat{F}(Q^{\mathrm{L}}(\mathbf{r}_{rs}),Q^{\mathrm{R}}(\mathbf{r}_{rs}),\boldsymbol{n}_r)A_r\\&\int_{T_i}\nabla\xi_j\cdot\mathbf{F}\,\mathrm{d}V\approx\sum_{s=1}^{nv}w_s\nabla\xi_j(\mathbf{r}_s)\cdot\mathbf{F}(Q_i(\mathbf{r}_s))V_{i.}\end{aligned} \tag{10}$$

where K is the number of planar faces 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, $\mathbf{r}_{rs}$ and $\mathbf{r}_s$ are the Gauss quadrature points.

Let $U^i=\{Q_i^1,\ldots,Q_i^n\}^{\mathrm{T}}$ be the DOFs for element T_i., and W^i denote the mass matrix $\{\int_{T_i}\xi_j\xi_l\,\mathrm{d}V\}$. Equation (8) can be further written as

$$\frac{\mathrm{d}U^i}{\mathrm{d}t}+(W^i)^{-1}\left(\oint_{\partial T_i}\boldsymbol{\xi}\mathbf{F}\cdot\boldsymbol{n}\,\mathrm{d}S-\int_{T_i}\nabla\boldsymbol{\xi}\cdot\mathbf{F}\,\mathrm{d}V\right)=0 \tag{11}$$

By assembling together all the elemental contributions, a system of ordinary differential equations that govern the evolution of the discrete solution can be written as

$$\frac{\mathrm{d}U}{\mathrm{d}t}=R(U) \tag{12}$$

where U is the global vector of DOFs, and $R(U)$ is the global residual vector with the element vector being

$$R^i(U) = -(W^i)^{-1}\left(\oint_{\partial T_i} \xi \mathbf{F}\cdot \boldsymbol{n}\,\mathrm{d}S - \int_{T_i} \nabla\xi\cdot\mathbf{F}\,\mathrm{d}V\right) \tag{13}$$

2.2. Time integration

An explicit multi-stage third-order TVD (total variation diminishing) Runge–Kutta scheme is employed for time integration [22]. The Runge–Kutta scheme can be expressed in the following form:

$$\begin{aligned} U^{(1)} &= U^n + \Delta t R(U^n) \\ U^{(2)} &= \tfrac{3}{4}U^n + \tfrac{1}{4}[U^{(1)} + \Delta t R(U^{(1)})] \\ U^{n+1} &= \tfrac{1}{3}U^n + \tfrac{2}{3}[U^{(2)} + \Delta t R(U^{(2)})] \end{aligned} \tag{14}$$

2.3. 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 possible 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 of the conservative variables. In 1D, the former has the nice property of naturally degenerating to the scalar case if the hyperbolic system is linear. In multiple dimensions, characteristic variables are defined in a particular direction, e.g. in the face normal direction. In a fully unstructured grid, there is no co-ordinate direction to define a characteristic variable. Therefore, it is difficult to design characteristics-based limiters in multiple dimensions. In this paper, we choose the component-wise approach for the limiter, which should also be much more efficient than the characteristic approach. To this end, we first establish the following numerical monotonicity criterion for each element:

$$\overline{Q_i}^{\min} \leqslant Q_i(\mathbf{r}_s) \leqslant \overline{Q_i}^{\max} \tag{15}$$

where $\overline{Q_i}^{\min}$ and $\overline{Q_i}^{\max}$ are the minimum and maximum cell-averaged solutions among all its neighbouring elements sharing a face with T_i, and $Q_i(\mathbf{r}_s)$ is the solution at any of the quadrature points. If (15) is TVD. If (15) 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 forced locally linear, i.e.

$$Q_i(\mathbf{r}) = \overline{Q_i} + \nabla Q_i\cdot(\boldsymbol{r} - \boldsymbol{r}_i), \quad \forall \boldsymbol{r}\in T_i \tag{16}$$

where $\boldsymbol{r}_i$ 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 (15). The original polynomial is used to compute an initial guess of the gradient, i.e.

$$\nabla Q_i = \left(\frac{\partial Q_i}{\partial x}, \frac{\partial Q_i}{\partial y}\right)\bigg|_{r_i}$$

This gradient may not satisfy (15). Therefore, it is limited by multiplying a scalar limiter $\varphi \in [0,1]$ so that the following solution satisfies (15):

$$Q_i(x) = \overline{Q_i} + \varphi \nabla Q_i \cdot (\boldsymbol{r} - \boldsymbol{r}_i) \tag{17}$$

The scalar limiter can be obtained by examining the numerical solutions at all the quadrature points [15].

3. SPECTRAL VOLUME METHOD

In the SV method, the element T_i is named a *spectral volume*, which is further partitioned into subcells named CVs, indicated by $C_{i,j}$, as shown in Figure 1. To represent the solution as a polynomial of degree k in 2D, we need to partition the SV into $n = (k+1)(k+2)/2$ sub-cells. The 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

$$p_i(\mathbf{r}) = \sum_{j=1}^{n} L_j(\mathbf{r}) \overline{Q_{i,j}} \tag{18}$$

where $L_j(\mathbf{r})$ are also degree k polynomials satisfying

$$\int_{C_{i,j}} L_m(\mathbf{r}) \, \mathrm{d}V = V_{i,j} \delta_{jm} \tag{19}$$

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

$$\frac{\mathrm{d}\overline{Q_{i,j}}}{\mathrm{d}t} V_{i,j} + \sum_{r=1}^{K} \int_{A_r} (\mathbf{F} \cdot \boldsymbol{n}) \, \mathrm{d}S = 0 \tag{20}$$

where K is the total number of faces in $C_{i,j}$. The flux integral in (20) is then replaced by a Gauss-quadrature formula that is exact for polynomials of degree k

$$\int_{A_r} (\mathbf{F} \cdot \boldsymbol{n}) \, \mathrm{d}S \approx \sum_{s=1}^{ne} w_{rs} \mathbf{F}(Q(\mathbf{r}_{rs})) \cdot \boldsymbol{n}_r A_r \tag{21}$$

where ne is the number of quadrature points on the rth face, w_{rs} are the Gauss quadrature weights, $\mathbf{r}_{rs}$ are the Gauss quadrature points. Since the reconstructed polynomials are piece-wise continuous, the solution is 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 again computed using approximate Riemann solvers given the left and right reconstructed solutions. The Runge–Kutta scheme is again used for time integration.

The TVD limiter in the SV method [15] is very similar to the one described in the last section. The main difference is that the limiter is applied for the sub-cell averaged state

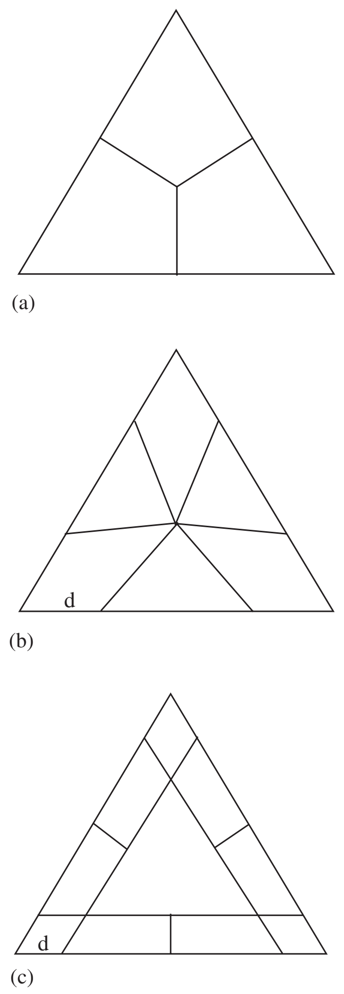


Figure 1. Spectral volumes of various degrees: (a) Linear SV; (b) quadratic SV; and (c) cubic SV.

variables, rather than for the averaged state variables of macro element, i.e. the SV. This is possible because of the inherent local resolution in the SV method. 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, we need to specify the governing equation and the Riemann solver. Two equations are considered in this paper. One is the 2D scalar linear conservation law in which Q is a scalar, and $f = aQ$ and $g = bQ$ with a and b being constants. The other equation is the 2D Euler equations. In both cases, the Rusanov flux [23] (also called local Lax–Friedrichs flux) is selected. The Rusanov flux for the scalar conservation laws takes the following form:

$$\hat{F}(Q^{\mathrm{L}}, Q^{\mathrm{R}}, \boldsymbol{n}) = \begin{cases} Q^{\mathrm{L}}(an_x + bn_y) & \text{if } (an_x + bn_y) > 0 \\ Q^{\mathrm{R}}(an_x + bn_y) & \text{otherwise} \end{cases} \tag{22}$$

Since modern computers can execute multiplications as fast as additions, one operation is defined to be one multiplication or one addition. Internal functions such as *sqrt* is assumed to cost 10 operations. In addition, each *if* statement is also counted as 1 operation. In this case, this scalar Riemann solver costs $M_{\mathrm{R}} = 5$ operations (3 operations to compute $an_x + bn_y$, one *if* statement, and another multiplication). The analytical flux takes $M_{\mathrm{a}} = 4$ operations.

For the Euler equations, the Rusanov Riemann flux [23] takes the following form:

$$\hat{F}(Q^{\mathrm{L}}, Q^{\mathrm{R}}, \mathbf{n}) = \tfrac{1}{2}\{[F(Q^{\mathrm{L}}) + F(Q^{\mathrm{R}})] \cdot \mathbf{n} - \alpha(Q^{\mathrm{R}} - Q^{\mathrm{L}})\} \tag{23}$$

where $\alpha = |\bar{v}_n| + \bar{c}$, $\bar{v}_n$ is the average face normal velocity, and $\bar{c}$ the average speed of sound at the interface. Given the vector of conservative variables, it is estimated that an analytical flux evaluation costs $M_{\mathrm{a}} = 24$ operations, and a Riemann flux takes $M_{\mathrm{R}} = 85$ operations.

For simplicity we have not considered the cost of limiters in the number of operations. We do believe that the limiters in the SV method are more expensive to implement than those in the DG method because data limiting is carried out for each element in the DG method, but for each subcell (CV) in the SV method.

4.1. DG method

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

$$R^i(U) = -(W^i)^{-1} \begin{bmatrix} \displaystyle\int_{T_i} (\mathbf{F} \cdot \nabla \xi_1)\,\mathrm{d}V - \oint_{\partial T_i} \hat{F}\xi_1\,\mathrm{d}S \\ \vdots \\ \displaystyle\int_{T_i} (\mathbf{F} \cdot \nabla \xi_n)\,\mathrm{d}V - \oint_{\partial T_i} \hat{F}\xi_n\,\mathrm{d}S \end{bmatrix} \tag{24}$$

The total number of operations can be roughly divided into three main parts, corresponding to the cost for computing the state 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. We need n multiplications and $n - 1$ additions to compute one state variable given the DOFs. Assume Q has nc component. Then the total number of operations to compute the solutions at all the quadrature points is then

$$N_1 = nc * (2 * n - 1) * (nv + 3 * ns) \tag{25}$$

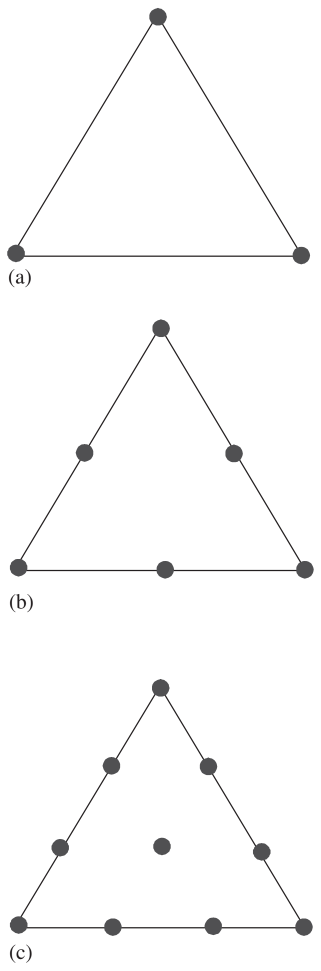


Figure 2. The degrees of freedom in the DG method: (a) Linear element; (b) quadratic element; and (c) cubic element.

Note that we have ignored the number of operations to compute the limiter for simplicity. 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 neighbouring elements. Therefore, we need to halve the number of operations for the Riemann fluxes when evaluating the number of operations per element. We also need to include the number of operations to carry out the Gauss quadrature formula. Thus we obtain

$$N_2 = n * nv * M_a + 3 * ns * M_R/2 + nc * n * (2 * nv - 1) + nc * n * (3 * ns - 1) \tag{26}$$

N_3 is simply the cost of a square matrix multiplying a vector, which is $2 * n * n - n$ for one component. For *nc* components, we therefore have $N_3 = nc * (2 * n * n - n)$. Note that $N_3 = 0$ if

Table I. Number of operations for the DG method.

Equation	k	N	nv	ns	N_T
Scalar conservation law	1	3	3	2	141
	2	6	6	3	512
	3	10	12	4	1496
Euler equations	1	3	3	2	831
	2	6	6	3	2627
	3	10	12	4	7334

an orthogonal basis is used. The total cost to compute the residual vector for a single element is then

$$N_T = N_1 + N_2 + N_3 \tag{27}$$

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

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

- Two solutions; one at the current time step, and the other at the last time step.
- Residual.
- Volume, centroid co-ordinates, face area, and face unit normal.
- Co-ordinates of quadrature points (face, cell).
- Gradient of shape function on quadrature points (face, cell).
- Shape functions, and their gradients at the centroid of the elements.

The storage requirement is roughly 90 words per element for a second-order DG scheme, 221 words per element for a third-order DG scheme, and 512 words per element for a fourth-order DG scheme for the 2D Euler equations.

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 residual can be expressed as

$$R^i(U) = \begin{bmatrix} -\oint_{\partial C_{i,1}} \hat{F}\,\mathrm{d}S \\ \vdots \\ -\oint_{\partial C_{i,n}} \hat{F}\,\mathrm{d}S \end{bmatrix} \tag{28}$$

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

Table II. Number of operations for the SV method.

Equation	k	N	ne	nf	nr	N_T
Scalar conservation law	1	3	1	9	6	81
	2	6	2	15	9	468
	3	10	2	27	12	1287
Euler equations	1	3	1	9	6	543
	2	6	2	15	9	2553
	3	10	2	27	12	6168

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 $nc * nf * ne * (2n - 1)$. 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 neighbouring SVs, the number of operations is again halved. We also include the number of operations to carry out the Gauss quadrature formula $(2 * ne - 1) * nf * nc$. Since the mass matrix in the SV method is always the identity matrix, number of operations in the SV method can be written as

$$N_T = N_1 + N_2 + N_3$$

where

$$\begin{gathered} N_1 = nc * nf * ne * (2n - 1) \\ N_2 = (nf - nr) * ne * M_a + nr * ne * M_R/2 + (2 * ne - 1) * nf * nc \\ N_3 = 0 \end{gathered} \tag{29}$$

The numbers of operations for the SV schemes of second to fourth orders are listed in Table II for both the scalar and system conservation laws. From Tables I and II, it is clear that the second-order SV scheme is significant faster than the second-order DG scheme, while the higher order schemes have similar computational costs.

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 residual, volumes and centroid co-ordinates for the CVs, the face unit normals and areas for the sub-cell grid.
- Face to cell and face to node connectivity for the sub-cell grid.
- Co-ordinates 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 for the 2D Euler equations 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 third and fourth 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. Scalar conservation laws

We first test the performance of both methods for the following linear scalar conservation law:

$$u_t + u_x + u_y = 0, \quad 0<x<2, \quad 0<y<2$$

with $u(x, y, 0) = \sin(\pi(x + y))$, and periodic boundary condition. The numerical simulation was carried out until $t = 1$ on two different grids, one regular and one irregular as shown in Figure 3. The finer meshes are produced recursively from the coarser meshes by dividing each triangle into four smaller triangles. The third-order TVD Runge–Kutta time integration scheme was used with a sufficiently small time step that the errors are independent of the time step. The same time step was used in both the DG and SV methods although larger time steps are permitted in the SV method for stability. The errors are computed based on the cell-averaged state variable on the element or the SV. No limiters were employed in the simulations since the problem is smooth. Tables III and IV present the errors and CPU times using both methods on the regular mesh, while Tables V and VI display the errors and CPU times using both methods on the irregular mesh. Note that on the regular mesh, both methods achieved the expected numerical order of accuracy in both the L_1 and L_∞ norms. However, the DG method consistently produced L_1 and L_∞ errors of smaller magnitude than the SV method. The SV method, on the other hand, is about 15–140% faster than the DG method depending on the order of accuracy of the scheme. On the irregular grid, the DG method is capable of achieving the expected order of accuracy in both the L_1 and L_∞ norms. Although the SV method achieved the expected order of accuracy in the L_1 norm, the third-order SV scheme showed a reduction of half an order in the L_∞ norm. This may indicate the quality of the quadratic SV partition can be further improved. Again, the SV method is consistently faster than the DG method on the irregular grid.

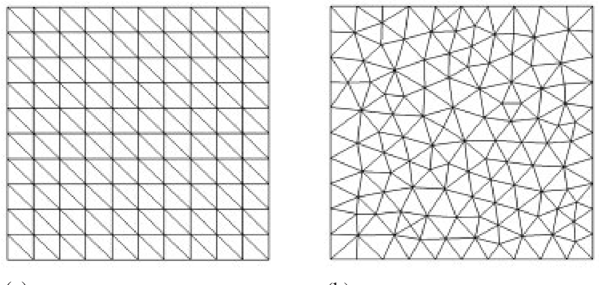

(a) (b)

Figure 3. Regular and irregular grids: (a) Regular (10×10×2); and (b) irregular (10 × 10 × 2).

Table III. Errors and CPU times at $t = 1$ for a 2D linear equation using the DG method on the regular mesh.

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU (s)
2	$10 \times 10 \times 2$	1.14e−02	—	2.43e−02	—	3.33e−01
	$20 \times 20 \times 2$	2.31e−03	2.30	5.83e−03	2.06	2.76e+00
	$40 \times 40 \times 2$	5.09e−04	2.19	1.42e−03	2.04	2.23e+01
	$80 \times 80 \times 2$	1.18e−04	2.11	3.49e−04	2.02	1.81e+02
	$160 \times 160 \times 2$	2.84e−05	2.06	8.65e−05	2.01	1.45e+03
3	$10 \times 10 \times 2$	3.45e−04	—	7.65e−04	—	7.590e−01
	$20 \times 20 \times 2$	4.27e−05	3.02	9.66e−05	2.99	6.37e+00
	$40 \times 40 \times 2$	5.32e−06	3.00	1.21e−05	3.00	5.09e+01
	$80 \times 80 \times 2$	6.65e−07	3.00	1.51e−06	3.00	4.27e+02
	$160 \times 160 \times 2$	8.31e−08	3.00	1.89e−07	3.00	3.37e+03
4	$10 \times 10 \times 2$	1.39e−05	—	2.43e−05	—	1.66e+00
	$20 \times 20 \times 2$	8.59e−07	4.02	1.52e−06	4.00	1.33e+01
	$40 \times 40 \times 2$	5.34e−08	4.01	9.54e−08	4.00	1.08e+02
	$80 \times 80 \times 2$	3.33e−09	4.00	5.97e−09	4.00	8.47e+02
	$160 \times 160 \times 2$	2.08e−10	4.00	3.73e−10	4.00	7.28e+03

Table IV. Errors and CPU times at $t = 1$ for a 2D linear equation using the SV method on the regular mesh.

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU (s)
2	$10 \times 10 \times 2$	4.02e−02	—	5.86e−02	—	1.21e−01
	$20 \times 20 \times 2$	1.06e−02	1.92	1.59e−02	1.88	9.47e−01
	$40 \times 40 \times 2$	2.71e−03	1.97	4.09e−03	1.96	8.81e+00
	$80 \times 80 \times 2$	6.83e−04	1.99	1.03e−03	1.99	8.39e+01
	$160 \times 160 \times 2$	1.71e−04	2.00	2.59e−04	1.99	6.05e+02
3	$10 \times 10 \times 2$	3.73e−03	—	5.21e−03	—	4.68e−01
	$20 \times 20 \times 2$	4.77e−04	2.97	7.12e−04	2.87	4.19e+00
	$40 \times 40 \times 2$	6.04e−05	2.98	9.05e−05	2.98	3.67e+01
	$80 \times 80 \times 2$	7.59e−06	2.99	1.14e−05	2.98	2.91e+02
	$160 \times 160 \times 2$	9.51e−07	3.00	1.43e−06	2.99	2.21e+03
4	$10 \times 10 \times 2$	5.90e−05	—	8.40e−05	—	1.32e+00
	$20 \times 20 \times 2$	3.73e−06	3.98	5.37e−06	3.97	1.35e+01
	$40 \times 40 \times 2$	2.35e−07	3.99	3.34e−07	4.01	8.61e+01
	$80 \times 80 \times 2$	1.48e−08	3.99	2.09e−08	4.00	6.90e+02
	$160 \times 160 \times 2$	9.24e−10	4.00	1.31e−09	4.00	5.72e+03

5.2. Vortex propagation problem

This is an idealized problem for the Euler equations in 2D, which was used by Shu [8]. 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

Table V. Errors and CPU times at $t = 1$ for a 2D linear equation using the DG method on the irregular mesh.

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10×10	2.17e−02	—	6.05e−02	—	3.77e−01
	20×20	4.67e−03	2.22	1.64e−02	1.88	3.26e+00
	40×40	1.07e−03	2.12	4.17e−03	1.97	2.65e+01
	80×80	2.56e−04	2.06	1.05e−03	1.99	2.13e+02
	160×160	6.26e−05	2.03	2.62e−04	2.00	1.75e+03
3	10×10	7.34e−04	—	2.56e−03	—	9.12e−01
	20×20	8.72e−05	3.07	4.42e−04	2.53	7.50e+00
	40×40	1.07e−05	3.03	6.34e−05	2.80	6.00e+01
	80×80	1.33e−06	3.01	8.19e−06	2.95	4.8e+02
	160×160	1.66e−07	3.00	1.03e−06	2.99	4.29e+03
4	10×10	4.10e−05	—	1.80e−04	—	1.93e+00
	20×20	2.41e−06	4.09	1.30e−05	3.79	1.57e+01
	40×40	1.47e−07	4.04	8.54e−07	3.92	1.27e+02
	80×80	9.08e−09	4.01	5.72e−08	3.90	1.02e+03
	160×160	5.65e−10	4.01	3.72e−09	3.94	8.87e+03

Table VI. Errors and CPU times at $t=1$ for a 2D linear equation using the SV method on the irregular mesh.

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10×10	6.71e−02	—	1.18e−01	—	1.38e−01
	20×20	1.83e−02	1.87	3.40e−02	1.80	1.30e+00
	40×40	4.71e−03	1.96	9.25e−03	1.88	1.17e+01
	80×80	1.19e−03	1.98	2.42e−03	1.94	8.61e+01
	160×160	3.00e−04	1.99	6.20e−04	1.96	8.38e+02
3	10×10	8.36e−03	—	1.68e−02	—	5.59e−01
	20×20	1.15e−03	2.86	2.95e−03	2.51	4.79e+00
	40×40	1.52e−04	2.92	5.28e−04	2.48	3.88e+01
	80×80	2.01e−05	2.91	1.31e−04	2.01	3.27e+02
	160×160	2.64e−06	2.93	2.85e−05	2.20	2.71e+03
4	10×10	2.28e−04	—	7.39e−04	—	1.53e+00
	20×20	1.37e−05	4.06	5.45e−05	3.76	1.35e+01
	40×40	8.50e−07	4.01	3.54e−06	3.94	1.03e+02
	80×80	5.33e−08	4.00	2.21e−07	4.00	7.98e+02
	160×160	3.35e−09	3.99	1.34e−08	4.04	7.31e+03

entropy $S = p/\rho^\gamma$:

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

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

$$\delta S = 0$$

Table VII. Errors and CPU times for the propagating vortex case at $t = 10$ using the DG method on the regular mesh.

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	$10 \times 10 \times 2$	7.74e−04	—	4.02e−03	—	8.91e+00
	$20 \times 20 \times 2$	1.05e−04	2.88	1.12e−03	1.84	7.40e+01
	$40 \times 40 \times 2$	1.52e−05	2.79	2.73e−04	2.04	6.12e+02
	$80 \times 80 \times 2$	2.39e−06	2.67	1.21e−04	1.17	4.70e+03
3	$10 \times 10 \times 2$	2.86e−04	—	2.22e−03	—	2.11e+01
	$20 \times 20 \times 2$	7.54e−05	1.92	9.98e−04	1.15	1.72e+02
	$40 \times 40 \times 2$	1.26e−05	2.58	1.68e−04	2.57	1.36e+03
	$80 \times 80 \times 2$	1.14e−06	3.47	2.60e−05	2.69	1.10e+04
4	$10 \times 10 \times 2$	1.20e−04	—	8.12e−04	—	4.45e+01
	$20 \times 20 \times 2$	6.60e−06	4.18	7.14e−05	3.51	3.65e+02
	$40 \times 40 \times 2$	1.47e−07	5.49	3.42e−06	4.38	2.89e+03
	$80 \times 80 \times 2$	3.90e−09	5.24	2.10e−07	4.03	2.31e+04

Table VIII. Errors and CPU time for the propagating vortex case at $t = 10$ using the SV method on the regular mesh.

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	$10 \times 10 \times 2$	9.50e−04	—	6.19e−03	—	3.77e+00
	$20 \times 20 \times 2$	1.92e−04	2.31	1.61e−03	1.94	2.96e+01
	$40 \times 40 \times 2$	4.14e−05	2.21	8.65e−04	0.90	2.38e+02
	$80 \times 80 \times 2$	9.92e−06	2.06	2.96e−04	1.55	1.91e+03
3	$10 \times 10 \times 2$	9.42e−04	—	5.49e−03	—	2.19e+01
	$20 \times 20 \times 2$	9.20e−05	3.36	9.91e−04	2.47	1.68e+02
	$40 \times 40 \times 2$	9.84e−06	3.22	2.45e−04	2.02	1.30e+03
	$80 \times 80 \times 2$	1.11e−06	3.15	3.56e−05	2.78	1.02e+04
4	$10 \times 10 \times 2$	1.82e−04	—	1.20e−03	—	5.05e+01
	$20 \times 20 \times 2$	1.01e−05	4.17	9.03e−05	3.73	3.91e+02
	$40 \times 40 \times 2$	4.90e−07	4.37	1.01e−05	3.16	3.35e+03
	$80 \times 80 \times 2$	3.16e−08	3.95	5.81e−07	4.12	2.46e+04

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 $[0, 10] \times [0, 10]$, 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 to simulate this problem. The numerical simulation was carried out until $t = 10$ on the two different grids shown in Figure 3.

The errors are computed based on the volume-averaged density on the element or the SV. Tables VII and VIII present the errors and recorded CPU times of both methods on the regular

Table IX. Errors and CPU times for propagating vortex case at $t = 10$ using the DG method on the irregular mesh.

Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10×10	6.29e−04	—	3.32e−03	—	1.02e+01
	20×20	1.15e−04	2.45	9.97e−04	1.74	8.80e+01
	40×40	2.78e−05	2.05	3.05e−04	1.71	7.07e+02
	80×80	4.20e−06	2.73	8.93e−05	1.77	5.62e+03
3	10×10	1.27e−04	—	8.88e−04	—	2.41e+01
	20×20	1.83e−05	2.79	3.26e−04	1.45	2.05e+02
	40×40	1.93e−06	3.25	9.86e−05	1.73	1.60e+03
	80×80	1.52e−07	3.67	1.61e−05	2.61	1.29e+04
4	10×10	4.23e−05	—	3.69e−04	—	5.12e+01
	20×20	2.56e−06	4.05	6.17e−05	2.58	4.94e+02
	40×40	9.80e−08	4.71	3.08e−06	4.32	3.38e+03
	80×80	2.67e−09	5.20	2.12e−07	3.86	2.69e+04

Table X. Errors and CPU times for the propagating vortex case at $t = 10$ using the SV method on the irregular mesh.

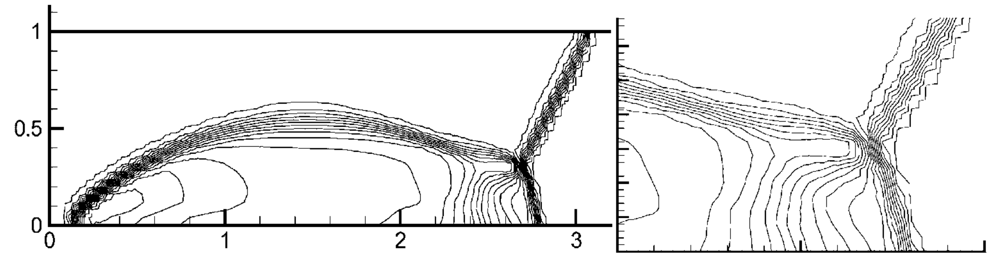
Order of accuracy	Grid	L_1 error	L_1 order	L_∞ error	L_∞ order	CPU
2	10×10	1.04e−03	—	5.45e−03	—	4.38e+00
	20×20	2.54e−04	2.03	1.92e−03	1.51	3.42e+01
	40×40	8.98e−05	1.50	8.72e−04	1.14	3.16e+02
	80×80	2.34e−05	1.94	2.90e−04	1.59	2.22e+03
3	10×10	5.06e−04	—	3.47e−03	—	2.53e+01
	20×20	7.44e−05	2.77	7.25e−04	2.26	1.93e+02
	40×40	9.75e−06	2.93	1.63e−04	2.15	1.51e+03
	80×80	1.51e−06	2.69	3.37e−05	2.27	1.19e+04
4	10×10	1.09e−04	—	5.57e−04	—	5.71e+01
	20×20	7.36e−06	3.89	8.76e−05	2.67	4.45e+02
	40×40	3.84e−07	4.26	5.75e−06	3.93	3.51e+03
	80×80	2.22e−08	4.11	4.00e−07	3.85	2.81e+04

mesh, while Tables IX and X display the errors and CPU times on the irregular mesh. Note that the DG method is more consistent in achieving the expected order of accuracy than the SV method on both the regular and irregular grids. The third and fourth-order DG schemes appear to have smaller error magnitude than the corresponding SV schemes.

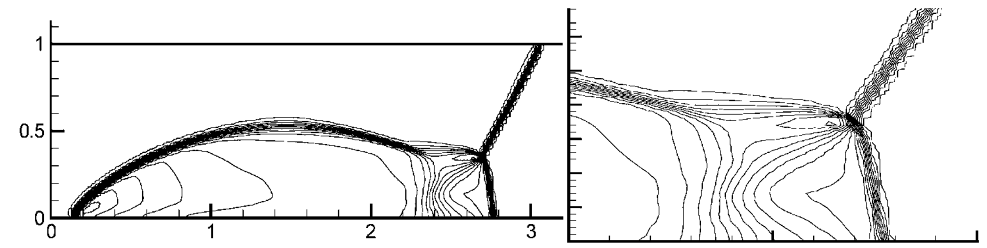
Based on the CPU times for the regular mesh, we note for per time step per element that the DG method takes 43.47, 104.14, and 212.35 μs for linear, quadratic, and cubic elements respectively, while SV method spends 27.42, 99.88, and 237.69 μs for second-, third-, and fourth-order schemes. The SV method is faster than the DG method at second and third order, but is slightly slower at fourth order.

5.3. Double mach reflection

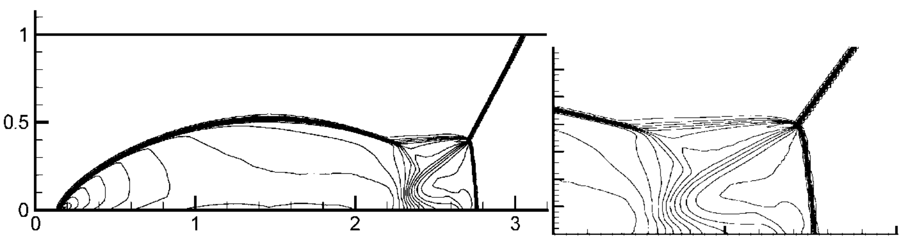
This problem is also a standard test case [24] for high-resolution schemes, and has been studied extensively by many researchers. The computational domain for this problem is chosen to be $[0,4]\times[0,1]$. The reflecting wall lies at the bottom of the computational domain starting from $x=\frac{1}{6}$. Initially a right-moving Mach 10 shock is positioned at $x=\frac{1}{6}$, $y=0$ 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 $\frac{1}{6}$ 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.



(a)

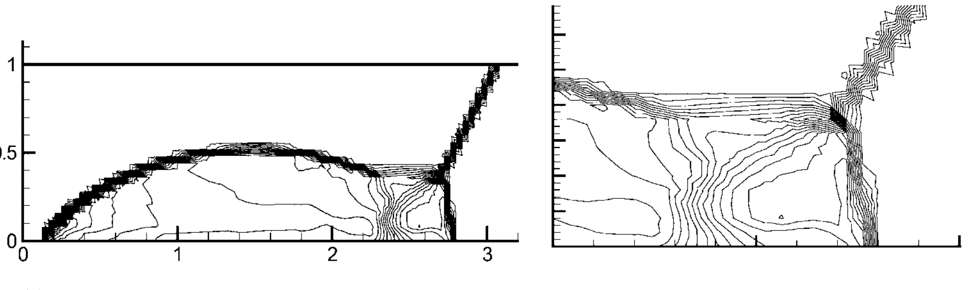


(b)

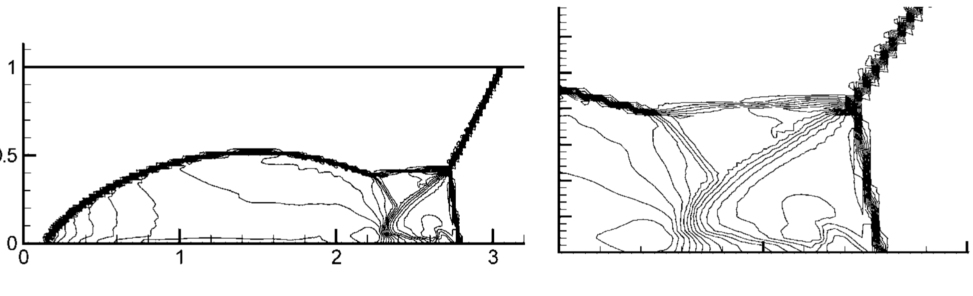


(c)

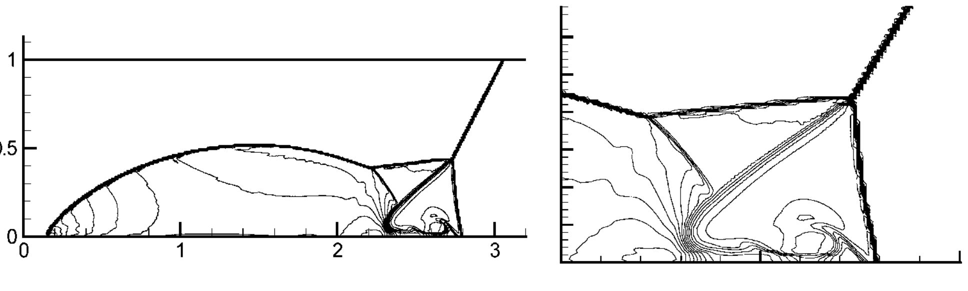
Figure 4. Density contours computed with the second-order DG scheme using a TVD limiter (30 equally spaced contour lines from $\rho = 1.528$ to 20.863): (a) coarse grid; (b) medium grid; and (c) fine grid.



(a)



(b)



(c)

Figure 5. Density contours computed with the second-order SV scheme using a TVD limiter (30 equally spaced contour lines from $\rho = 1.528$ to 20.863): (a) coarse grid; (b) medium grid; and (c) fine grid.

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 20.863 are shown in Figures 4 and 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 scalar conservation laws and Euler equations. Generally speaking, the DG method has a lower error magnitude than the SV method. In the scalar case, the SV schemes are consistently faster than the DG schemes of the same order of accuracy for each residual evaluation. For the Euler equations, the second-order SV scheme is faster than the second-order DG scheme. However, third- and fourth-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 both the 2D scalar conservation laws and Euler equations.

ACKNOWLEDGEMENTS

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. Patera AT. A Spectral element method for fluid dynamics: laminar flow in a channel expansion. *Journal of Computational Physics* 1984; **54**:468–488.
2. Kopriva DA. Multidomain spectral solutions of the Euler gas-dynamics equations. *Journal of Computational Physics* 1991; **96**:428.
3. Barth TJ, Frederickson PO. High-order solution of the Euler equations on unstructured grids using quadratic reconstruction. *AIAA Paper No*. 90-0013, 1990.
4. Delanaye M, Liu Y. Quadratic reconstruction finite volume schemes on 3D arbitrary unstructured polyhedral grids. *AIAA Paper No*. 99-3259-CP, 1999.
5. Abgrall R. On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation. *Journal of Computational Physics* 1994; **114**:45–58.
6. Harten A, Engquist B, Osher S, Chakravarthy S. Uniformly high order essentially non-oscillatory schemes III. *Journal of Computational Physics* 1987; **71**:231.
7. Hu C, Shu CW. Weighted essentially non-oscillatory schemes on triangular meshes. *Journal of Computational Physics* 1999; **150**:97–127.
8. Shu CW. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, Cockburn B, Johnson C, Shu CW, Tadmor E (eds). *Lecture Notes in Mathematics*, vol. 1697. Springer: Berlin 1998; 325–432.
9. Bassi F, Rebay S. High-order accurate discontinuous finite element solution of the 2D Euler equations. *Journal of Computational Physics* 1997; **138**:251–285.
10. Cockburn B, Hou S, Shu CW. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case. *Mathematics of Computation* 1990; **54**:545–581.
11. Cockburn B, Shu CW. The Runge–Kutta discontinuous Garlerkin method for conservation laws V: multidimensional systems. *Journal of Computational Physics* 1998; **141**:199–224.
12. Hesthaven JS, Teng CH. Stable spectral methods on tetrahedral elements. *SIAM Journal of Scientific Computing* 2000; **21**:2352–2380.

13. Abgrall R, Roe PL. High order fluctuation splitting schemes on triangular mesh. *Journal of Scientific Computing* 2003; **19**:3–36.
14. Wang ZJ. Spectral (finite) volume method for conservation laws on unstructured grids: basic formulation. *Journal of Computational Physics* 2002; **178**:210–251.
15. Wang ZJ, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids II: extension to two-dimensional scalar equation. *Journal of Computational Physics* 2002; **179**:665–697.
16. Wang ZJ, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids III: one-dimensional systems and partition optimization. *Journal of Scientific Computing* 2004; **20**:137–157.
17. Wang ZJ, Zhang L, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids IV: extension to two-dimensional Euler equations. *Journal of Computational Physics* 2004; **194**(2):716–741.
18. Godunov SK. A finite-difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. *Matematicheskii Sbornik* 1959; **47**:271.
19. Zhang M, Shu CW. An analysis of and a comparison between the discontinuous Galerkin and the spectral finite volume methods. *Computer and Fluids* (in press).
20. Sun Y, Wang ZJ. Evaluation of discontinuous Galerkin and spectral volume methods for conservation laws on unstructured grids. *AIAA Paper No*. 2003-0253.
21. Sun Y, Wang ZJ. Evaluation of discontinuous Galerkin and spectral volume methods for 2d Euler equations on unstructured grids. *AIAA Paper No*. 2003-3680.
22. Shu CW. Total-variation-diminishing time discretization. *SIAM Journal on Scientific and Statistical Computing* 1988; **9**:1073–1084.
23. Rusanov VV. Calculation of interaction of non-steady shock waves with obstacles. *Journal of Computational Mathematics and Physics USSR* 1961; **1**:267–279.
24. Woodward P, Colella P. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics* 1984; **54**:115–173.