JOURNAL OF COMPUTATIONAL PHYSICS **141,** 199–224 (1998)
ARTICLE NO. CP985892

The Runge–Kutta Discontinuous Galerkin Method for Conservation Laws V

Multidimensional Systems

Bernardo Cockburn*,[1] and Chi-Wang Shu†,[2]

**School of Mathematics, University of Minnesota, Minneapolis, Minnesota 55455;*
†*Division of Applied Mathematics, Brown University, Providence, Rhode Island 02912*
E-mail: cockburn@math.umn.edu, shu@cfm.brown.edu

Received May 30, 1997; revised December 9, 1997

This is the fifth paper in a series in which we construct and study the so-called Runge–Kutta discontinuous Galerkin method for numerically solving hyperbolic conservation laws. In this paper, we extend the method to multidimensional nonlinear systems of conservation laws. The algorithms are described and discussed, including algorithm formulation and practical implementation issues such as the numerical fluxes, quadrature rules, degrees of freedom, and the slope limiters, both in the triangular and the rectangular element cases. Numerical experiments for two-dimensional Euler equations of compressible gas dynamics are presented that show the effect of the (formal) order of accuracy and the use of triangles or rectangles on the quality of the approximation.

Key Words: discontinuous Galerkin; slope limiters; Euler equations.

1. INTRODUCTION

This is the fifth article of a series [13–16] devoted to the construction and study of the so-called Runge–Kutta discontinuous Galerkin (RKDG) method. The RKDG method is a method devised to numerically solve the initial boundary value problem associated with the

[1] Research partially supported by NSF Grant DMS-9407952 and by the University of Minnesota Supercomputing Institute.

[2] Research supported by ARO Grant DAAH04-94-G-0205, NSF Grant DMS-9500814, NASA Langley Grant NAG-1-1145, and Contract NAS1-19480 while this author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23681-0001, and AFOSR Grant 95-1-0074.

0021-9991/98 $25.00

conservation law

$$\partial_t u + \operatorname{div} \mathbf{f}(u) = 0 \quad \text{in } \Omega \times (0, T), \tag{1.1}$$

where $\Omega \subset R^d$ and $u = (u_1, \ldots, u_m)^t$, which is assumed to be hyperbolic; that is, $\mathbf{f}(u)$ is assumed to be such that any real combination of the Jacobians $\sum_{i=1}^{d} \xi_i (\partial f_i / \partial u)$ has m real eigenvalues and a complete set of eigenvectors. In this paper, we continue our work in [13–16] and extend (and improve) the RKDG method to the case of multidimensional systems. To place this paper under a proper perspective, we first discuss the work done in this series of papers and papers by other authors which has been prompted by the remarkable compactness and parallelizability of the RKDG method and by its ability to easily handle boundary conditions and complicated geometry.

The original discontinuous Galerkin method was introduced by Reed and Hill [31], and analyzed by LeSaint and Raviart [26], Johnson and Pitkaränta [25], Richter [32], and by Peterson [29]. All these were for the linear equations. Our work was concentrated on treating nonlinear equations, which call for different techniques. The first (one-dimensional) RKDG method was introduced in [13] by combining the piecewise-linear discontinuous finite elements used for the space discretization of one-dimensional conservation laws by Chavent and Cockburn [11] with one of the explicit, TVD time discretizations developed by Shu [34], and Shu and Osher [35, 36]. The resulting scheme was shown to be formally uniformly second-order accurate (a fact confirmed by numerical experiments) and was proven to be total variation diminishing in the means (TVDM). Later, in [14], the RKDG schemes were defined using a general framework that allowed piecewise polynomials of degree $k \in N$ approximate solutions. These fully explicit schemes were proven to be TVBM (total variation bounded in the means) and were shown to be formally uniformly $(k+1)$th order accurate, facts that were both verified numerically. The extension of the RKDG schemes to one-dimensional systems was carried out in [15] and the multidimensional case for the scalar conservation law was treated in [16], where it was proven that for some fairly general triangulations, the approximate solution given by the RKDG method satisfies a local maximum principle independently of the degree k. A projection, or generalized "slope limiting," was constructed which enforced the above maximum principle without destroying the formal accuracy of the method. Theoretical indications that the method is uniformly $(k+1)$th order-accurate when polynomials of degree k are used were given and numerical validation of this claim was presented for piecewise-linear approximations $k = 1$ in uniform grids made of triangles. The case $k = 2$ was worked out by Hou [23].

To define the RKDG method for multidimensional systems, only the generalized "slope limiter" of the method requires a nontrivial extension from the scalar case treated in [16]; everything else remains the same or can be trivially extended. The extension of the RKDG method to the two-dimensional Euler equations of gas dynamics was carried out in[17], where piecewise-linear approximations were used. In this paper, we complete and improve the work started in [17]. The main contribution of this paper is thus the devising of a *practical* generalized "slope limiting" procedure for multidimensional systems. The construction of this procedure, which is essential for nonsteady-state problems, is inspired by the theoretically proven, "slope limiting" devised in [16], but is remarkably simpler and gives better numerical results.

In related work, Atkins and Shu [1] studied an alternative quadrature-free implementation of the RKDG method. Bey and Oden [8] used the RKDG method with arbitrary

quadrilaterals and piecewise-linear approximate solutions, to solve 2D Euler equations. Jiang and Shu [24] proved a cell entropy inequality for the square entropy for arbitrary order of accuracy and arbitrary triangulations, without using the nonlinear limiters for the semidiscrete (continuous in time) case. This also implied the L^2 stability of the method for nonlinear shocked cases. Lowrie, Roe, and van Leer [27] studied the discontinuous Galerkin method in space and time; see also the related studies previously made by Bar-Yoseph [2] and Bar-Yoseph and Elata [3].

The important issue of the parallelizability of the RKDG method has been explored by several authors. Biswas, Devine, and Flaherty [9] have shown that the RKDG method (with a new, interesting limiting) has a "solution parallel efficiency" of 99% in the NCUBE/2—a reflection of the fact that the RKDG method uses only the information of immediate neighbors to march in time. These authors have also constructed *h*- and *p*-adaptive versions of the RKDG method with remarkable results; see also the application to the Euler equation of gas dynamics by deCougny *et al.* [19]. The important issue of "dynamic load balancing," essential for adaptive methods, has been addressed by Devine *et al.* [21], by Özturan *et al.* [28], and by Devine *et al.* [20].

The effect of the quality of the approximation of curved boundaries on the quality of the approximate solution has been explored in a recent paper by Bassi and Rebay [4]; in this paper, we only consider computational domains with Lipschitz boundaries.

Extensions of the method to the compressible Navier Stokes equations and general convection diffusion equations can be found in Bassi and Rebay [5] and Cockburn and Shu [18], respectively.

We are now ready to give a detailed description of the contents of this paper. In Section 2, we give a general formulation of the RKDG method for multidimensional systems, including the discussion on slope limiters. Section 3 contains the algorithm and implementation details, including the numerical fluxes, quadrature rules, degrees of freedom, and slope limiters of the RKDG method for both piecewise-linear and piecewise-quadratic approximations in both triangular and rectangular elements. In Section 4, we present several test problems for the two-dimensional Euler equations of gas dynamics intended to illustrate the effect of the degree k and the effect of the use of triangles or rectangles on the accuracy of the method. Concluding remarks are given in Section 5.

2. ALGORITHM FORMULATION

To define the RKDG method, we proceed as in [16].

2.1. *Space Discretization*

First, we discretize (1.1) in space using the discontinuous Galerkin method. For each time $t \in [0, T]$, the approximate solution $u_h(t)$ is sought in the finite element space of **discontinuous** functions

$$V_h = \{v_h \in L^\infty(\Omega) : v_h|_K \in V(K), \forall K \in \mathcal{T}_h\}, \tag{2.1}$$

where $\mathcal{T}_h$ is a triangulation of the domain Ω and $V(K)$ is the so-called local space. In this paper, $V(K)$ is taken to be P^k, the collection of polynomials of degree k, for $k = 1$ and 2.

To determine the approximate solution $u_h(t)$, we need the weak formulation of (1.1):

$$\frac{d}{dt}\int_K u(x,t)\,v(x)\,dx+\sum_{e\in\partial K}\int_e \mathbf{f}(u(x,t))\cdot n_{e,K}\,v(x)\,d\Gamma-\int_K \mathbf{f}(u(x,t))\cdot\operatorname{grad} v(x)\,dx=0,$$

for any smooth function $v(x)$. Here $n_{e,K}$ denotes the outward unit normal to the edge e.

We replace the integrals by quadrature rules as

$$\int_e \mathbf{f}(u(x,t))\cdot n_{e,K}\,v_h(x)\,d\Gamma\approx\sum_{l=1}^{L}\omega_l\mathbf{f}(u(x_{el},t))\cdot n_{e,K}\,v(x_{el})|e|, \tag{2.2}$$

$$\int_K \mathbf{f}(u(x,t))\cdot\operatorname{grad} v(x)\,dx\approx\sum_{j=1}^{M}\underline{\omega}_j\mathbf{f}(u(x_{Kj},t))\cdot\operatorname{grad} v(x_{Kj})|K|. \tag{2.3}$$

Then, the flux $\mathbf{f}(u(x,t))\cdot n_{e,K}$ is replaced by the *numerical* flux $h_{e,K}(x,t)$, the exact solution u is replaced by the approximate solution u_h, and the test function v by $v_h\in V(K)$, resulting in the scheme:

$$\begin{aligned}&u_h(t=0)=P_{V_h}(u_0),\\ &\frac{d}{dt}\int_K u_h(x,t)v_h(x)\,dx+\sum_{e\in\partial K}\sum_{l=1}^{L}\omega_l\,h_{e,K}(x_{el},t)v(x_{el})|e|\\ &\quad-\sum_{j=1}^{M}\underline{\omega}_j\mathbf{f}(u_h(x_{Kj},t))\cdot\operatorname{grad} v_h(x_{Kj})|K|=0\quad\forall v_h\in V(K)\ \forall K\in\mathcal{T}_h.\end{aligned} \tag{2.4}$$

The operator P_{V_h} is, for example, the standard L^2-projection into the finite element space V_h.

The value of the numerical flux at the point (x,t), $h_{e,K}(x,t)$, where x belongs to the edge e of the boundary of the element K, depends on the two values of the approximate solution at (x,t). One is the value obtained from the *interior* of the element K, namely,

$$u_h\big(x^{int(K)},t\big)=\lim_{y\to x,\,y\in K}u_h(y,t),$$

and the other is the value obtained from the *exterior* of the element K, namely,

$$u_h\big(x^{ext(K)},t\big)=\begin{cases}\gamma_h(x,t), & \text{if } x\in\partial\Omega,\\ \lim_{y\to x,\,y\notin K}u_h(y,t), & \text{otherwise.}\end{cases}$$

The discrete boundary values, γ_h, are the L^2-projection of the exact boundary data γ into the finite element space obtained by taking the traces of the elements of V_h into $\partial\Omega$.

The numerical flux is defined as $h_{e,K}(x,t)=h_{e,K}(u_h(x^{int(K)},t),u_h(x^{ext(K)},t))$, where $h_{e,K}$ is any two-point Lipschitz flux which is monotone in the scalar case and is an exact or approximate Riemann solver in the system case. It is also consistent with $\mathbf{f}(u)\cdot n_{e,K}$, that is,

$$h_{e,K}(u,u)=\mathbf{f}(u)\cdot n_{e,K},$$

and conservative, that is,

$$h_{e,K}\big(u_h\big(x^{int(K)}\big),u_h\big(x^{ext(K)}\big)\big)+h_{e,K'}\big(u_h\big(x^{int(K')}\big),u_h\big(x^{ext(K')}\big)\big)=0,\quad K'\cap K=e.$$

An example is the following (local) Lax–Friedrichs flux

$$h_{e,K}(a,b) = \frac{1}{2}[\mathbf{f}(a)\cdot n_{e,K} + \mathbf{f}(b)\cdot n_{e,K} - \alpha_{e,K}(b-a)], \tag{2.5}$$

where $\alpha_{e,K}$ is an estimate of the biggest eigenvalue of the Jacobian $(\partial/\partial u)\mathbf{f}(u_h(x,t))\cdot n_{e,K}$ for (x,t) in a neighborhood of the edge e.

It is convenient to take the local spaces $V(K)$ to be the space of polynomials of total degree smaller or equal to k, $P^k(K)$; in this case, we denote V_h by V_h^k. (Note that this choice is possible regardless of the shape of the elements K since the functions in V_h are discontinuous.) There are two reasons for this choice. First, if the local space $V(K)$ includes $P^k(K)$, it is possible to find $(k+1)$th order accurate approximations in $V(K)$ to any function in $W^{1,k+1}(K)$. Second, if $V(K)$ consists of polynomials only and does not include $P^{k+1}(K)$, it is not possible to find $(k+2)$th order accurate approximations in $V(K)$ to functions in $W^{1,k+2}(K)$; see [12].

Moreover, if V_h includes V_h^k, the approximation to div $\mathbf{f}(u)$ provided by the above space discretization is $(k+1)$th accurate for sufficiently smooth u, provided that the quadrature rules for the edges of the elements, (2.2), are exact for polynomials of degree $2k+1$, and the quadrature rules for the interior of the elements, (2.3), are exact for polynomials of degree $2k$ (see [16, Proposition 2.1]). It is thus reasonable to expect that the resulting scheme gives an $(k+1)$th order accurate approximation when the exact solution is smooth enough.

For the choice $V_h = V_h^0$ and quadrature rules over the edges exact for constants, the resulting scheme is nothing but a finite volume, monotone scheme in the scalar case. Thus, the discretization by the discontinuous Galerkin method can be considered as a high-order accurate extension of finite volume, monotone schemes.

2.2. *Time Discretization*

The equations defining the approximate solution can be rewritten in ODE form as $(d/dt)u_h = L_h(u_h, \gamma_h)$ after inverting the "mass" matrix. Since the functions of V_h are discontinuous, the "mass" matrix is block-diagonal and the blocks, whose orders are equal to the dimensions of the local spaces $V(K)$, can be easily inverted by hand. If a locally orthogonal basis is chosen, the mass matrix is diagonal.

If we are using a finite element space V_h included in V_h^k, we would like to discretize in time the above system of ODEs with a method that is at least $(k+1)$th-order accurate. To do that, we use the TVD Runge–Kutta time discretization introduced in [34, 35]. Thus, if $\{t^n\}_{n=0}^N$ is a partition of $[0,T]$ and $\Delta t^n = t^{n+1} - t^n$, $n = 0, \ldots, N-1$, our time-marching algorithm reads as follows:

- Set $u_h^0 = P_{V_h}(u_0)$;
- For $n = 0, \ldots, N-1$ compute u_h^{n+1} as follows:
  1. set $u_h^{(0)} = u_h^n$;
  2. for $i = 1, \ldots, k+1$ compute the intermediate functions:

$$u_h^{(i)} = \left\{ \sum_{l=0}^{i-1} \alpha_{il} u_h^{(l)} + \beta_{il}\Delta t^n L_h\left(u_h^{(l)}, \gamma_h(t^n + d_l \Delta t^n)\right) \right\};$$

  3. set $u_h^{n+1} = u_h^{(k+1)}$.

TABLE 1
Parameters of Some Practical Runge–Kutta Time Discretizations

Order	α_{il}	β_{il}	d_l	$\max\{\beta_{il}/\alpha_{il}\}$
2	1	1	0	1
	$\frac{1}{2}$ $\frac{1}{2}$	0 $\frac{1}{2}$	1	
3	1	1	0	
	$\frac{3}{4}$ $\frac{1}{4}$	0 $\frac{1}{4}$	1	1
	$\frac{1}{3}$ 0 $\frac{2}{3}$	0 0 $\frac{2}{3}$	$\frac{1}{2}$	

Note that this method is very easy to code since only a subroutine defining $L_h(u_h, \gamma_h(t))$ is needed. In this paper, we use the second-order and third-order accurate Runge–Kutta time discretizations listed below in Table 1 for piecewise linear P^1 and piecewise quadratic P^2 finite element approximations, respectively.

2.3. *The Local Slope Limiting*

In the case in which piecewise-constant approximations are considered, that is, when $V_h = V_h^0$, the artificial viscosity that the numerical flux introduces in the scheme, due to upwinding, is enough to render the scheme stable. However, when the local spaces are richer, the stabilizing influence of the numerical fluxes is not enough to guarantee the absence of spurious oscillations. To enhance the stability of the method and eliminate possible spurious oscillations in the approximate solution, a local slope limiting operator $\Lambda\Pi_h$ is introduced in the time-marching algorithm as follows:

- Set $u_h^0 = \Lambda\Pi_h\ P_{V_h}(u_0)$;
- For $n = 0, \ldots, N-1$ compute u_h^{n+1} as follows:

  1. set $u_h^{(0)} = u_h^n$;
  2. for $i = 1, \ldots, k+1$ compute the intermediate functions:

$$u_h^{(i)} = \Lambda\Pi_h \left\{ \sum_{l=0}^{i-1} \alpha_{il} u_h^{(l)} + \beta_{il} \Delta t^n L_h\left(u_h^{(l)}, \gamma_h(t^n + d_l \Delta t^n)\right) \right\};$$

  3. set $u_h^{n+1} = u_h^{(k+1)}$.

Theoretical studies of the operator $\Lambda\Pi_h$ can be found in [14] for the one-dimensional case and in [16] for the multidimensional case. Guided by these results, we use in this paper very simple, practical, and effective slope limiting operators $\Lambda\Pi_h$. To compute $\Lambda\Pi_h u_h$, we rely on the *assumption* that spurious oscillations are present in u_h only if they are present in its P^1 part u_h^1, which is its L^2-projection into the space of piecewise linear functions V_h^1; a theoretical justification of this assumption is still an open problem. Thus, if they are not present in u_h^1, i.e., if

$$u_h^1 = \Lambda\Pi_h u_h^1,$$

then we assume that they are not present in u_h and, hence, do not do any limiting:

$$\Lambda\Pi_h u_h = u_h.$$

On the other hand, if spurious oscillations are present in the P^1 part of the solution u_h^1, i.e., if

$$u_h^1 \neq \Lambda\Pi_h u_h^1,$$

then we chop off the higher order part of the numerical solution and limit the remaining P^1 part:

$$\Lambda\Pi_h u_h = \Lambda\Pi_h u_h^1.$$

In this way, in order to define $\Lambda\Pi_h$ for arbitrary space V_h, we need to actually define it for piecewise linear functions V_h^1. The exact way to do that, both for the triangular elements and for the rectangular elements, will be discussed in the next section.

3. ALGORITHM AND IMPLEMENTATION DETAILS

In this section we give the algorithm and implementation details, including numerical fluxes, quadrature rules, degrees of freedom, fluxes, and limiters of the RKDG method for both piecewise-linear and piecewise-quadratic approximations in both triangular and rectangular elements.

3.1. *Fluxes*

For the numerical flux needed in (2.4), we use the simple Lax–Friedrichs flux (2.5):

$$h_{e,K}(a, b) = \tfrac{1}{2}[\mathbf{f}(a) \cdot n_{e,K} + \mathbf{f}(b) \cdot n_{e,K} - \alpha_{e,K}(b - a)].$$

The numerical viscosity constant $\alpha_{e,K}$ should be an estimate of the biggest eigenvalue of the Jacobian $(\partial/\partial u)\mathbf{f}(u_h(x, t)) \cdot n_{e,K}$ for (x, t) in a neighborhood of the edge e. For the triangular elements, we have used the local Lax–Friedrichs recipe:

- Take $\alpha_{e,K}$ to be the larger one of the largest eigenvalue (in absolute value) of $(\partial/\partial u)\mathbf{f}(\bar{u}_K) \times n_{e,K}$ and that of $(\partial/\partial u)\mathbf{f}(\bar{u}_{K'}) \times n_{e,K}$, where $\bar{u}_K$ and $\bar{u}_{K'}$ are the means of the numerical solution in the elements K and K' sharing the edge e.

For the rectangular elements, we have used both the local Lax–Friedrichs recipe (in Examples 4.1 and 4.2) and the global Lax–Friedrichs recipe (in Example 4.3):

- Take $\alpha_{e,K}$ to be the largest of the largest eigenvalue (in absolute value) of $(\partial/\partial u)\mathbf{f}(\bar{u}_{K''}) \times n_{e,K}$, where $\bar{u}_{K''}$ is the mean of the numerical solution in the element K'', which runs over all elements on the same line (horizontally or vertically, depending on the direction of $n_{e,K}$) with K and K' sharing the edge e.

Usually, the global Lax–Friedrichs recipe is more dissipative, but it is more robust than the local Lax–Friedrichs recipe, especially for problems involving low velocities and low density/pressure near wall boundaries. There are recipes in between the two, such as taking the maximum over several neighboring elements in obtaining $\alpha_{e,K}$, but we have not used them in this paper.

3.2. *Quadrature Rules*

According to the analysis, the quadrature rules for the edges of the elements, (2.2), must be exact for polynomials of degree $2k+1$, and the quadrature rules for the interior of the elements, (2.3), must be exact for polynomials of degree $2k$, if P^k methods are used. Here we discuss the quadrature points used for P^1 and P^2 in the triangular and rectangular element cases.

3.2.1. *The rectangular elements.* For the edge integral, we use the following two point Gaussian rule

$$\int_{-1}^{1} g(x)\,dx \approx g\left(-\frac{1}{\sqrt{3}}\right) + g\left(\frac{1}{\sqrt{3}}\right) \tag{3.1}$$

for the P^1 case and the following three point Gaussian rule

$$\int_{-1}^{1} g(x)\,dx \approx \frac{5}{9}\left[g\left(-\frac{\sqrt{3}}{\sqrt{5}}\right) + g\left(\frac{\sqrt{3}}{\sqrt{5}}\right)\right] + \frac{8}{9}g(0) \tag{3.2}$$

for the P^2 case, suitably scaled to the relevant intervals.

For the interior of the elements, we could use a tensor product of (3.1), with four quadrature points, for the P^1 case. But to save cost, we "recycle" the values of the fluxes at the element boundaries and only add one new quadrature point in the middle of the element. The quadrature rule is, thus,

$$\begin{aligned}\int_{-1}^{1}\int_{-1}^{1} g(x,y)\,dx\,dy \;\approx\; \frac{1}{4}\Bigg[&g\left(-1,\frac{1}{\sqrt{3}}\right) + g\left(-1,-\frac{1}{\sqrt{3}}\right) + g\left(-\frac{1}{\sqrt{3}},-1\right)\\ &+ g\left(\frac{1}{\sqrt{3}},-1\right) + g\left(1,-\frac{1}{\sqrt{3}}\right) + g\left(1,\frac{1}{\sqrt{3}}\right) + g\left(\frac{1}{\sqrt{3}},1\right)\\ &+ g\left(-\frac{1}{\sqrt{3}},1\right)\Bigg] + 2g(0,0).\end{aligned}$$

For the P^2 case, we use a tensor product of (3.2), with nine quadrature points.

3.2.2. *The triangular elements.* For the edge integral, we use the same two point or three point Gaussian quadratures as in the rectangular case, (3.1) and (3.2), for the P^1 and P^2 cases, respectively.

For the interior integrals (2.3), we use the three mid-point rule

$$\int_K g(x,y)\,dx\,dy \approx \frac{|K|}{3}\sum_{i=1}^{3} g(m_i),$$

where m_i are the mid-points of the edges, for the P^1 case. For the P^2 case, we use a seven-point quadrature rule which is exact for polynomials of degree 5 over triangles, given in Table A.4, on page 343 of [10].

3.3. *Basis and Degrees of Freedom*

We emphasize that the choice of basis and degrees of freedom does not affect the algorithm, as it is completely determined by the choice of function space V_h in (2.1),

the numerical fluxes in (2.4), the quadrature rules, the slope limiting, and the time discretization. However, a suitable choice of basis and degrees of freedom may simplify the implementation and calculation.

3.3.1. *The rectangular elements.* For the P^1 case, we use the following expression for the approximate solution $u_h(x, y, t)$ inside the rectangular element $[x_{i-1/2},\ x_{i+1/2}] \times [y_{j-1/2},\ y_{j+1/2}]$,

$$u_h(x, y, t) = \bar{u}(t) + u_x(t)\phi_i(x) + u_y(t)\psi_j(y), \tag{3.3}$$

where

$$\phi_i(x) = \frac{x - x_i}{\Delta x_i/2}, \quad \psi_j(y) = \frac{y - y_j}{\Delta y_j/2}, \tag{3.4}$$

and

$$\Delta x_i = x_{i+1/2} - x_{i-1/2}, \quad \Delta y_j = y_{j+1/2} - y_{j-1/2}.$$

The degrees of freedoms, to be evolved in time, are then

$$\bar{u}(t), \quad u_x(t), \quad u_y(t).$$

Here we have omitted the subscripts ij these degrees of freedom should have, to indicate that they belong to the element ij which is $[x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}]$.

Notice that the basis functions

$$1, \quad \phi_i(x), \quad \psi_j(y),$$

are orthogonal; hence, the local mass matrix is diagonal:

$$M = \Delta x_i \Delta y_j \operatorname{diag}\left(1, \frac{1}{3}, \frac{1}{3}\right).$$

For the P^2 case, the expression for the approximate solution $u_h(x, y, t)$ inside the rectangular element $[x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}]$ is

$$\begin{aligned} u_h(x, y, t) = \bar{u}(t) &+ u_x(t)\phi_i(x) + u_y(t)\psi_j(y) + u_{xy}(t)\phi_i(x)\psi_j(y) \\ &+ u_{xx}(t)\left(\phi_i^2(x) - \frac{1}{3}\right) + u_{yy}(t)\left(\psi_j^2(y) - \frac{1}{3}\right), \end{aligned} \tag{3.5}$$

where $\phi_i(x)$ and $\psi_j(y)$ are defined by (3.4). The degrees of freedom, to be evolved in time, are

$$\bar{u}(t), \quad u_x(t), \quad u_y(t), \quad u_{xy}(t), \quad u_{xx}(t), \quad u_{yy}(t).$$

Again the basis functions

$$1, \quad \phi_i(x), \quad \psi_j(y), \quad \phi_i(x)\psi_j(y), \quad \phi_i^2(x) - \frac{1}{3}, \quad \psi_j^2(y) - \frac{1}{3}$$

are orthogonal; hence, the local mass matrix is diagonal:

$$M = \Delta x_i \Delta y_j \operatorname{diag}\left(1, \frac{1}{3}, \frac{1}{3}, \frac{1}{9}, \frac{4}{45}, \frac{4}{45}\right).$$

3.3.2. *The triangular elements.* For the P^1 case, we use the expression for the approximate solution $u_h(x, y, t)$ inside the triangle K,

$$u_h(x, y, t) = \sum_{i=1}^{3} u_i(t)\varphi_i(x, y),$$

where the degrees of freedom $u_i(t)$ are values of the numerical solution at the midpoints of edges, and the basis function $\varphi_i(x, y)$ is the linear function which takes the value 1 at the mid-point m_i of the ith edge, and the value 0 at the mid-points of the two other edges. The mass matrix is diagonal:

$$M = |K|\text{diag}\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right).$$

For the P^2 case, we use the expression for the approximate solution $u_h(x, y, t)$ inside the triangle K,

$$u_h(x, y, t) = \sum_{i=1}^{6} u_i(t)\xi_i(x, y),$$

where the degrees of freedom, $u_i(t)$, are values of the numerical solution at the three midpoints of edges and the three vertices. The basis function $\xi_i(x, y)$, is the quadratic function which takes the value 1 at the point i of the six points mentioned above (the three midpoints of edges and the three vertices), and the value 0 at the remaining five points. The mass matrix, S, this time is not diagonal (see page 11 in [23]):

$$S = |K| \begin{pmatrix} 1/30 & -1/180 & -1/180 & -1/45 & 0 & 0 \\ -1/180 & 1/30 & -1/180 & 0 & -1/45 & 0 \\ -1/180 & -1/180 & 1/30 & 0 & 0 & -1/45 \\ -1/45 & 0 & 0 & 8/45 & 4/45 & 4/45 \\ 0 & -1/45 & 0 & 4/45 & 8/45 & 4/45 \\ 0 & 0 & -1/45 & 4/45 & 4/45 & 8/45 \end{pmatrix}.$$

3.4. *Limiting*

We construct slope-limiting operators $\Lambda\Pi_h$ on piecewise linear functions u_h in such a way that the following properties are satisfied:

1. Accuracy: if u_h is linear then $\Lambda\Pi_h u_h = u_h$.
2. Conservation of mass: for every element K of the triangulation $\mathcal{T}_h$, we have

$$\int_K \Lambda\Pi_h u_h = \int_K u_h.$$

3. Slope limiting: on each element K of $\mathcal{T}_h$, the gradient of $\Lambda\Pi_h u_h$ is not bigger than that of u_h.

The actual form of the slope limiting operators is closely related to that of the slope limiting operators studied in [14, 16].

3.4.1. *The rectangular elements.* The limiting is performed on u_x and u_y in (3.3), using the differences of the means. For a scalar equation, u_x would be limited (replaced) by

$$\bar{m}(u_x, \bar{u}_{i+1,j} - \bar{u}_{ij}, \bar{u}_{ij} - \bar{u}_{i-1,j}), \tag{3.6}$$

where the function $\bar{m}$ is the TVB corrected *minmod* function [33, 14] defined by

$$\bar{m}(a_1, \ldots, a_m) = \begin{cases} a_1, & \text{if } |a_1| \le M\Delta x^2, \\ m(a_1, \ldots, a_m), & \text{otherwise,} \end{cases} \tag{3.7}$$

with the *minmod* function m defined by

$$m(a_1, \ldots, a_m) = \begin{cases} s \min_i |a_i|, & \text{if } s = \text{sign}\,(a_1) = \cdots = \text{sign}\,(a_m), \\ 0, & \text{otherwise.} \end{cases}$$

The TVB correction is needed to avoid unnecessary limiting near smooth extrema, where the quantity u_x or u_y is on the order of $O(\Delta x^2)$ or $O(\Delta y^2)$. For an estimate of the TVB constant M in terms of the second derivatives of the function, see [14]. Usually, the numerical results are not sensitive to the choice of M in a large range. In all the calculations in this paper we take M to be 50.

Similarly, u_y is limited (replaced) by

$$\bar{m}(u_y, \bar{u}_{i,j+1} - \bar{u}_{ij}, \bar{u}_{ij} - \bar{u}_{i,j-1})$$

with a change of Δx to Δy in (3.7).

For systems, we perform the limiting in the local characteristic variables. To limit the vector u_x in the element ij, we proceed as follows:

- Find the matrix R and its inverse R^{-1} which diagonalize the Jacobian evaluated at the mean in the element ij in the x-direction,

$$R^{-1} \frac{\partial f_1(\bar{u}_{ij})}{\partial u} R = \Lambda,$$

where Λ is a diagonal matrix containing the eigenvalues of the Jacobian. Notice that the columns of R are the right eigenvectors of $\partial f_1(\bar{u}_{ij})/\partial u$ and the rows of R^{-1} are the left eigenvectors.

- Transform all quantities needed for limiting, i.e., the three vectors u_{xij}, $\bar{u}_{i+1,j} - \bar{u}_{ij}$, and $\bar{u}_{ij} - \bar{u}_{i-1,j}$, to the characteristic fields. This is achieved by left-multiplying these three vectors by R^{-1}.
- Apply the scalar limiter (3.6) to each of the components of the transformed vectors.
- The result is transformed back to the original space by left multiplying R on the left.

3.4.2. *The triangular elements.* To construct the slope-limiting operators for triangular elements, we proceed as follows. We start by making a simple observation. Consider the triangles in Fig. 3.1, where m_1 is the mid-point of the edge on the boundary of K_0 and b_i denotes the barycenter of the triangle K_i for $i = 0, 1, 2, 3$.

Since we have that

$$m_1 - b_0 = \alpha_1(b_1 - b_0) + \alpha_2(b_2 - b_0)$$

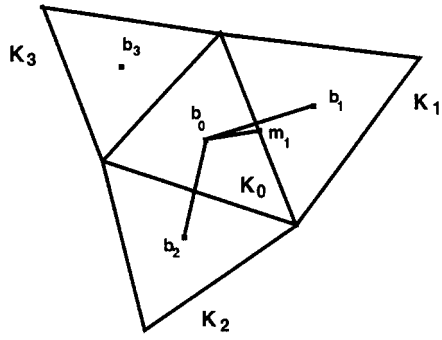

FIG. 3.1. Illustration of limiting.

for some nonnegative coefficients $\alpha_1,\ \alpha_2$ which depend only on m_1 and the geometry, we can write for any linear function u_h

$$u_h(m_1) - u_h(b_0) = \alpha_1(u_h(b_1) - u_h(b_0)) + \alpha_2(u_h(b_2) - u_h(b_0)),$$

and since

$$\bar{u}_{K_i} = \frac{1}{|K_i|}\int_{K_i} u_h = u_h(b_i), \quad i = 0, 1, 2, 3,$$

we have that

$$\tilde{u}_h(m_1, K_0) \equiv u_h(m_1) - \bar{u}_{K_0} = \alpha_1\left(\bar{u}_{K_1} - \bar{u}_{K_0}\right) + \alpha_2\left(\bar{u}_{K_2} - \bar{u}_{K_0}\right) \equiv \Delta\bar{u}(m_1, K_0).$$

Now, we are ready to describe the slope limiting. Let us consider a piecewise linear function u_h, and let m_i, $i = 1, 2, 3$ be the three mid-points of the edges of the triangle K_0. We then can write for $(x, y) \in K_0$

$$u_h(x, y) = \sum_{i=1}^{3} u_h(m_i)\varphi_i(x, y) = \bar{u}_{K_0} + \sum_{i=1}^{3} \tilde{u}_h(m_i, K_0)\varphi_i(x, y).$$

To compute $\Lambda\Pi_h u_h$, we first compute the quantities

$$\Delta_i = \bar{m}(\tilde{u}_h(m_i, K_0), \nu\Delta\bar{u}(m_i, K_0)),$$

where $\bar{m}$ is the TVB modified *minmod* function defined in (3.7), and $\nu > 1$. We take $\nu = 1.5$ in our numerical runs. Then, if $\sum_{i=1}^{3} \Delta_i = 0$, we simply set

$$\Lambda\Pi_h u_h(x, y) = \bar{u}_{K_0} + \sum_{i=1}^{3} \Delta_i\varphi_i(x, y).$$

If $\sum_{i=1}^{3} \Delta_i \neq 0$, we compute

$$\text{pos} = \sum_{i=1}^{3} \max(0, \Delta_i), \quad \text{neg} = \sum_{i=1}^{3} \max(0, -\Delta_i)$$

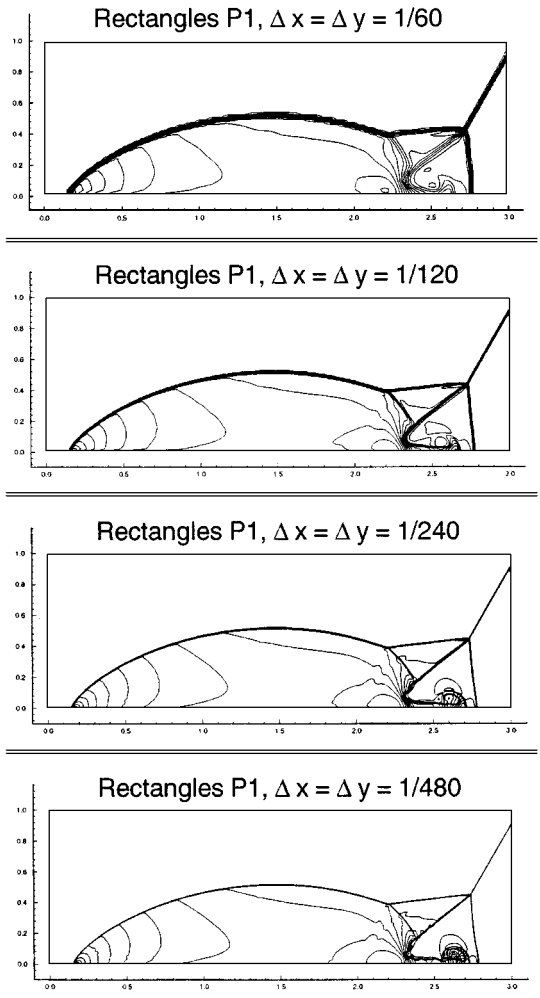

FIG. 4.1. Double Mach reflection problem. Second-order P^1 results. Density ρ; 30 equally spaced contour lines from $\rho = 1.3965$ to $\rho = 22.682$. Mesh refinement study. From top to bottom: $\Delta x = \Delta y = \frac{1}{60}, \frac{1}{120}, \frac{1}{240}$, and $\frac{1}{480}$.

and set

$$\theta^+ = \min\left(1, \frac{\text{neg}}{\text{pos}}\right), \quad \theta^- = \min\left(1, \frac{\text{pos}}{\text{neg}}\right).$$

Then, we define

$$\Lambda\Pi_h u_h(x, y) = \bar{u}_{K_0} + \sum_{i=1}^{3} \hat{\Delta}_i \varphi_i(x, y),$$

where

$$\hat{\Delta}_i = \theta^+ \max(0, \Delta_i) - \theta^- \max(0, -\Delta_i).$$

It is very easy to see that this slope-limiting operator satisfies the three properties listed above.

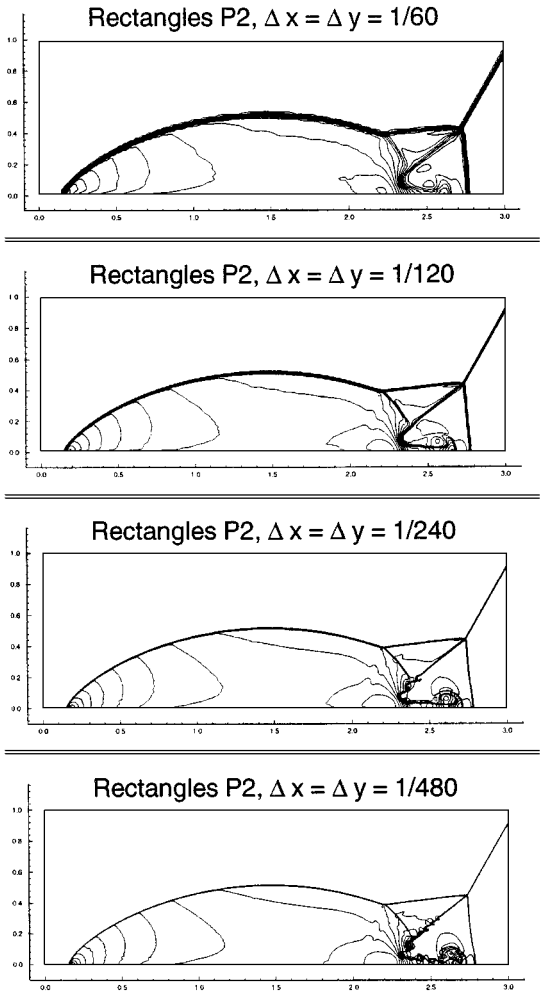

FIG. 4.2. Double Mach reflection problem. Third-order P^2 results. Density ρ; 30 equally spaced contour lines from $\rho = 1.3965$ to $\rho = 22.682$. Mesh refinement study. From top to bottom: $\Delta x = \Delta y = \frac{1}{60}, \frac{1}{120}, \frac{1}{240}$, and $\frac{1}{480}$.

For systems, we perform the limiting in the local characteristic variables. To limit Δ_i, we proceed as in the rectangular case, the only difference being that we work with the Jacobian

$$\frac{\partial}{\partial u} f\left(\overline{u}_{K_0}\right) \cdot \frac{m_i - b_0}{|m_i - b_0|}.$$

4. NUMERICAL RESULTS

In this section we present several numerical results obtained with the P^1 and P^2 (second- and third-order accurate) RKDG methods with either rectangular or triangular elements. We consider several standard test problems for Euler equations of compressible gas dynamics.

For all the runs, we take $CFL = 0.3$ for P^1 and $CFL = 0.18$ for P^2; recall that the von Neumann analysis for the one-dimensional case gives the stability condition $CFL \leq 1/3$ for P^1 and $CFL \leq 1/5$ for P^2 (see [14]). For the rectangles, we take CFL equal to the maximum over the rectangles of $((|v_x| + c)/\Delta x + (|v_y| + c)/\Delta y)\Delta t$, where c is the speed

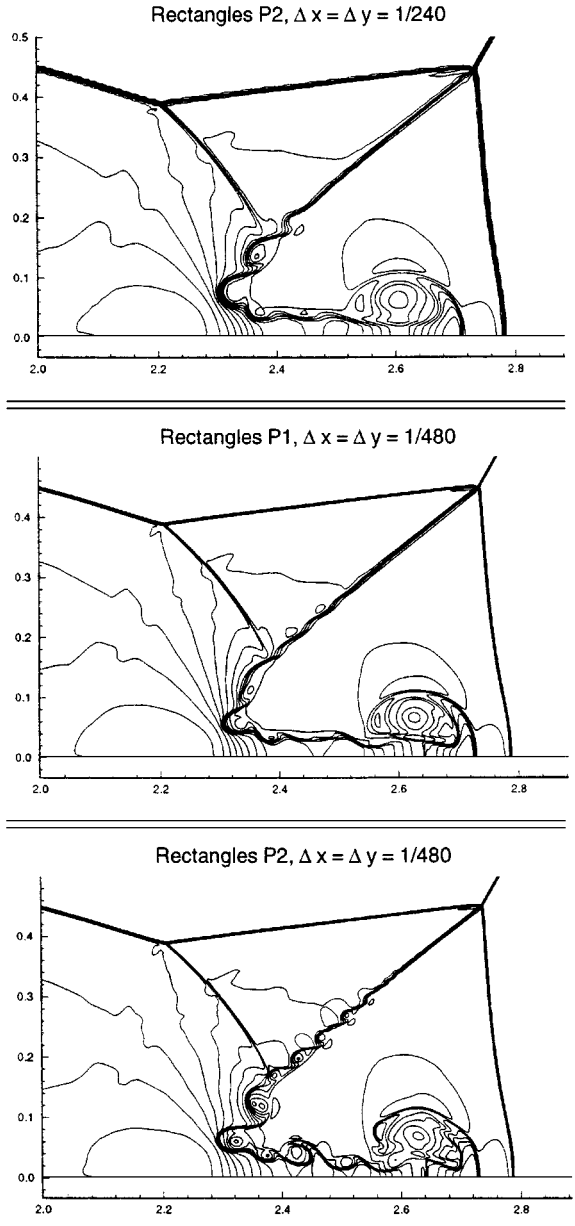

FIG. 4.3. Double Mach reflection problem. Blown-up region around the double Mach stems. Density ρ. Third-order P^2 with $\Delta x = \Delta y = \frac{1}{240}$ (top); second-order P^1 with $\Delta x = \Delta y = \frac{1}{480}$ (middle); and third-order P^2 with $\Delta x = \Delta y = \frac{1}{480}$ (bottom).

of sound and (v_x, v_y) is the velocity both evaluated at the local average. For the triangles, we take *CFL* to be a more conservative quantity, namely, the maximum over the triangles of $(\|(v_x, v_y)\| + c)\text{perimeter}(K)\Delta t/|K|$.

EXAMPLE 4.1. Double Mach reflection of a strong shock. This problem was studied extensively in Woodward and Colella [37] and later by many others. We use exactly the same setup as in [37], namely, a Mach 10 shock initially making a 60° angle with a reflecting wall. The undisturbed air ahead of the shock has a density of 1.4 and a pressure of 1.

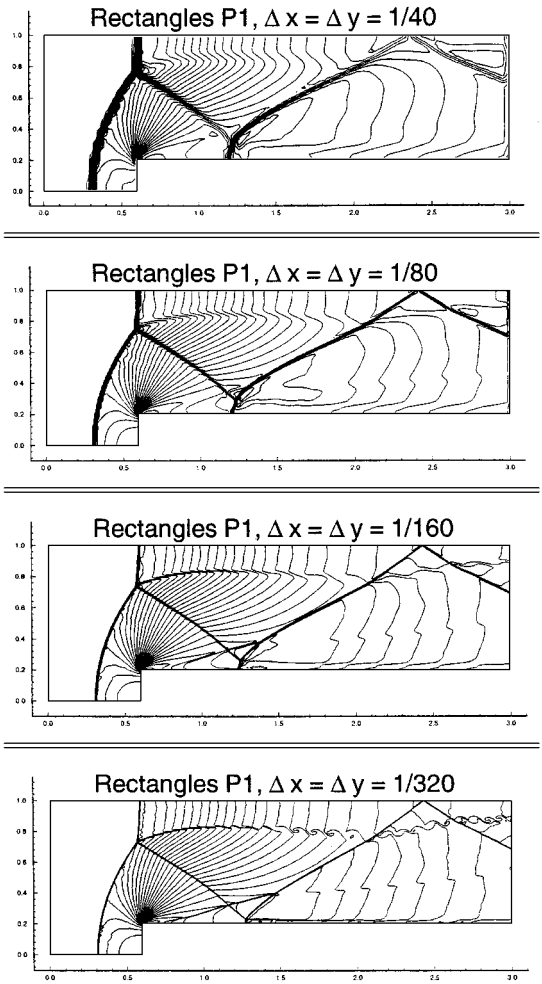

FIG. 4.4. Forward-facing step problem. Second-order P^1 results. Density ρ; 30 equally spaced contour lines from $\rho = 0.090338$ to $\rho = 6.2365$. Mesh refinement study. From top to bottom: $\Delta x = \Delta y = \frac{1}{40}, \frac{1}{80}, \frac{1}{160}$, and $\frac{1}{320}$.

We use rectangular elements for this problem. The computational domain is $[0, 4] \times [0, 1]$, as in [37]. The reflecting wall lies at the bottom of the computational domain for $\frac{1}{6} \leq x \leq 4$. 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 postshock condition is imposed for the part from $x = 0$ to $x = \frac{1}{6}$, to mimic an angled wedge. Reflective boundary condition is used for the rest. At the top boundary of our computational domain, the flow values are set to describe the exact motion of the Mach 10 shock. Inflow/outflow boundary conditions are used for the left and right boundaries. The results at $t = 0.2$ are shown. As in [37], only the results in $[0, 3] \times [0, 1]$ are displayed.

Four different uniform meshes are used: 240×60 elements ($\Delta x = \Delta y = \frac{1}{60}$); 480×120 elements ($\Delta x = \Delta y = \frac{1}{120}$); 960×240 elements $\Delta x = \Delta y = \frac{1}{240}$); and 1920×480 elements ($\Delta x = \Delta y = \frac{1}{480}$). The density is plotted in Fig. 4.1 for the P^1 case and in Fig. 4.2 for the P^2 case. In all the plots, we use 30 contours equally distributed from $\rho = 1.3965$ to $\rho = 22.682$.

It is not easy to observe any significant difference between the P^1 and P^2 results in these pictures. However, if we show a "blown-up" portion around the double Mach region, as in

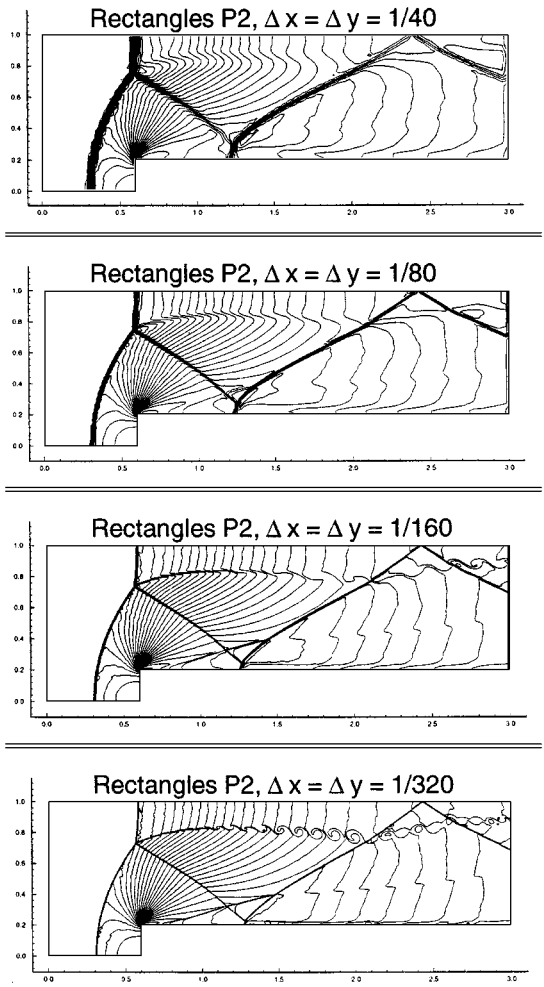

FIG. 4.5. Forward-facing step problem. Third-order P^2 results. Density ρ; 30 equally spaced contour lines from $\rho = 0.090338$ to $\rho = 6.2365$. Mesh refinement study. From top to bottom: $\Delta x = \Delta y = \frac{1}{40}, \frac{1}{80}, \frac{1}{160}$, and $\frac{1}{320}$.

Fig. 4.3, we can see that P^2 with $\Delta x = \Delta y = \frac{1}{240}$ has qualitatively the same resolution as P^1 with $\Delta x = \Delta y = \frac{1}{480}$ for the fine details of the complicated structure in this region. Notice that this detailed structure is of physical interest and was studied before with an adaptive grid calculation in [7]. P^2 with $\Delta x = \Delta y = \frac{1}{480}$ gives a much better resolution for these structures than P^1 with the same number of elements.

The conclusion here is that, if one is interested in such fine structures, then one can use the third-order scheme P^2 with only half of the mesh points in each direction as in P^1. This translates to a reduction of a factor of 8 in space-time cells for 2D time dependent problems and will more than offset the increase of cost per cell and the smaller CFL number by using the higher order P^2 method (the cpu saving for this problem is around a factor of 2.1 in our implementation). This saving will be even more significant for 3D.

The optimal strategy, of course, is to use adaptivity and concentrate cells around the interesting region, and/or to change the degree of the polynomial in different regions.

EXAMPLE 4.2. Flow past a forward facing step. This problem was again studied extensively in Woodward and Colella [37] and later by many others. The setup of the problem

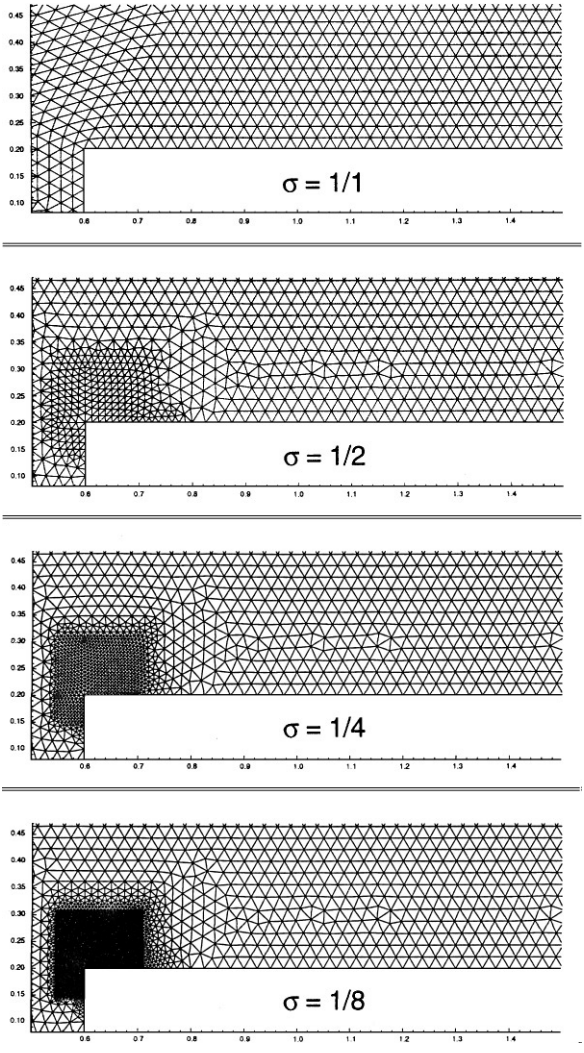

FIG. 4.6. Forward-facing step problem. The triangulations used for results in Figs. 4.7 and 4.8. σ is the ratio between the typical size of the triangles near the corner and that elsewhere.

is the following: a right-going Mach 3 uniform flow enters a wind tunnel of 1 unit wide and 3 units long. The step is 0.2 units high and is located 0.6 units from the left-hand end of the tunnel. The problem is initialized by a uniform, right-going Mach 3 flow. Reflective boundary conditions are applied along the walls of the tunnel and in-flow and out-flow boundary conditions are applied at the entrance (left-hand end) and the exit (right-hand end), respectively. The results at $t = 4$ are shown.

The corner of the step is a singularity, which we study carefully in our numerical experiments. Unlike in [37] and in many other papers, we do not modify our scheme near the corner in any way. It is well known that this leads to an erroneous entropy layer at the downstream bottom wall, as well as a spurious Mach stem at the bottom wall. However, these artifacts decrease when the mesh is refined. In Fig. 4.4, second-order P^1 results using rectangular elements are shown for a mesh refinement study using

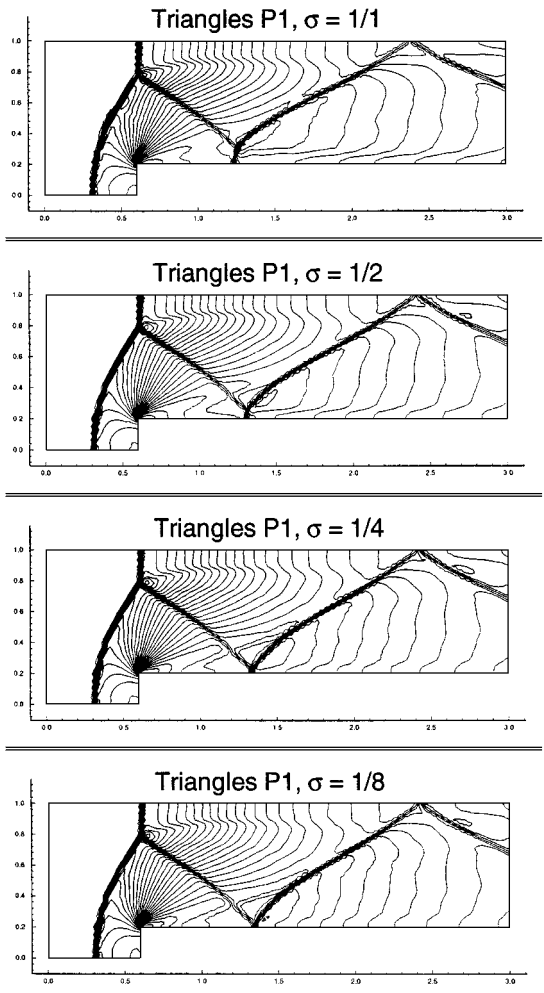

FIG. 4.7. Forward-facing step problem. Second-order P^1 results. Density ρ; 30 equally spaced contour lines from $\rho = 0.090338$ to $\rho = 6.2365$. Triangle code. Progressive refinement near the corner, using the triangulations shown in Fig. 4.6.

$\Delta x = \Delta y = \frac{1}{40}, \frac{1}{80}, \frac{1}{160}$, and $\frac{1}{320}$ as element sizes. We can clearly see the improved resolution (especially at the upper slip line from the triple point) and decreased artifacts caused by the corner, with decreased element sizes. In Fig. 4.5, third-order P^2 results using the same sequence of elements are shown. Comparing with the P^1 results in Fig. 4.4, we can see that the resolution is improved, especially for the slip line issued from the triple point.

In order to verify that the erroneous entropy layer at the downstream bottom wall and the spurious Mach stem at the bottom wall are both artifacts caused by the poor resolution of the corner singularity, we use our triangle code to locally refine near the corner progressively. A sequence of such triangulation is shown in Fig. 4.6, where σ is the ratio between the typical size of the triangles near the corner and that elsewhere. The resolution of the meshes away from the corner is roughly equal to a rectangular element case of $\Delta x = \Delta y = \frac{1}{40}$, i.e., the top pictures in Figs. 4.4 and 4.5. The density results using P^1 and these triangulations are shown in Fig. 4.7, those using P^2 are shown in Fig. 4.8. We can see that, with more

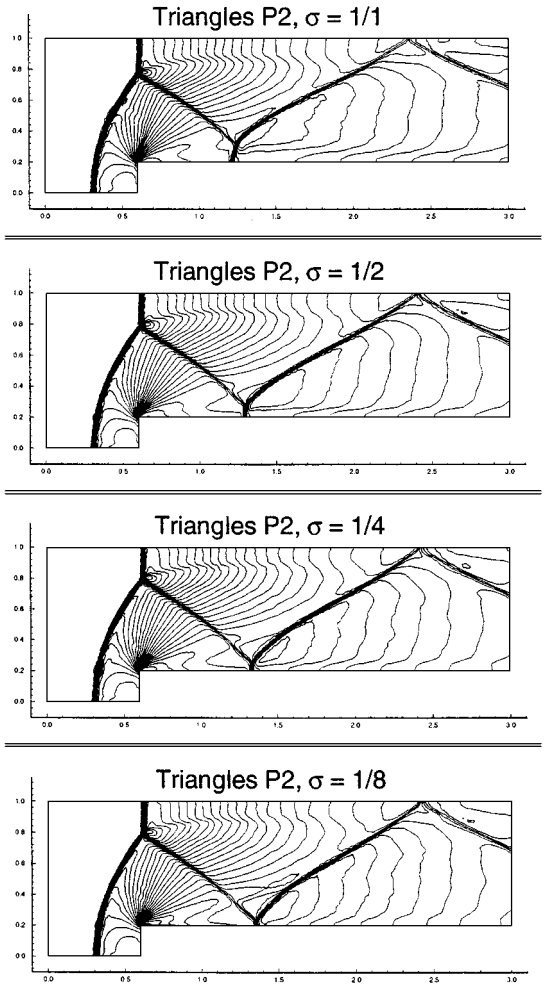

FIG. 4.8. Forward-facing step problem. Third-order P^2 results. Density ρ; 30 equally spaced contour lines from $\rho = 0.090338$ to $\rho = 6.2365$. Triangle code. Progressive refinement near the corner, using the triangulations shown in Fig. 4.6.

triangles concentrated near the corner, the artifacts gradually decrease. Notice that there is a strong spurious entropy production near the corner, which pollutes the flow downstream. With progressive refinement near the corner, this spurious entropy production decreases; see Figs. 4.9 and 4.10.

These are the only triangular element runs we present in this paper. We can see that the triangular elements can give results of the same resolution quality as the rectangular case with roughly the same mesh density for both P^1 and P^2. We do observe, however, that a positivity correction procedure is needed for the triangular element runs for this case. During the projection of the linear part, we check whether the density and the total energy are negative at the three mid-points of the edges of K. If they are, further limiting is performed to bring them to 10^{-10} in a conservative way.

EXAMPLE 4.3. Shock passing a backward facing corner (diffraction). This example has been used in [22, 30] (see also the experimental results in [6]). The setup of the problem is the

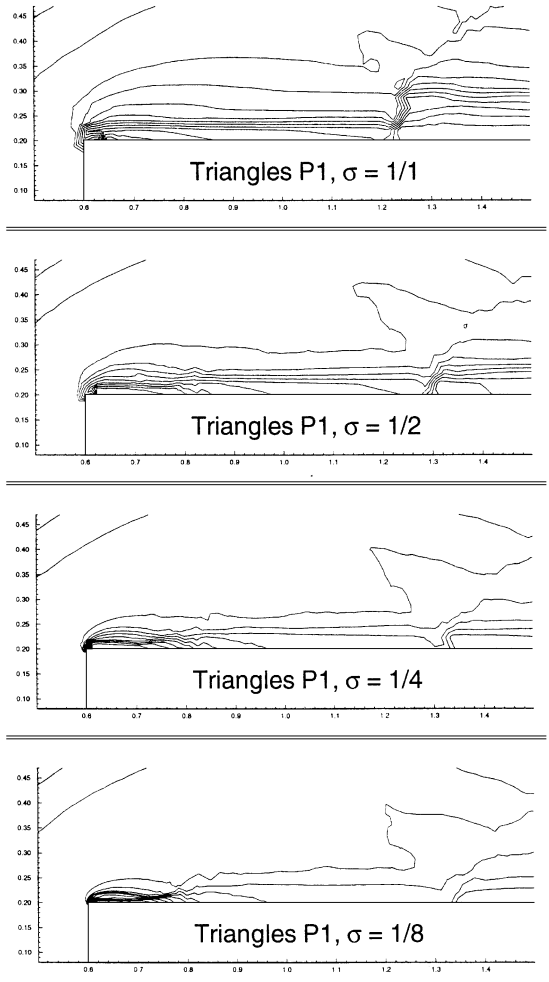

FIG. 4.9. Forward-facing step problem. Second-order P^1 results. Spurious entropy production near the corner. Progressive refinement near the corner, using the triangulations shown in Fig. 4.6.

following: the computational domain is the union of $[0, 1]\times[6, 11]$ and $[1, 13]\times[0, 11]$; the initial condition is a pure right-moving shock of *Mach* $= 5.09$, initially located at $x = 0.5$ and $6 \le y \le 11$, moving into undisturbed air ahead of the shock with a density of 1.4 and a pressure of 1. The boundary conditions are inflow at $x = 0$, $6 \le y \le 11$, outflow at $x = 13$, $0 \le y \le 11$, reflective at $0 \le x \le 1$, $y = 6$ and at $x = 1$, $0 \le y \le 6$, and Neumann at $1 \le x \le 13$, $y = 0$ and at $0 \le x \le 13$, $y = 11$. No special treatment is done at the corner which is a singularity of the solution. The density at $t = 2.3$ is presented in Fig. 4.11 for the P^1 case and in Fig. 4.12 for the P^2 case. Rectangular meshes are used with four different mesh sizes $\Delta x = \Delta y = \frac{1}{10}, \frac{1}{20}, \frac{1}{40}$, and $\frac{1}{80}$, respectively.

We remark that it is easy to get negative density and/or pressure for this problem. In both our P^1 and P^2 runs, we found it necessary to perform a positivity correction procedure.

For the P^1 case, we check for each element whether the density, as a linear function, is too close to zero in the element. Specifically, using the notation of (3.3), we check if

$$\overline{u} - |u_x| - |u_y| < \tfrac{1}{2}\overline{u},$$

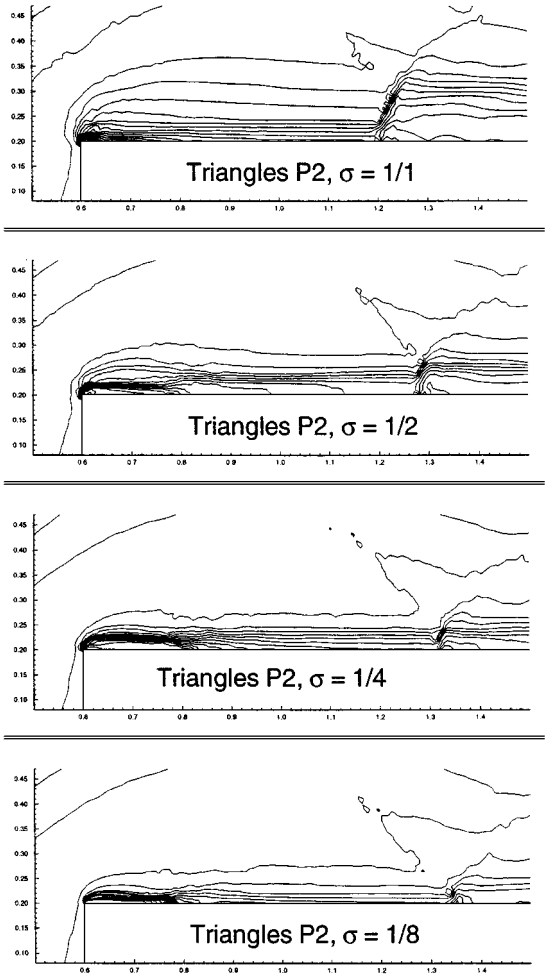

FIG. 4.10. Forward-facing step problem. Third-order P^2 results. Spurious entropy production near the corner. Progressive refinement near the corner, using the triangulations shown in Fig. 4.6.

and, if yes, the slopes u_x and u_y are reduced by a factor:

$$\text{factor} = \frac{1}{2}\frac{\overline{u}}{|u_x| + |u_y|}.$$

The same correction procedure is performed on the total energy. We do not modify the two momenta.

For the P^2 case, a somewhat stronger positivity correction procedure is needed. We check for each element whether the density or the total energy is too close to zero. Using the notation of (3.5), we check if

$$\overline{u} - |u_x| - |u_y| - |u_{xy}| - \tfrac{2}{3}(|u_{xx}| + |u_{yy}|) < 0$$

for either the density or the total energy, and, if yes, *all* the degrees of freedom, except the mean

$$u_x, \quad u_y, \quad u_{xy}, \quad u_{xx}, \quad u_{yy},$$

of all the components (density, two momenta, and total energy), are reduced by a factor

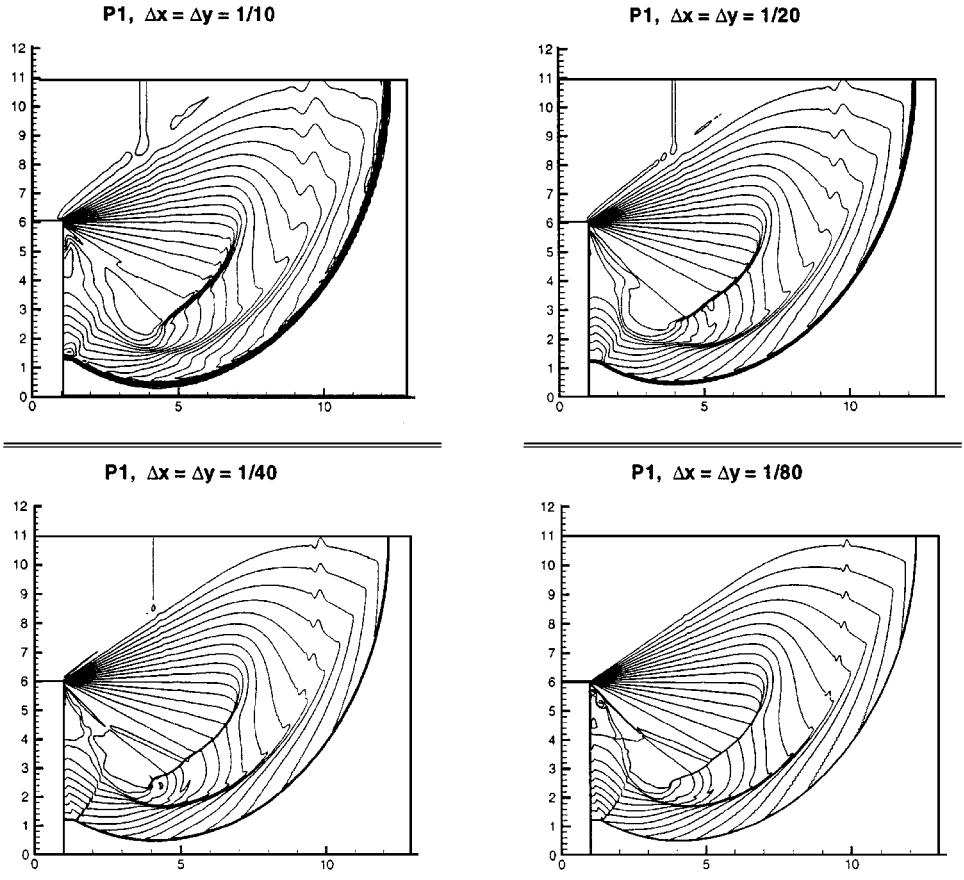

FIG. 4.11. Shock diffraction problem. Second-order P^1 results. Density ρ; 20 equally spaced contour lines from $\rho = 0.066227$ to $\rho = 7.0668$. Mesh refinement study. From top to bottom: $\Delta x = \Delta y = \frac{1}{10}, \frac{1}{20}, \frac{1}{40}$, and $\frac{1}{80}$.

which is the smaller of the two quantities

$$\frac{\bar{u}}{|u_x| + |u_y| + |u_{xy}| + \frac{2}{3}(|u_{xx}| + |u_{yy}|)},$$

computed from the density and the total energy.

We remark that the positivity correction procedures described above are conservative and do not degrade the formal accuracy of the schemes.

5. CONCLUDING REMARKS

We have presented the algorithm formulation and practical implementation issues of the RKDG (Runge–Kutta discontinuous Galerkin) methods, for multidimensional systems and in particular for the compressible Euler equations of gas dynamics. Numerical results are shown. We conclude in particular that for detailed features in the flow, such as the structure near the triple Mach stem in the double Mach reflection problem, a higher order method gives better cpu performance than a lower order one, to obtain the same resolution. We also conclude that triangular elements and rectangular elements perform in a similar way.

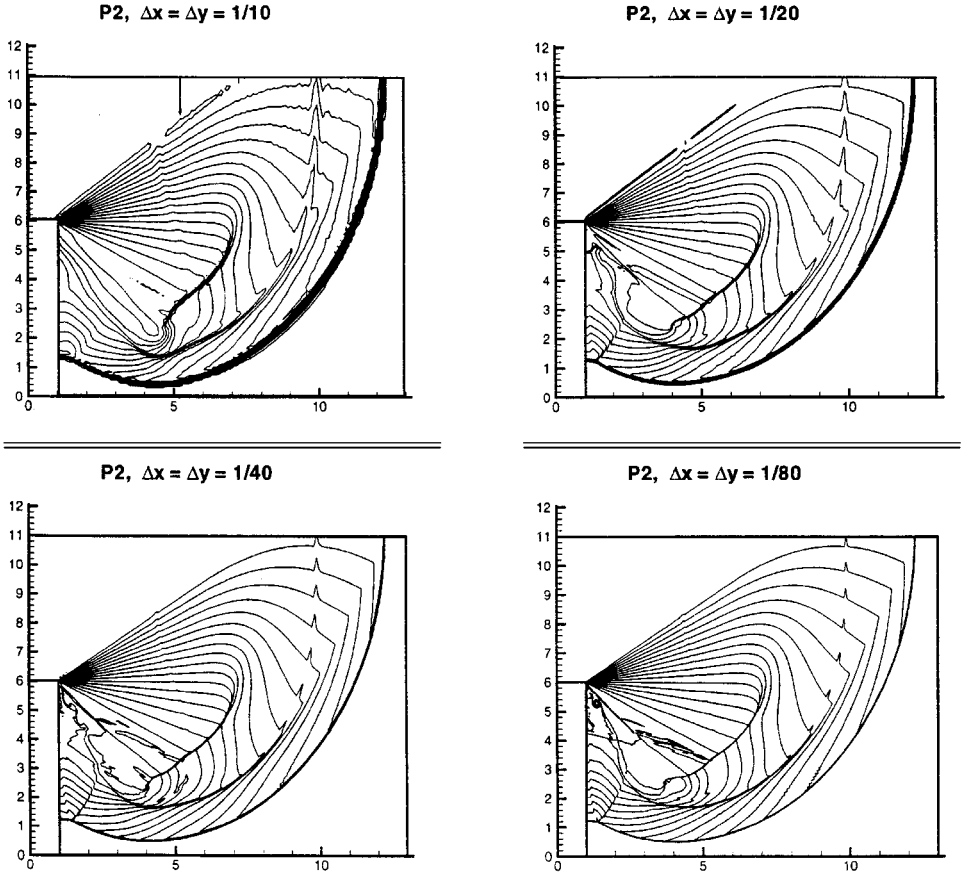

FIG. 4.12. Shock diffraction problem. Third-order P^2 results. Density ρ; 20 equally spaced contour lines from $\rho = 0.066227$ to $\rho = 7.0668$. Mesh refinement study. From top to bottom: $\Delta x = \Delta y = \frac{1}{10}, \frac{1}{20}, \frac{1}{40}$, and $\frac{1}{80}$.

ACKNOWLEDGMENTS

The paper was finished when both authors were visiting the Institute for Mathematics and Its Applications, University of Minnesota. We thank IMA for support and hospitality.

REFERENCES

1. H. L. Atkins and C.-W. Shu, *Quadrature-free Implementation of Discontinuous Galerkin Methods for Hyperbolic Equations*, ICASE Report 96-51, 1996. [Submitted to *AIAA J.*]
2. P. Bar-Yoseph, Space-time discontinuous finite element approximations for multi-dimensional nonlinear hyperbolic systems, *Comput. Mech.* **5**, 145 (1989).
3. P. Bar-Yoseph and D. Elata, An efficient L_2 Galerkin finite element method for multi-dimensional nonlinear hyperbolic systems, *Int. J. Numer. Methods Eng.* **29**, 1229 (1990).
4. F. Bassi and S. Rebay, High-order accurate discontinuous finite element solution of the 2D Euler equations, *J. Comput. Phys.*, to appear.
5. 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**, 267 (1997).
6. S. Bazhenova, L. Gvozdeva, and M. Nettleton, Unsteady interactions of shock waves, *Prog. Aerosp. Sci.* **21**, 249 (1984).

7. M. Berger and A. Colella, Local adaptive mesh refinement for shock hydrodynamics, *J. Comput. Phys.* **82**, 64 (1989).
8. K. S. Bey and J. T. Oden, A Runge–Kutta discontinuous Galerkin finite element method for high speed flows, *AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 24–27, 1991*.
9. R. Biswas, K. D. Devine, and J. Flaherty, Parallel, adaptive finite element methods for conservation laws, *Applied Numerical Mathematics* **14**, 255 (1994).
10. G. F. Carey and J. T. Oden, *Finite Elements: Computational Aspects, III* (Prentice-Hall, Englewood Cliffs, NJ, 1984).
11. G. Chavent and B. Cockburn, The local projection P^0P^1-discontinuous Galerkin finite element method for scalar conservation laws, M^2AN **23**, 565 (1989).
12. P. Ciarlet, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1975.
13. B. Cockburn and C. W. Shu, The Runge-Kutta local projection P^1-discontinuous Galerkin method for scalar conservation laws, M^2AN **25**, 337 (1991).
14. B. Cockburn and C. W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework, *Math. Comp.* **52**, 411 (1989).
15. B. Cockburn, S. Y. Lin, and C. W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems, *J. Comput. Phys.* **84**, 90 (1989).
16. 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, *Math. Comp.* **54**, 545 (1990).
17. B. Cockburn and C. W. Shu, *The P^1-RKDG Method for Two-dimensional Euler Equations of Gas Dynamics*, ICASE Report 91-32, 1991.
18. B. Cockburn and C. W. Shu, The local discontinuous Galerkin method for time-dependent convection diffusion systems, *SIAM J. Numer. Anal.*, to appear.
19. H. L. deCougny, K. D. Devine, J. E. Flaherty, R. M. Loy, C. Özturan, and M. S. Shephard, High-order accurate discontinuous discontinuous finite element solution of the 2D Euler equations, *Applied Numerical Mathematics* **16**, 157 (1994).
20. K. D. Devine, J. E. Flaherty, R. M. Loy, and S. R. Wheat, *Parallel Partitioning Strategies for the Adaptive Solution of Conservation Laws*, Rensselaer Polytechnic Institute Report No. 94-1, 1994.
21. K. D. Devine, J. E. Flaherty, S. R. Wheat, and A. B. Maccabe, *A Massively Parallel Adaptive Finite Element Method with Dynamic Load Balancing*, SAND Report 93-0936C, 1993.
22. R. Hillier, Computation of shock wave diffraction at a ninety degrees convex edge, *Shock Waves* **1**, 89 (1991).
23. S. Hou, *A Finite Element Method for Conservation Laws: Multidimensional Case* (Ph.D. Thesis, School of Mathematics, University of Minnesota, 1991).
24. G. Jiang and C.-W. Shu, On cell entropy inequality for discontinuous Galerkin methods, *Math. Comp.* **62**, 531 (1994).
25. C. Johnson and J. Pitkäranta, An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation, *Math. Comp.* **46**, 1 (1986).
26. P. LeSaint and P. A. Raviart, *On a finite element method for solving the neutron transport equation*, Mathematical aspects of finite elements in partial differential equations (C. de Boor, Ed.), Academic Press, 89 (1974).
27. R. B. Lowrie, P. L. Roe and B. van Leer, Space-time discontinuous Galerkin: I. Theory and properties, Preprint.
28. C. Özturan, H. L. deCougny, M. S. Shephard, and J. E. Flaherty, Parallel adaptive mesh refinement and redistribution on distributed memory computers, *Comput. Methods Appl. Mech. Engrg.* **119**, 123 (1994).
29. T. Peterson, A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation, *SIAM J. Numer. Anal.* **28**, 133 (1991).
30. J. Quirk, A construction to the great Riemann solver debate, *Int. J. Numer. Meth. Fluids* **18**, 555 (1994).
31. W. H. Reed and T. R. Hill, *Triangular Mesh Methods for the Neutron Transport Equation*, Los Alamos Scientific Laboratory Report LA-UR-73-479, 1973.
32. G. R. Richter, An optimal-order error estimate for the discontinuous Galerkin method, *Math. Comp.* **50**, 75 (1988).

33. C.-W. Shu, TVB uniformly high-order schemes for conservation laws, *Math. Comp.* **49**, 105 (1987).
34. C.-W. Shu, Total-variation-diminishing time discretizations, *SIAM J. Sci. Stat. Comput.* **9**, 1073 (1988).
35. C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *J. Comput. Phys.* **77**, 439 (1988).
36. C.-W. Shu and S. Osher, Efficient implementation of essentially non-oscillatory shock capturing schemes, II, *J. Comput. Phys.* **83**, 32 (1989).
37. P. Woodward and P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, *J. Comput. Phys.* **54**, 115 (1984).