Journal of Computational Physics **150,** 143–180 (1999)
Article ID jcph.1998.6167, available online at http://www.idealibrary.com on IDEAL®

Conservative Smoothing on an Adaptive Quadrilateral Grid

M. Sun* and K. Takayama†

Shock Wave Research Center, Institute of Fluid Science, Tohoku University, Katahira 2-1-1, Aoba, Sendai 980, Japan
E-mail: *sun@ceres.ifs.tohoku.ac.jp and †takayama@ifs.tohoku.ac.jp

Received December 31, 1997; revised December 3, 1998

The Lax–Wendroff scheme can be freed of spurious oscillations by introducing conservative smoothing. In this paper the approach is first tested in 1-D modeling equations and then extended to multidimensional flows by the finite volume method. The scheme is discretized by a space-splitting method on an adaptive quadrilateral grid. The artificial viscosity coefficients in the conservative smoothing step are specially designed to capture slipstreams and vortices. Algorithms are programmed using a vectorizable data structure, under which not only the flow solver but also the adaptation procedure is well vectorized. The good resolution and high efficiency of the approach are demonstrated in calculating both unsteady and steady compressible flows with either weak or strong shock waves.

Key Words: central scheme; artificial dissipation; conservative smoothing; vectorization; adaptation; quadrilateral grid.

1. INTRODUCTION

Consider the convection equation

$$u_t + cu_x = 0, \tag{1}$$

where c is a wave speed. Independent variables are time t and spatial coordinate x. Among finite difference schemes for (1), the second-order Lax–Wendroff scheme is one of the most important, due to its uniqueness for linear equations and its essential role as the guideline for many schemes. A well-known deficiency of the scheme is that it may produce spurious oscillations around discontinuities.

In order to remove the unavoidable oscillations, artificial dissipation [21] is often added, but the amount of dissipation necessary is not clearly known. Artificial dissipation terms are usually designed to be second order or less in order to maintain the accuracy of the original

0021-9991/99 $30.00

scheme. However, an exact solution of Navier–Stokes equations shows that the thickness of a shock wave changes linearly with the physical viscosity coefficient (see for example [15, p. 266]). If we try to numerically imitate the physical dissipation by an artificial one and attempt to produce a shock wave in a few cells, the artificial viscosity coefficient should also be a linear function of the cell size, or the coefficient should be first order. Adding a first-order term to (1), it becomes

$$u_t + cu_x = \frac{\Delta x^2}{\Delta t}(\varepsilon u_x)_x, \tag{2}$$

where ε is a dimensionless artificial viscosity coefficient. Equation (2) can be split into the *convection step*

$$\tilde{u}_t + c\tilde{u}_x = 0 \tag{3}$$

and the *smoothing step*

$$u_t = \frac{\Delta x^2}{\Delta t}(\varepsilon \tilde{u}_x)_x. \tag{4}$$

The second-order Lax–Wendroff scheme is employed to solve the convection step. The computational sequence of the two steps is trivial because the difference is only at the first time step. In the smoothing step, we propose a viscosity coefficient

$$\varepsilon = \begin{cases} 0, & \text{if } \phi < \varepsilon_1; \\ \varepsilon_2, & \text{if } \phi \geq \varepsilon_1, \end{cases} \tag{5}$$

where

$$\phi = \left| \frac{1/2\tilde{u}''\Delta x^2}{\tilde{u}'\Delta x} \right| \tag{6}$$

and $\varepsilon_1 = 0.7$, $\varepsilon_2 = 1/4$. This nonlinear coefficient does not deteriorate the order of the Lax–Wendroff scheme. The value $\varepsilon_1 = 0.7$ was optimized by numerical experiments. Another value $\varepsilon_2 = 1/4$ may dissipate spurious oscillations with the shortest wavelength most efficiently. The indicator ϕ is a measure of the degree of flow variations. Further discussions on these parameters will be presented in Section 2.

The first successful example of a central scheme with nonlinear dissipation, to our knowledge, is that of Boris and Book's antidiffusion method or flux-corrected-transport (FCT) method [3]. The approach can be interpreted as having two stages:

1. adding artificial viscosity everywhere during solving of the convection equation;
2. recovering the lost accuracy in the first stage by applying negative diffusion everywhere except near extrema where oscillations might occur.

Note that the net diffusion is coupled in solving the convection equation. If we sum these two stages and rearrange the dissipation order, a simpler logic is:

1. compute the convection equation without any additional artificial viscosity;
2. add artificial viscosity in the position where oscillations might occur.

These are just the convection step and the smoothing step. It is seen that the smoothing method is simpler and hopefully more efficient in removing oscillations since the dissipation is added after solving the convection equation. A one-step scheme which can be proven to be TVD for the scalar convection equation will be presented in a separate paper.

Actually, the efficiency of the smoothing step in removing oscillations has been recognized for many years. It was applied to suppress the instability in solving the K-dV equation [1]. Engquist *et al.* [5] devised a few nonlinear filtering algorithms in conjunction with numerical schemes. Shyy *et al.* [16] further showed that these filters could be a remedy to eliminate the oscillations in a few model problems. They also numerically demonstrated the effects of a filter on waves with different wavelengths. Their observations will be explained by the spectral analysis in Section 2 (Fig. 2b). However, a similar filter which they used had been generally tested in solving Euler equations about a decade before by Russian colleagues ([9], and references therein cited). They tried to correct the nonmonotonic property of the MacCormack scheme. A recent numerical example can be found in [6]. However, this result still suffers from "quivering" contours and excessive smoothness across a slipstream. Up to now all authors only tested the smoothing methods on structured grids.

In the present work, the proposed smoothing method is first tested on 1-D model problems and then extended to adaptive unstructured quadrilaterals by the finite volume method. The robustness of the approach is validated by calculating a variety of gas dynamic problems. This paper is organized as follows. Section 2 describes the performance of the smoothing method in solving the linear wave equation and the 1-D Euler equations. Section 3 puts forward a vectorized locally adaptive method which will be combined with the flow solver. Section 4 is devoted to the smoothing step and the convection step on unstructured quadrilaterals. The vectorization of the approach is also discussed. Section 5 gives numerical examples for unsteady and steady flow calculations. Section 6 summarizes the paper and remarks on what can be improved in future.

2. MODEL PROBLEMS

2.1. *The Scalar Wave Equation*

For the convection step (3), the Lax–Wendroff scheme reads

$$\tilde{u}_i^{n+1} = u_i^n - \frac{\sigma}{2}\left(u_{i+1}^n - u_{i-1}^n\right) + \frac{\sigma^2}{2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right), \tag{7}$$

where $\sigma = c\frac{\Delta t}{\Delta x}$ is the CFL number. The smoothing step (4) is conservatively discretized by the central difference

$$u_i^{n+1} = \tilde{u}_i^{n+1} + \varepsilon_{i+1/2}\left(\tilde{u}_{i+1}^{n+1} - \tilde{u}_i^{n+1}\right) - \varepsilon_{i-1/2}\left(\tilde{u}_i^{n+1} - \tilde{u}_{i-1}^{n+1}\right) \tag{8}$$

and $\varepsilon_{i+1/2} = \mathrm{Max}(\varepsilon_i, \varepsilon_{i+1})$. Note that for constant $\varepsilon = \varepsilon_2$, (8) becomes

$$u_i^{n+1} = \tilde{u}_i^{n+1} + \varepsilon_2\left(\tilde{u}_{i+1}^{n+1} - 2\tilde{u}_i^{n+1} + \tilde{u}_{i-1}^{n+1}\right). \tag{9}$$

Since ε_2 is independent of Δx and Δt, (9) is simply to smooth the solution of (7). It is clear that for $\varepsilon_2 = 0$ the smoothing step has no influence on the solution of the convection step, so it does not change the accuracy; but for $\varepsilon_2 \neq 0$ it smoothes the solution and decreases the accuracy to first order. Numerically (6) is discretized, by a central difference for a uniform grid, to give

$$\phi = \frac{|u_{i+1} + u_{i-1} - 2u_i|}{\varepsilon_0 + |u_{i+1} - u_{i-1}|}, \tag{10}$$

where ε_0 is a small value to prevent a zero denominator for constant u regions.

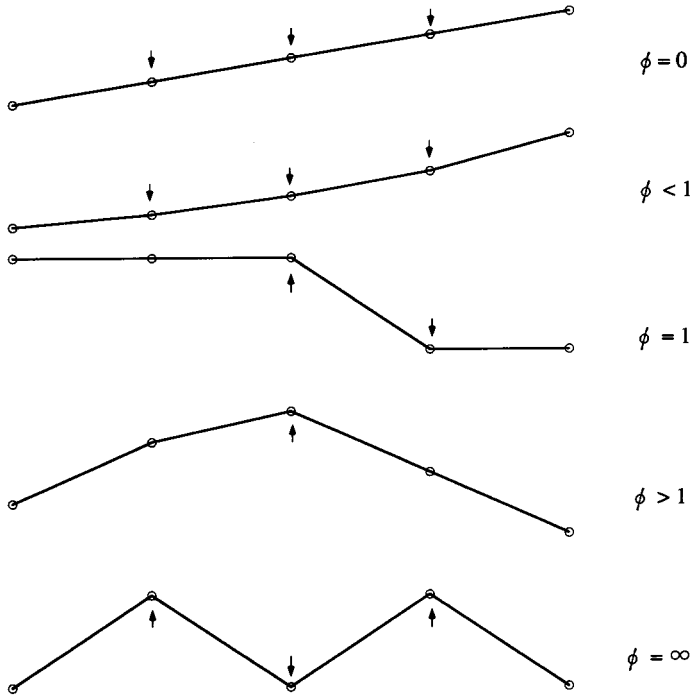

FIG. 1. Five representative stages of ϕ.

The ratio ϕ in (10) plays an essential role in limiting the artificial viscosity in (5). Understanding its significance is helpful in choosing the parameter ε_1. Figure 1 shows five representative stages of ϕ. For any linear function, ϕ is always equal to 0; for a monotonic function, ϕ is not larger than 1; for the critical case of a monotonic function with sharp changes, $\phi = 1$; if there is an extremum in a function, ϕ at the extremum is greater than 1; for the worst case of an oscillation with wavelength $2\Delta x$, the ratio ϕ approaches infinity. Therefore, the ratio ϕ is a quantitative *indicator* of whether the function is smooth or oscillating as well as providing some information about the properties of the smooth and/or oscillating regions. The critical value $\phi = 1$ corresponds to shock discontinuities. As it is generally known that sufficient artificial viscosity is needed near a shock discontinuity, one can conclude that ε_1 should be close to 1. It can be easily proven by an algebraic method that a sufficient and necessary condition of a monotonic sequence is that $\phi < 1$ (see the Appendix). Note that $\phi = 1$ or $\phi = \infty$ is independent of the jump of a discontinuity or an oscillation. It is an asset to be able to compute discontinuities with different strengths and to remove various oscillations using the same indicator.

The stability limitation is solved by the von Neumann method. Considering a single harmonic wave applied to (9), one gets the amplification factor of the smoothing step, G_s, by inserting

$$\tilde{u}^{n+1} = e^{ikx} \tag{11}$$

into (9), for $\phi > \varepsilon_1$,

$$G_s = \frac{u_i^{n+1}}{\tilde{u}_i^{n+1}} = 1 - 2\varepsilon_2 + 2\varepsilon_2 \cos(k\Delta x). \tag{12}$$

The stability condition is

$$0 < \varepsilon_2 < 1/2. \tag{13}$$

We are especially interested in the high-frequency wave with $k\Delta x = \pi$, or wavelength $2\Delta x$. Its amplification factor is

$$G_s = 1 - 4\varepsilon_2. \tag{14}$$

Note that an oscillation cannot be further damped for $\varepsilon_2 > 1/4$, but changes the sign and may generate a new extremum. A more suitable limitation of the smoothing step is $0 < \varepsilon_2 \leq 1/4$. The most efficient dissipation for the shortest wave, corresponding to $G_s = 0$, is when

$$\varepsilon_2 = 1/4. \tag{15}$$

Using this value, a harmonic wave with wavelength $2\Delta x$ is dissipated in one smoothing step. It will be invariably applied to solve one-dimensional Euler equations.

We now consider the effect of ε_1 on the harmonic wave. Substituting (11) into (10), one may obtain

$$\phi = \frac{|\cos(k\Delta x) - 1|}{|\sin(k\Delta x)i|} \approx \tan(k\Delta x/2), \tag{16}$$

where the phase shift in the denominator has been neglected.[1] Then the amplification factor of the smoothing step becomes

$$G_s = \begin{cases} 1, & \text{if } \tan(k\Delta x/2) < \varepsilon_1; \\ 1 - 2\varepsilon_2 + 2\varepsilon_2 \cos(k\Delta x), & \text{if } \tan(k\Delta x/2) \geq \varepsilon_1. \end{cases} \tag{17}$$

The amplification factor of the Lax–Wendroff scheme is (see, for example, [10])

$$|G_{LW}| = |1 - \sigma^2 + \sigma^2 \cos(k\Delta x) - i\sigma \sin(k\Delta x)|, \tag{18}$$

which is shown in Fig. 2a. The present scheme combines the Lax–Wendroff scheme and the smoothing step, so the amplification factor is $|G_s G_{LW}|$, which is shown in Fig. 2b for $\varepsilon_1 = 0.7$ and $\varepsilon_2 = 1/4$. It is seen that for long waves the factor is close to 1, and for short waves it jumps to a much smaller value, then decreases to zero for the wavelength $\lambda = 2\Delta x$. The spectral meaning of the two parameters ε_1 and ε_2 is clear: ε_1 defines the jump position $k\Delta x = 2\tan^{-1}(\varepsilon_1)$, while ε_2 gives the minimum amplification factor $1 - 4\varepsilon_2$. For $\varepsilon_1 = 0.7$ and $\varepsilon_2 = 1/4$, all oscillations with $\lambda \leq 5\Delta x$ are efficiently dampened. This is not so for the Lax–Wendroff scheme.

[1] As pointed out by a reviewer, neglecting the phase shift here gives the wrong norm for stability analysis, but gives some rough guide.

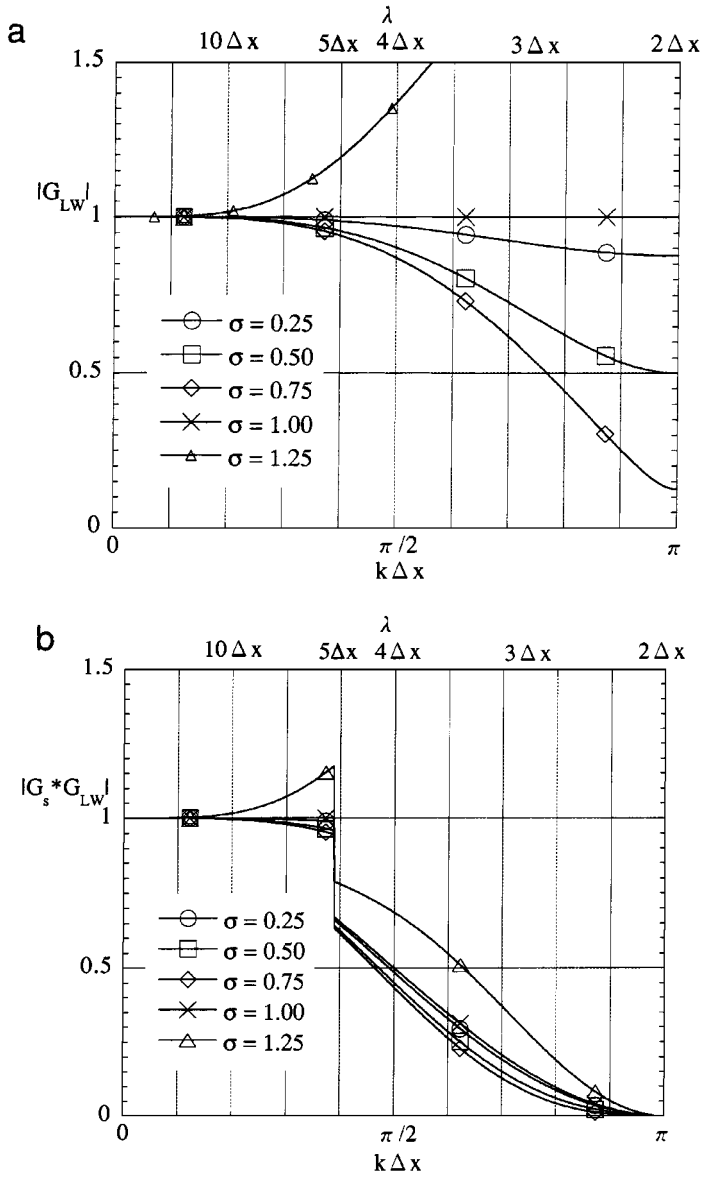

FIG. 2. Amplification factors: (a) the Lax–Wendroff scheme; (b) the Lax–Wendroff scheme with the smoothing step, $\varepsilon_1 = 0.7$, $\varepsilon_2 = 1/4$.

The effects of ε_1 and ε_2 are further tested by numerical experiments. The results of the propagation of a step wave running at different ε_1 are shown in Fig. 3a for $\sigma = 0.95$. It is seen that $\varepsilon_1 = 0.7$ is small enough to obtain a virtually monotonic solution. The effect of the artificial viscosity coefficient ε_2 is investigated by setting $\varepsilon_1 = 0.8$ while changing ε_2. Figure 3b shows the results. $\varepsilon_2 = 0$ corresponds to the Lax–Wendroff scheme, plotted by crosses. For all $\varepsilon_2 > 0$, the amplitudes of the spurious oscillations are attenuated. Note that only for $\varepsilon_2 = 1/4$ is the numerical solution monotonic behind the sharp discontinuity, while the smoothing does not introduce much dissipation in front of it. Therefore, the following results appearing in this paper are run at $\varepsilon_1 = 0.7$ and $\varepsilon_2 = 1/4$.

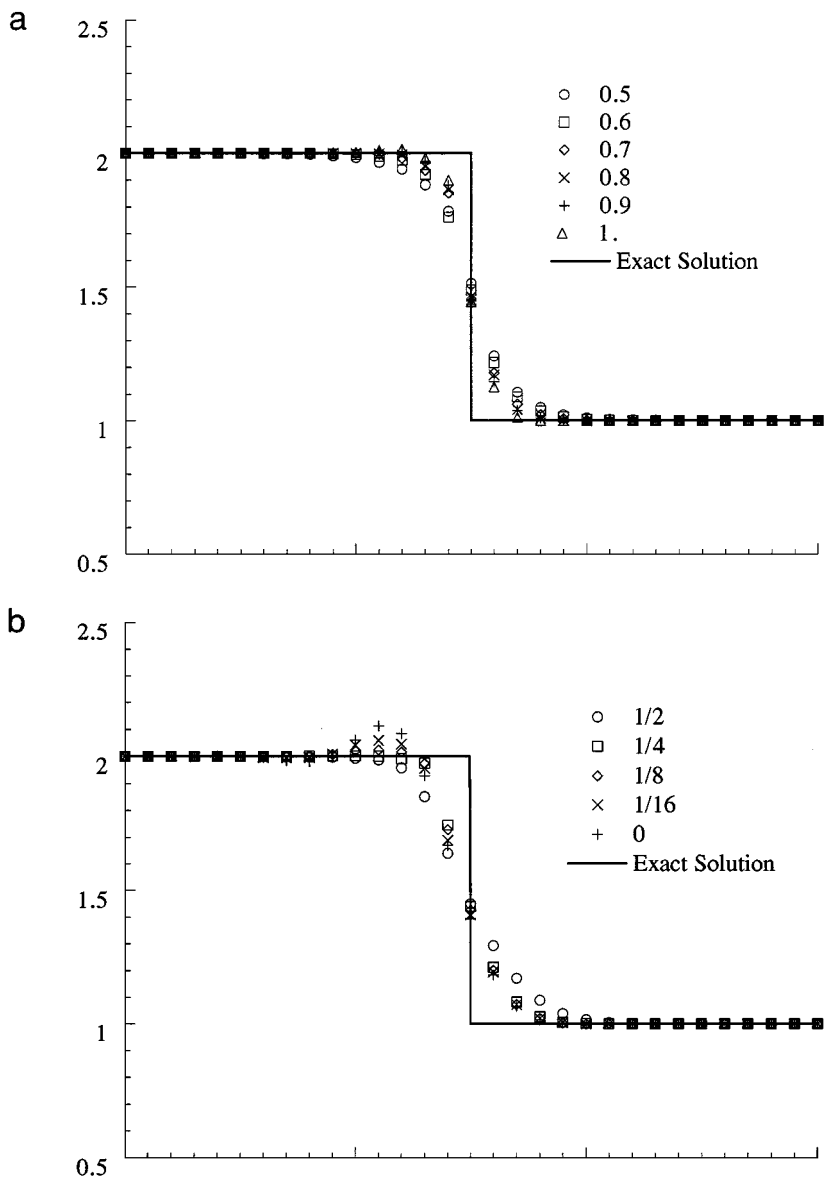

FIG. 3. Effects of ε_1 and ε_2 on step wave convection, $\sigma = 0.95$: (a) $\varepsilon_1 = 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$ for $c_2 = 1/4$; (b) $\varepsilon_2 = 1/2, 1/4, 1/8, 1/16, 0$ for $\varepsilon_1 = 0.8$.

2.2. *The Euler Equation*

We now consider one-dimensional Euler equations, which in their complete 1-D form are

$$\begin{gathered} U_t + F_x = 0 \\ U = \begin{pmatrix} \rho \\ \rho u \\ \rho e \end{pmatrix}, \qquad F = \begin{pmatrix} \rho u \\ \rho u u + p \\ \rho e u + p u \end{pmatrix}, \end{gathered} \tag{19}$$

where ρ, u, and p are density, velocity, and pressure, respectively. The specific total energy

e satisfies, for an ideal gas,

$$e = \frac{1}{\gamma - 1}\frac{p}{\rho} + 1/2u^2. \tag{20}$$

We solve the equations in two steps, convection step and smoothing step. The convection step is discretized in a conventional way. To avoid the estimation of the Jacobians, a two-step Lax–Wendroff scheme, known as the Richtmyer scheme, is chosen. In the linear case the two-step scheme becomes identical to a single-step Lax–Wendroff scheme. The smoothing step is the same as that in the scalar case. The full scheme is

$$U_{i+1/2}^{n+1/2} = 1/2\big(U_i^n + U_{i+1}^n\big) - \frac{\Delta t}{2\Delta x}\big(F_{i+1}^n - F_i^n\big) \tag{21}$$

$$\tilde{U}_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}\big(F_{i+1/2}^{n+1/2} - F_{i-1/2}^{n+1/2}\big) \tag{22}$$

$$U_i^{n+1} = \tilde{U}_i^{n+1} + \varepsilon_{i+1/2}\big(\tilde{U}_{i+1}^{n+1} - \tilde{U}_i^{n+1}\big) - \varepsilon_{i-1/2}\big(\tilde{U}_i^{n+1} - \tilde{U}_{i-1}^{n+1}\big) \tag{23}$$

$$\varepsilon_{i+1/2} = \max(\varepsilon_i, \varepsilon_{i+1}), \qquad \varepsilon_i = \varepsilon_2(1 + \operatorname{sign}(\phi - \varepsilon_1))/2$$

$$\phi = \frac{|\tilde{\rho}_{i+1} + \tilde{\rho}_{i-1} - 2\tilde{\rho}_i|}{\varepsilon_0\tilde{\rho}_i + |\tilde{\rho}_{i+1} - \tilde{\rho}_{i-1}|}, \qquad \varepsilon_1 = 0.7, \varepsilon_2 = 1/4.$$

One can recognize (21) and (22) as being identical to the Richtmyer scheme. Compared with the scalar case, the density $\tilde{\rho}$ is chosen as a key variable to indicate the regions with high gradients in the smoothing step (23). Since the density is always positive, the small value ε_0 is replaced with $\varepsilon_0\tilde{\rho}_i$ to make the indicator dimensionless, and $\varepsilon_0 = 10^{-4}$ during computations. To limit artificial viscosity, another generally used key variable is pressure, which, however, fails to detect a contact surface. During our early computations, both key variables were used. We found no clear differences when using just the density. The viscosity coefficient ε_i is written in a more concise form as in the relation (5). ε_1 and ε_2 are simply taken as constants. Since the parameters are unchanged, the scheme has no free parameter in this sense.

The Sod shock tube problem [17] is solved by the scheme for $\gamma = 1.4$. The left-side and right-side values are given as

$$(\rho, u, p)_L = (1, 0, 1),$$
$$(\rho, u, p)_R = (0.1, 0, 0.125).$$

The condition is an ordinary shock tube problem which has been tested by many authors. The initial diaphragm is at $x = 0.5$, and 101 nodes are used. The solution contains a shock wave, a contact surface, and expansion waves. The numerical results are shown in Fig. 4. There are no significant oscillations associated with any discontinuities. Similar results are also obtained for other shock tube problems which will not be shown here. Concerning the computational efficiency, the scheme costs about 25% more CPU time than the original Lax–Wendroff scheme in the 1-D tests.

3. THE VECTORIZED LOCALLY ADAPTIVE ALGORITHM

In recent years, unstructured meshes have been widely used in computational fluid dynamics [20]. In order to have more accurate numerical solutions, various grid adaptation methods have been successfully applied to solve problems with multiscale physical

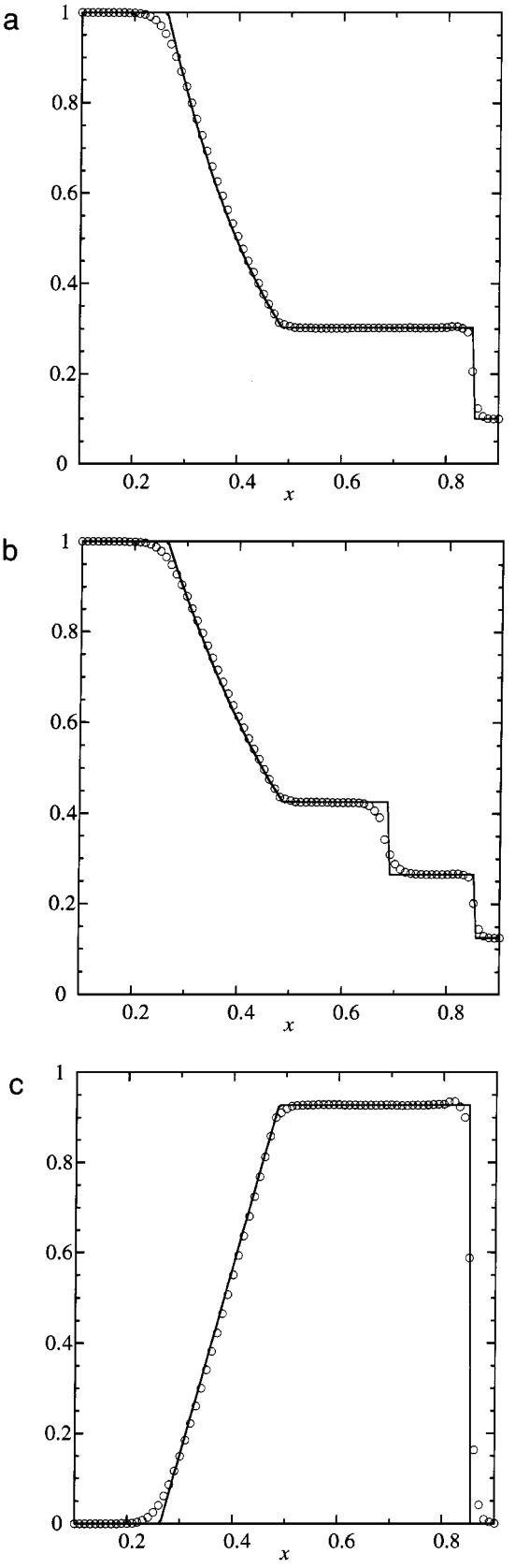

FIG. 4. 1-D shock tube problem: (a) pressure; (b) density; (c) velocity.

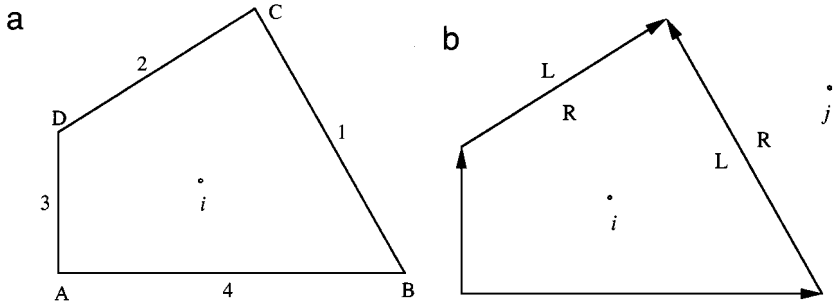

FIG. 5. Cell-edge data structure: (a) cell information—a cell points to four edges; (b) edge information—an edge points to two neighboring cells.

phenomena, such as shock waves. In most shock-capturing calculations, a locally refined grid may contain nodes which are orders of magnitude less than those of a uniform grid achieving the same accuracy. These adaptive methods have reached a level of maturity [13]. However, adaptation methods were usually organized for scalar execution, and the area of vector-adaptive algorithms is relatively unexplored [11]. To be able to vectorize the procedure, we put forward a locally adaptive method for quadrilateral cells in this section. A data structure is designed to vectorize the adaptation procedure as well as the flow solver. So, we start from a description of the data structure.

A fundamental nature of the data structure is that every cell points to its four edges, and every edge points to its two neighboring cells. The basic information saved for a cell is its location and the indices of four edges. The four edges are uniquely ordered. In Fig. 5a, edges BC, CD, DA, AB are neighboring edges NE1, NE2, NE3, NE4, respectively. Every edge is defined as a directional segment, as shown in Fig. 5b. The direction assures that the left of NE1, NE4 and the right of NE2, NE3 are the cell itself, while the right of NE1, NE4 and the left of NE2, NE3 correspond to the right, below, above, and left neighboring cells, respectively.

The basic information saved for an edge is the locations of its two ends and the indices of its left and right cells. Note that a cell never directly points to another cell but rather through their common edge. For instance, if cell i needs the information from its right cell j, then it should get the right index of its NE1, as shown in Fig. 5b. It is clear that one adjacent connectivity only needs two memory reads during computation.

Basic cell-edge addressing is well known (see, for example, [14]). The novel feature of the present addressing is the definition of edge direction and edge order for a cell, which is described above. This strict definition reduces conditional statements in the code, especially in adaptation, and thus enhances the efficiency. Note that the definition requires no additional memory, but stores the necessary information in the definite order. Another advantage of the data structure is its convenience in vectorization. It is generally known that data dependency prevents vectorization. Since a cell does not refer to another cell except through an edge, any operation on a cell is independent of other cells. This property simplifies the vectorization not only in solving conservation laws but also in refining and coarsening grids.

We first describe the adaptation procedure. The criterion for adaptation is based on the Taylor series expansion of density. The criterion is

$$\textit{Refine} = \mathrm{Max}[\phi_i, \phi_j] > \varepsilon_r$$
$$\textit{Coarse} = \mathrm{Max}[\phi_i, \phi_j] < \varepsilon_c,$$

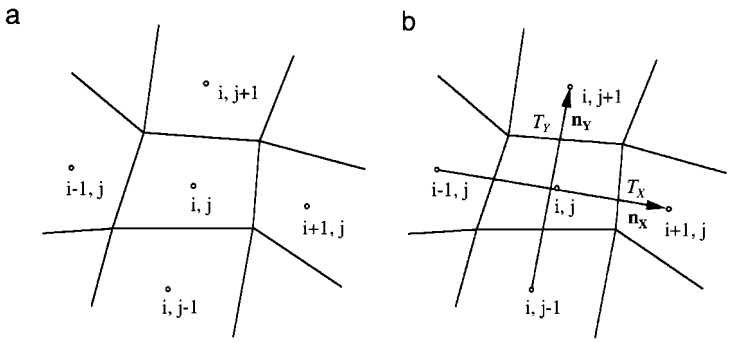

FIG. 6. Cell-adjacent information.

where *Refine* and *Coarse* are logical flags which indicate whether a cell needs to be refined or coarsened. ε_r and ε_c are threshold values for refinement and coarsening. Two error indicators, ϕ_i and ϕ_j, are given by the ratio of the second-order derivative term to the first-order one in the Taylor expansion, so that

$$\phi_i = \frac{|\tilde{\rho}_{i+1,j} + \tilde{\rho}_{i-1,j} - 2\tilde{\rho}_{i,j}|}{\alpha\tilde{\rho}_{i,j} + |\tilde{\rho}_{i+1,j} - \tilde{\rho}_{i-1,j}|}, \qquad \phi_j = \frac{|\tilde{\rho}_{i,j+1} + \tilde{\rho}_{i,j-1} - 2\tilde{\rho}_{i,j}|}{\alpha\tilde{\rho}_{i,j} + |\tilde{\rho}_{i,j+1} - \tilde{\rho}_{i,j-1}|}. \tag{24}$$

The locations of density appearing in (24) are shown in Fig. 6a. The tilde (˜) denotes that the density is a newly updated one in solving the conservative laws which will be discussed in the next section. Two subscripts in (24) represent the locations opposite cell (i, j) in two directions. It does not imply that a two-dimensional array is used to save data, since neighboring connectivity is obtained from the cell-edge data structure. Equation (24) is the exact ratio of the second-order term to the first-order one on a uniform grid. Because the ratio indicates an approximate solution error and its exact value is unnecessary, it is directly extended to arbitrary quadrilateral grid.

There are three parameters in the criterion. α is initially designed to prevent a zero denominator, and it can also filter extremely small flow variations. For example, if the amplitude of the variations in some regions is much less than $\alpha\tilde{\rho}_{i,j}$, then the error indicators are also small. Consequently cells will not be refined there. Three parameters are nearly independent of flow conditions to achieve the most economical adaptation. We choose $\varepsilon_r = 0.06$, $\varepsilon_c = 0.05$, and $\alpha = 0.03$ for all computations in this paper.

The refinement and coarsening procedures are handled separately. Both procedures have similar steps for vectorization:

1. handling the inside of cells which are flagged to be refined or coarsened;
2. handling the edges of the flagged cells;
3. arranging memory.

Step 1 is based on cells and updates all inside information, such as deleting inside edges and adding finer cells; these are done without changing the status of other outside cells. Step 2, based on edges, renews every edge of refined or coarsened cells and its two neighboring cells, which may be done without influencing other edges. Following this strategy, one may naturally remove the data dependency which often prevents vectorization in adaptation.

A cell to be refined must be divided into four subcells or *sons*, as shown in Fig. 7. If an edge is split into two subedges, we call the edge a *mother* and the subedges *daughters*.

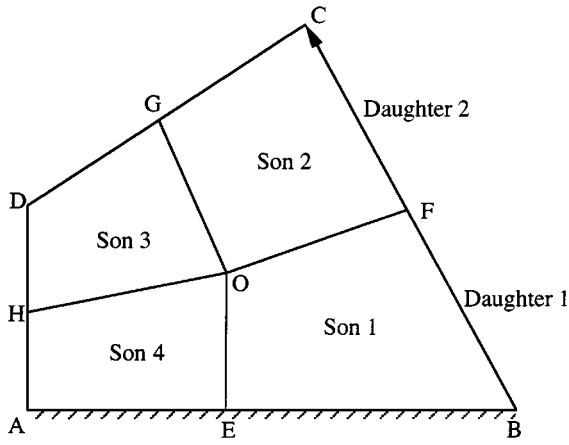

FIG. 7. Refinement strategy: father cell ABCD is divided into four sons, EBFO, OFCG, HOGD, and AEOH.

There are some options in the division. One is that the node inside can be optimized to get good shapes of the sons. In the present adaptation the node is set to the centroid of the father cell. Another option is the locations of new nodes on the outside edges, usually at the centers. However, if an edge is a boundary, AB, for instance, we choose a new location E to make edge OE perpendicular to the boundary if this choice does not generate extremely skewed cells. Only when all four sons are flagged *coarse* are they deleted and recovered to their father cell.

In the refinement procedure, it is required that no two neighboring cells differ by more than one refinement level. This rule prevents pathologically large volume ratios under certain circumstances. An illustration of this rule is shown in Fig. 8. If cell ABCD is one of the coarsest cells, its level is set to 0. The refinement level of a son equals that of his father plus 1. In Fig. 8, the level of cells 1, 2, 3, and 4 is 1; the level of cells 5, 6, 7, and 8 is 2, and so on. The difference between levels of two neighboring cells may not be more than 1. Refining cell 5 is then not allowed. The rule of one-level difference has also been used by other authors (see, for example, [22]).

In unsteady calculations, the rule that two neighboring cells differ by no more than one level may mismatch moving refined regions, say shock wave regions. An illustration is given in Fig. 9. The refinement levels of cells in columns *a* and *b* are 3; those in columns *c* and *d*

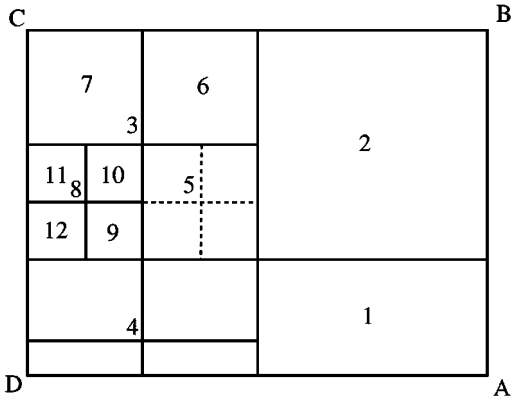

FIG. 8. The levels of refinement: refining cell 5 is not allowed.

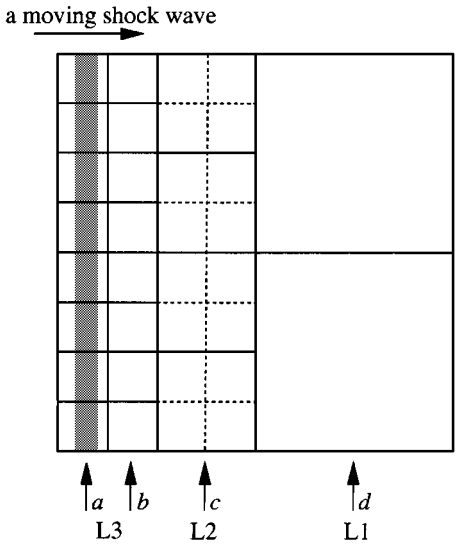

FIG. 9. Prerefinement. Cells in column *c* cannot be refined to capture the coming shock wave, because the level of refinement in column *d* is lower. The cells in column *d* have to be prerefined.

are 2 and 1, respectively. A shock wave moving from left to right lies in the finest cells. The shock wave will arrive at column *c* in two time steps by using a high CFL number; then the cells in column *c* should be refined to be able to capture the shock wave. Since a shock wave can be resolved within two cells by a high-resolution scheme, cells in column *d* are not able to detect that the shock is coming based only on the information from its closest neighboring cells. The adaptation criterion then labels the cells in column *d*, *refine* = *.false.*, and thus column *c* cannot be refined because of the rule. Consequently the shock wave moves into the region with coarser cells and is smeared dramatically. This indeed happened in our early unsteady calculations. To combat this problem, *prerefinement* is introduced. Once a cell cannot be refined due to the level difference between itself and its neighboring cells, the neighboring cells should be prerefined. In the case shown in Fig. 9, the cells in column *d* are prerefined, no matter what they are labeled.

Steps 1 and 2 change the status of some cells and edges, for example from sons to fathers. Step 3 just cleans and arranges the memory, and divides all cells and all edges into two groups, father/non-father and mother/non-mother, according to their updated status. The non-mother edges are further divided into boundary/non-boundary edges. These classifications generate an efficient data structure for a flow solver. Figure 10 illustrates memory

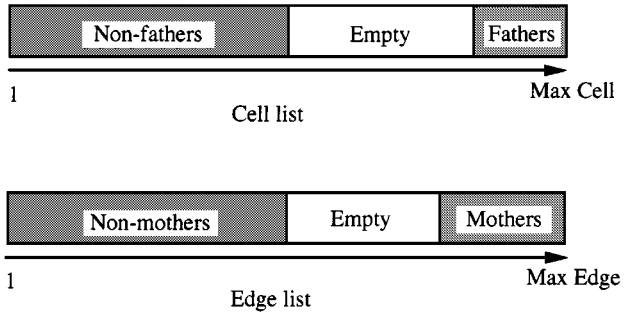

FIG. 10. Memory organization.

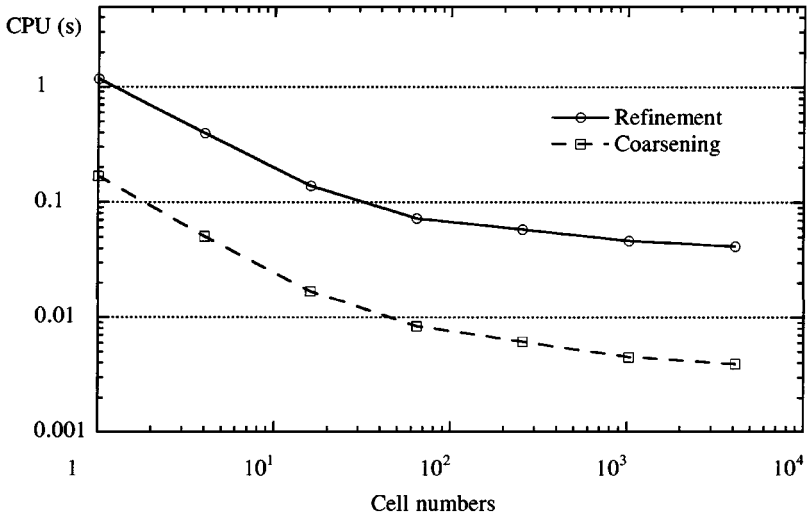

FIG. 11. The efficiency of vector processing of refinement and coarsening procedure.

strides of the edge list and the cell list after the adaptation procedure. All cell indices are saved in the cell list, while edge indices in the edge list. Both of them are constant strides, which is required for vectorization. The lists are very efficient especially for a flow solver using the finite volume method, because non-fathers are just non-overlapping physical control volumes and non-mothers are the interfaces on which flux evaluations are conducted.

The adaptation subroutine includes 34 loops and about 1,400 FORTRAN lines. All loops are vectorized. Its practical efficiency for vector processing is tested by refining one cell from level 0 to level 6 (4096 cells), and then coarsening the refined cells from level 6 to level 0. Figure 11 shows the average CPU time normalized by 10,000 cells. No flow solver is computed in this test. The CPU time is decreased with increasing total cell numbers. When the total cells are over 1,000, the speedup is about 30. This indicates that the adaptation procedure is well vectorized.

4. THE FLOW SOLVER

In the flow solver, we combine the smoothing step and the space- or operator-splitting convection step as

$$U^{n+2} = L_x L_y S L_y L_x S U^n,$$

where S represents the smoothing step. $L_x L_y$ and $L_y L_x$ represent the convection step. Two steps will be explained in Sections 4.1 and 4.2, respectively. The use of this Strang-type splitting on the quad-tree grid can be found in [4]. Note that each time step conducts the smoothing once, which is more computationally efficient than smoothing twice in two directions. If the adaptation is used, it is always conducted before the smoothing step. The vector processing of the flow solver under the cell-edge data structure will be discussed in Section 4.3.

4.1. *The Conservative Smoothing Step*

The conservative smoothing is required to dampen oscillations generated in solving Euler equations by the Lax–Wendroff scheme. In this subsection, we will describe how to realize

this smoothing on an unstructured quadrilateral grid. In the smoothing step, we add artificial dissipation by solving

$$U_t = \frac{(\Delta l)^2}{\Delta t} \nabla \cdot (\varepsilon^* \cdot \nabla \tilde{U}),$$

where U represent the conservative variables, and Δl is a local grid size. ε^* is a diagonal tensor of artificial viscosity. Following the finite volume approach, one gets

$$U_i = \tilde{U}_i + \frac{1}{\text{Vol}} \sum \varepsilon_{ij} (\tilde{U}_j - \tilde{U}_i) l_j l_{ij} \mathbf{l_{ij}} \cdot \mathbf{n_j}, \tag{25}$$

where l_j, l_{ij} are the length of an interface and the distance between two neighboring cells of the interface, $\mathbf{n_j}$ and $\mathbf{l_{ij}}$ are their unit vectors, respectively. A sketch of the geometry variables is shown in Fig. 12a. Vol is the volume of cell i, and $\sum$ sums the fluxes through four faces. The local grid size Δl has been taken as l_{ij} in (25). The performance of a smoothing step just depends on the control of artificial viscosity ε_{ij} at the interface.

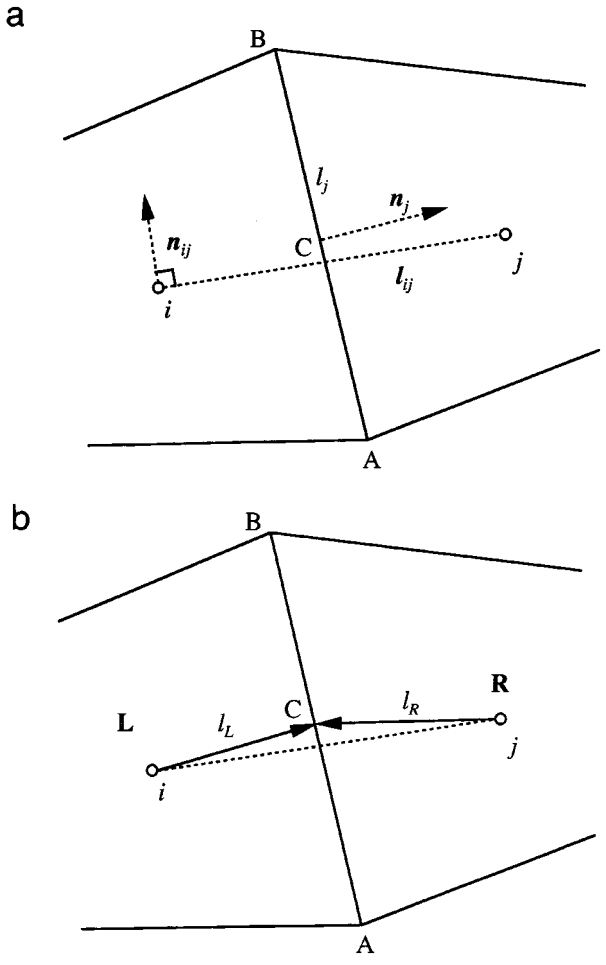

FIG. 12. Geometrical definition of an interface.

We first consider some requirements on a smoothing step in multidimensional flows. There are five basic physical phenomena in compressible flows: shock wave, expansion waves, contact surfaces, slipstreams, and vortices.[2] The first three exist in one-dimensional and multidimensional flows. They are locally close to be one-dimensional in multidimensional flows. The smoothing step needs only to dissipate them along their propagation direction, so that the most effective dissipation $\varepsilon_2 = 1/4$ can be applied. However, the last two are truly multidimensional phenomena. Although both slipstream and contact surface are discontinuous in numerical simulations, contact surfaces will possibly generate spurious oscillations, but a slipstream usually does not because of its relatively low normal speed and thus low fluxes. In other words, a contact surface needs artificial viscosity, but a slipstream does not. Furthermore, a solid boundary is similar to a slipstream, where the normal speed is zero but the tangential speed is arbitrarily large. Low artificial dissipation at boundary is critical for the calculation of the boundary layer. Another multidimensional phenomenon is related to a vortex. The core of a vortex has a physical minimum value of density and pressure. Conservative smoothing should be able to distinguish a vortex core from these unwanted extrema.

In brief, acceptable conservative smoothing should add sufficient dissipation around spurious oscillating regions in primary wave propagation direction, while it should not influence other smooth regions, vortices, and slipstreams. The present smoothing consists of two steps: first detecting oscillations and sharp gradients; then computing artificial viscous fluxes.

In detecting oscillations, we design indicators

$$\begin{aligned} T_X &= \mathrm{Min}[\phi_i(\tilde{\rho}), \phi_i(\rho\tilde{e}), \mathrm{Max}(\phi_i(\tilde{u}), \phi_i(\tilde{v}))], \\ T_Y &= \mathrm{Min}[\phi_j(\tilde{\rho}), \phi_j(\rho\tilde{e}), \mathrm{Max}(\phi_j(\tilde{u}), \phi_j(\tilde{v}))], \end{aligned} \tag{26}$$

where the functions $\phi_i(\varphi)$ and $\phi_j(\varphi)$ are defined as

$$\phi_i(\varphi) = \frac{|\varphi_{i+1,j} + \varphi_{i-1,j} - 2\varphi_{i,j}|}{\varepsilon_0\tilde{\rho}_{i,j} + |\varphi_{i+1,j} - \varphi_{i-1,j}|}, \qquad \phi_j(\varphi) = \frac{|\varphi_{i,j+1} + \varphi_{i,j-1} - 2\varphi_{i,j}|}{\varepsilon_0\tilde{\rho}_{i,j} + |\varphi_{i,j+1} - \varphi_{i,j-1}|}, \tag{27}$$

with $\varepsilon_0 = 10^{-4}$. T_X and T_Y are indicators in two directions, as shown in Fig. 6b.

Compared with the density indicator in the 1-D smoothing step (23), the speeds in two directions are added in (26) to distinguish a vortex from a spurious oscillation. This is based on the understanding that the velocity in a vortex is usually free of extrema although the core of it has a minimum of density. We suppose that these additional limitations have negligible influence on detecting spurious oscillations, or spurious oscillations are the simultaneous variation of velocity, density, and energy. If we use only the density in the indicator, as done in the 1-D calculations, the smoothing step can input excessive dissipation to a vortex and produce a "hollow" vortex core, as shown in Fig. 13a. This is verified by the indicator (26), as shown in Fig. 13b, where the spiral shape of the slipstream and the vortex core are clearly captured.

[2] Slipstream and contact surface satisfy the same compatible conditions; that is, across them both pressure and the normal velocity are continuous. We distinguish them qualitatively by the difference of tangential speed across them. If the difference is small, it is a contact surface; if the difference is large, it is a slipstream. In experiment a slipstream is usually much thicker than a contact surface because of strong viscosity.

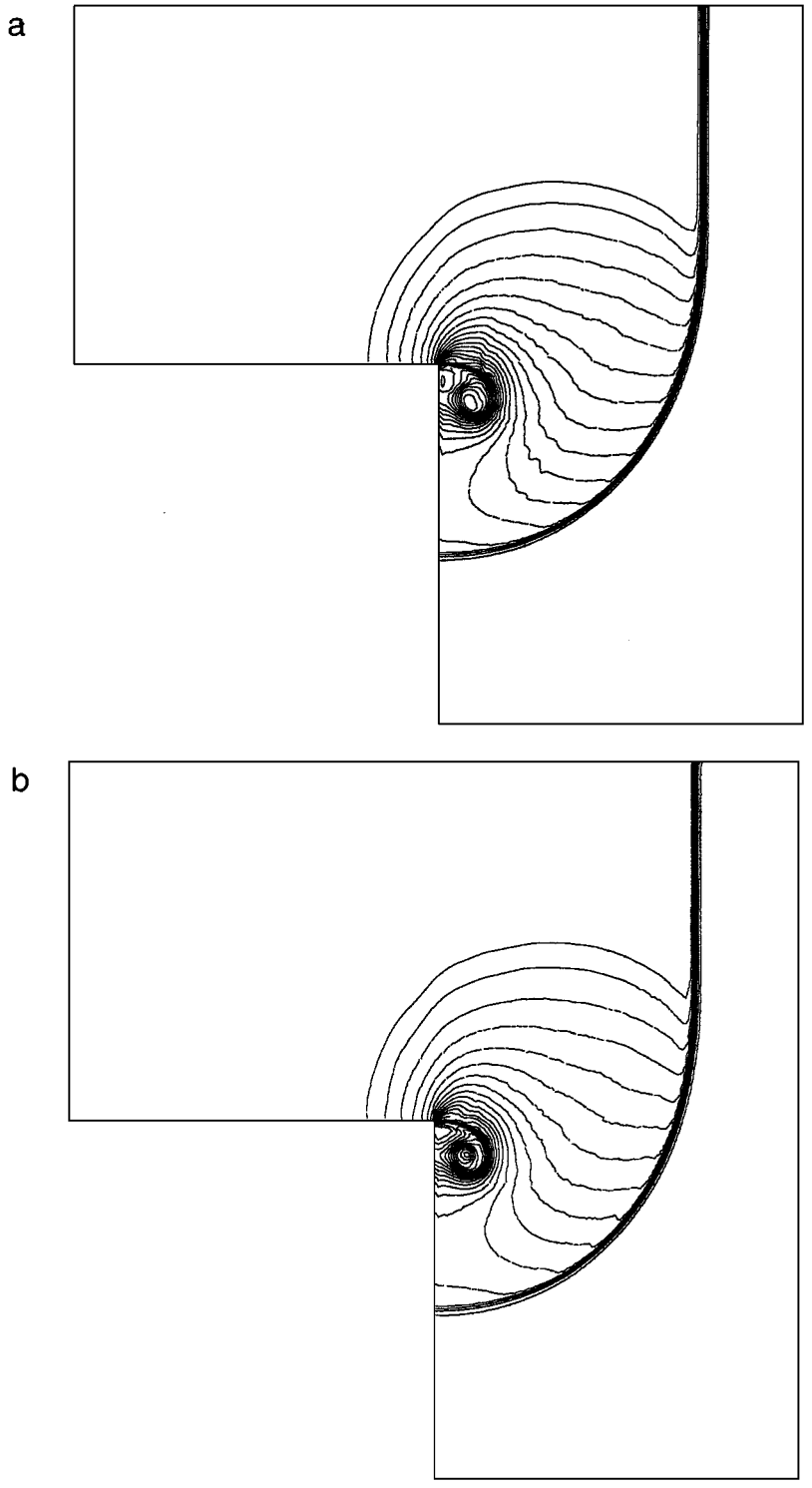

FIG. 13. Comparison between two indicators: shock diffraction over a 90° corner, $M_s = 1.5$. (a) 1-D density indicator: a hollow vortex; (b) indicator (26).

In order to add the dissipation only in oscillation directions, we define a vector artificial coefficient

$$\varepsilon_{\mathbf{i}} = \psi(\varepsilon_2, 0, T_X \geq \varepsilon_1)\mathbf{n_X} + \psi(\varepsilon_2, 0, T_Y \geq \varepsilon_1)\mathbf{n_Y}, \tag{28}$$

where ψ is defined as

$$\psi(x, y, z) = \begin{cases} x, & \text{if } z = .true. \\ y, & \text{if } z = .false. \end{cases} \tag{29}$$

Symbols $\mathbf{n_X}$ and $\mathbf{n_Y}$ are unit vectors, whose directions are shown in Fig. 6b. The criterion (5) has been applied in (28) along two directions. In this way, the dissipation will be added only in oscillation directions. Function ψ in (29) is a vectorizable intrinsic function.

After oscillations and their directions are detected for all cells, the viscous coefficient at interface is given by

$$\varepsilon_{ij} = W_t(|\varepsilon_{\mathbf{i}} \cdot \mathbf{n_j}| + |\varepsilon_{\mathbf{j}} \cdot \mathbf{n_j}|)/2, \tag{30}$$

where W_t is a weight function. The weight is defined according to a local wave direction,

$$W_t = \frac{(d\mathbf{v_{ij}} \cdot \mathbf{n_j})^2 + \varepsilon_0/2}{d\mathbf{v_{ij}} \cdot d\mathbf{v_{ij}} + \varepsilon_0}, \tag{31}$$

where $d\mathbf{v_{ij}}$ is velocity difference between cell i and j which is approximately the direction of wave propagation. The component along $\mathbf{n_j}$ of the velocity difference is estimated as the weight.

We note that if $\mathbf{n_j}$ is just the direction of wave propagation, then $W_t = 1$ (ε_0 is negligibly small). It is clear that the artificial dissipation is the same as that in solving 1-D Euler equations in the direction of wave propagation. Therefore one may expect that for the locally one-dimensional phenomena spurious oscillations are able to be dampened, as done in one-dimensional flows shown in Section 2. On the other hand, the weight (31) also shows that artificial dissipation across a slipstream is very small, because the change of normal velocity $(d\mathbf{v_{ij}} \cdot \mathbf{n_j})^2$ is much less than the total change $d\mathbf{v_{ij}} \cdot d\mathbf{v_{ij}}$ there. In the algorithm $W_t = \text{Max}[1/8, W_t]$ is adopted. The coefficient ε_{ij} at any boundary is always set to be zero.

4.2. *The Convection Step*

In solving 2-D Euler equations on a structured grid, the two-step Lax–Wendroff scheme designed by Zwas (see [10, p. 259]) consists of three nodes in each dimension, which is a stencil closest to the domain of physical wave propagation. However the scheme requires both cell-centered and cell-vertex addressings. It is very difficult for an unstructured data structure especially with grid adaptation to efficiently provide two addressings together. In the present paper the space-splitting method is chosen to avoid the cell-vertex addressing, while the computational stencil used is still three in each dimension.

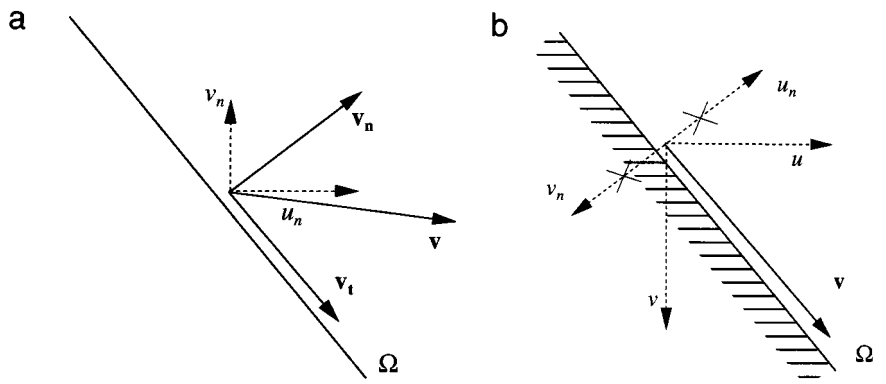

FIG. 14. Splitting of velocity at interface.

Two-dimensional conservation laws are

$$\frac{\partial}{\partial t}\int_{\text{Vol}} U dV + \int_{\Omega} F(un_x + vn_y)\, dl + \int_{\Omega} (P_x n_x + P_y n_y)\, dl = 0, \tag{32}$$

where

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{pmatrix}, \qquad F = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho e + p \end{pmatrix}, \qquad P_x = \begin{pmatrix} 0 \\ p \\ 0 \\ 0 \end{pmatrix}, \qquad P_y = \begin{pmatrix} 0 \\ 0 \\ p \\ 0 \end{pmatrix}.$$

We write the Euler equations in this form for its convenience in describing the splitting method. Let $\mathbf{v_n}$ and $\mathbf{v_t}$ denote the normal and the tangential velocity of $\mathbf{v} = (u, v)$ on interface Ω as shown in Fig. 14a. Then one may rewrite the second term in (32), as

$$un_x + vn_y = \mathbf{v} \cdot \mathbf{n} = (\mathbf{v_n} + \mathbf{v_t}) \cdot \mathbf{n} = \mathbf{v_n} \cdot \mathbf{n} = u_n n_x + v_n n_y, \tag{33}$$

then

$$\begin{aligned} &\frac{\partial}{\partial t}\int_{\text{Vol}} U dV + \int_{\Omega} F(u_n n_x + v_n n_y)\, dl + \int_{\Omega} (P_x n_x + P_y n_y)\, dl = 0; \\ &\frac{\partial}{\partial t}\int_{\text{Vol}} U dV + \int_{\Omega} (F u_n + P_x)\, dx + \int_{\Omega} (F v_n + P_y)\, dy = 0, \end{aligned} \tag{34}$$

where u_n, v_n are two components of $\mathbf{v_n}$ in the Cartesian coordinates. Equation (34) is just the conservative equations used for splitting. We split it into two equations

$$\frac{\partial}{\partial t}\int_{\text{Vol}} U dV + \int_{\Omega} (F u_n + P_x)\, dx = 0 \tag{35}$$

and

$$\frac{\partial}{\partial t}\int_{\text{Vol}} U dV + \int_{\Omega} (F v_n + P_y)\, dy = 0. \tag{36}$$

Transformation (33) is required to avoid spurious fluxes through a solid boundary which is not parallel to the coordinates as shown in Fig. 14b. Flows always move along the boundary, and the velocity normal to the boundary is zero. However, after one splits the velocity directly according to the coordinates, both u and v have non-zero components normal to the boundary, although the sum of them equals zero. Then, in solving the split equation, the non-zero component introduces fluxes through the boundary, which is physically incorrect. After the transformation, the splitting is based on the normal velocity which is zero at boundary, so the spurious fluxes disappear.

Equation (35) is discretized at half time step,

$$U^{n+1} = U^n - \frac{\Delta t}{\mathrm{Vol}} \sum \left(F^{n+1/2} u_n^{n+1/2} + P_x^{n+1/2} \right) \Delta x_j. \tag{37}$$

The values at the half time step are predicted by locally computing the Lax–Friedrichs scheme between two neighboring cells. The state at the interface center C is taken as the distance-weighted one from two cells,

$$U_C = \frac{l_R}{l_R + l_L} U_i + \frac{l_L}{l_R + l_L} U_j, \tag{38}$$

where l_R and l_L are distances from cells to the edge center C, as shown in Fig. 12b. The Lax–Friedrichs scheme is then solved along direction $\mathbf{l_{ij}}$ by assuming the flow is approximately one dimensional there,

$$\begin{aligned}
\rho_C^{n+1/2} &= \rho C + \Delta\tau \left[(\rho u_l)^L - (\rho u_l)^R \right] \\
(\rho u_l)_C^{n+1/2} &= (\rho u_l)_C + \Delta\tau \left[(\rho u_l)^L u_l^L + p^L - (\rho u_l)^R u_l^R - p^R \right] \\
(\rho u_n)_C^{n+1/2} &= (\rho u_n)_C + \Delta\tau \left[(\rho u_n)^L u_l^L - (\rho u_n)^R u_l^R \right] \\
(\rho e)_C^{n+1/2} &= (\rho e)_C + \Delta\tau \left[(\rho e)^L u_l^L + P^L u_l^L - (\rho e)^R u_l^R + p^R u_l^R \right],
\end{aligned} \tag{39}$$

where superscript letters ^{L} and ^{R} denote the states at cell i and j, respectively. u_l and u_n are components of velocity in directions $\mathbf{l_{ij}}$ and $\mathbf{n_{ij}}$, as shown in Fig. 12a, for instance

$$\begin{aligned}
(\rho u_l)^L &= (\rho \mathbf{v})_i \cdot \mathbf{l_{ij}} \\
(\rho u_n)^L &= (\rho \mathbf{v})_i \cdot \mathbf{n_{ij}},
\end{aligned} \tag{40}$$

and so on. The predicator step should choose $\Delta\tau = (1/2)\Delta t / l_{ij}$. We modify $\Delta\tau$ as follows,

$$\Delta\tau = \Delta t / l_{ij} \begin{cases} 1/2, & T_{ij} \le 1/2 \\ T_{ij}, & 1/2 < T_{ij} \le 3/4, \\ 3/4, & 3/4 \le T_{ij} \end{cases} \tag{41}$$

or in a concise form,

$$\Delta\tau = \mathrm{Max}[1/2, \mathrm{Min}(3/4, T_{ij})] \Delta t / l_{ij},$$

where T_{ij} is an indicator of flow variation, and $T_{ij} = \mathrm{Max}(|\mathbf{T_i} \cdot \mathbf{n_j}|, |\mathbf{T_j} \cdot \mathbf{n_j}|)$ where $\mathbf{T_i} = T_X \mathbf{n_X} + T_Y \mathbf{n_Y}$. This modification generally introduces more dissipation around sharp discontinuities. The smoothing step combined with this modified Lax–Wendroff scheme works

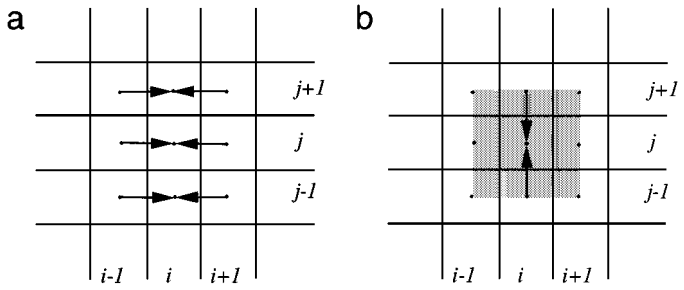

FIG. 15. Computational stencil for the splitting method.

better. Since the modification keeps $\Delta_\tau = 1/2\Delta t$ in continuous regions, it does not change the accuracy there. The fluxes through the interface are then given by using the predicted values.

Equation (36) can also be discretized similarly. Let operators L_x and L_y represent solving two equations; then in one time step we have either $U^{n+1} = L_y L_x U^n$ or $U^{n+1} = L_x L_y U^n$.

It is observed that the splitting method requires three nodes in each dimension on a regular grid. This is explained in Fig. 15 by considering $L_y L_x$ splitting. Step L_x collects the information along x-axis as shown in Fig. 15a. Then step L_y further collects all information from y-direction. Thus the stencil is finally three in each dimension as the shadow region shown in Fig. 15b.

Special treatments are needed to keep second order accuracy in space for interfaces between cells with different refined levels. *Pseudo-cells* are created for cells with a lower refined level, as shown in Fig. 16. Cells A and C have different refined levels. The coarser one, cell A, should generate some pseudo-cells, say *a* and *b*. Pseudo-cells *a* and *b* will have the same level as cell C. The conservative values of pseudo-cells are interpolated from their neighboring cells. For example these of pseudo-cell *a* are interpolated from cells A, B, and C. All fluxes are evaluated between same level cells. The flux between cells A and C is computed by cell C and pseudo-cell *a*, but the flux between cells A and B is still computed by these two cells. Without these pseudo-cells some tiny fluctuations would appear on contours. Illustrations are given in Fig. 13, which are early computations without pseudo-cells.

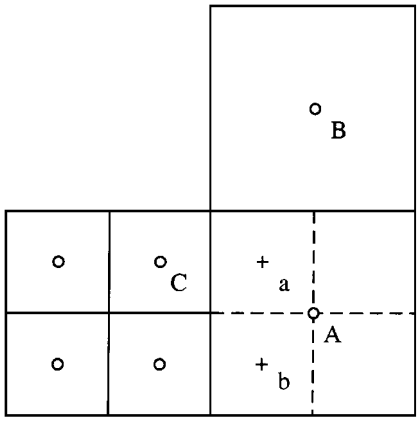

FIG. 16. Illustration of pseudo-cells.

4.3. *Flow Solver under the Cell-Edge Data Structure*

The flow solver can be efficiently constructed and easily vectorized under the cell-edge data structure. This is done following a routine which consists of four basic steps:

1. computing fluxes through non-mother non-boundary edges;
2. computing fluxes through non-mother boundary edges;
3. computing fluxes through mother edges by adding their daughters' fluxes;
4. computing the sum of four fluxes for every cell.

The computation of fluxes through an edge requires only its two neighboring cells whose indices are explicitly saved for every edge, so that the flux evaluation is easily vectorized. Similar efficiency is also achieved for other steps by the cell-edge data structure.

It is clear that adjacent connectivity information can be directly obtained from the cell-edge data structure. However, to get the same information, the well-known quad-tree and octree structure often need to climb up to the root of the branch and then climb down to neighboring cells, which is difficult, if not impossible, to vectorize. The climbing process is avoided by using the present cell-edge data structure. Note that because of step 3, step 4 simply accumulates the fluxes of four edges irrespective of whether the edges are split or not while preserving conservation.

The flow solver following the four steps has almost no conditional statements in determining connectivity information, which is usually very difficult for locally adaptive algorithms. In fact, the only conditional statement is in step 2, where it deals with different types of boundaries. Thus, the data structure is highly efficient for solving the conservation laws.

The global efficiency of all subroutines is measured by calculating a shock wave diffracting over a 90° corner. The test runs 400 time steps and uses about 10,000 cells (cell number cannot be fixed because of adaptation). The memory requirement for this computation is nearly 2 megawords. The numerical results will be discussed in the following section.

Figure 17 shows the CPU time distribution on the main subroutines of the algorithm. The flow solver takes 70% of the computational time. The remaining time is mainly spent

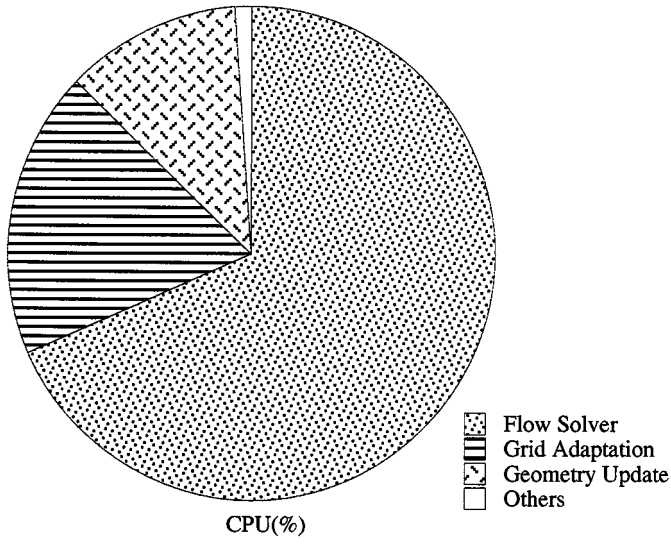

FIG. 17. CPU time distribution on major subroutines.

on grid adaptation and geometry update. The geometry update only computes cell volumes and edge lengths. It costs, of course, a very small portion of the total computation. The flow solver takes 6 times as much CPU time as that required by the geometry update, and the grid adaptation only takes 1.7 times. These results show that both the flow solver and the grid adaptation are computationally fast. According to hardware statistics, the average vector length is 117.6 (the maximum machine length is 128), and therefore the flow solver and the adaptation are well vectorized.

The average CPU time requires about $4\,\mu$s per cell per step. This speed is around two orders faster than that conducted in a workstation using upwind schemes with either exact or approximate Riemann solvers [8, 23]. The efficiency is not only due to the well-vectorized data structure but also due to the Euler solver, which is a central-difference scheme with the conservative smoothing. (Note: All computations are carried out on a Cray C90 in single processor mode. The flow solver tested here does not include pseudo-cells which have been discussed in Section 4.2. Pseudo-cells increase computer time up to 50% depending on the number of pseudo-cells.)

5. NUMERICAL EXAMPLES

In this section a few unsteady and steady gas dynamic problems are solved by the scheme presented in the previous sections. The ideal gas model and the ratio of specific heats $\gamma = 1.4$ are used. The parameter $\varepsilon_1 = 0.7$ and the artificial viscous coefficient $\varepsilon_2 = 1/4$ are constant in all computations. Other parameters, for instance, the thresholds for refinement and coarsening shown in Section 3, are also kept constant. Although fine tuning of these parameters can improve the resolution and efficiency a little, it will be demonstrated that these unchanged parameters give acceptable results for a variety of unsteady and steady problems.

5.1. *Unsteady Shock Diffraction*

Shock wave diffraction over a 90° corner is conducted in this subsection. The geometry consists of three 1×1 squares. Every square can be divided to many fine cells. A shock wave is initially at 0.5 to the left of the corner point. The CFL number is 0.9, and it is unchanged for unsteady computations. A discontinuity may pass one cell in one time step at such a high CFL number, so that the adaptation is performed at every time step.

Figure 18 gives a weak shock diffraction at three different levels of refinement. The density contours are virtually independent of the grid adaptation. Of course, shock waves become linearly sharper for high-level adaptations. Because the postshock flow is subsonic, expansion waves propagate upstream. The incident shock wave is gradually attenuated to the downstream wall. These phenomena are resolved similarly on three different grids.

The recorded computer time further shows that the adaptation procedure is highly efficient. The finest cells are economically distributed around shocks and vortices. In this example an additional level of refinement requires about 3.2 times as much CPU time, but it is 8 times (4 times cells, and 2 times time steps) as much on a uniformly refined grid. Although the adaptive unstructured algorithm runs a few times slower than a structured one, a two-level refinement more than compensates for this factor.

The results of strong shock diffraction, for $M_s = 2$, 4, 8, 16, are shown in Fig. 19. Real gas effects are not considered here, because the main object of this paper is not to investigate

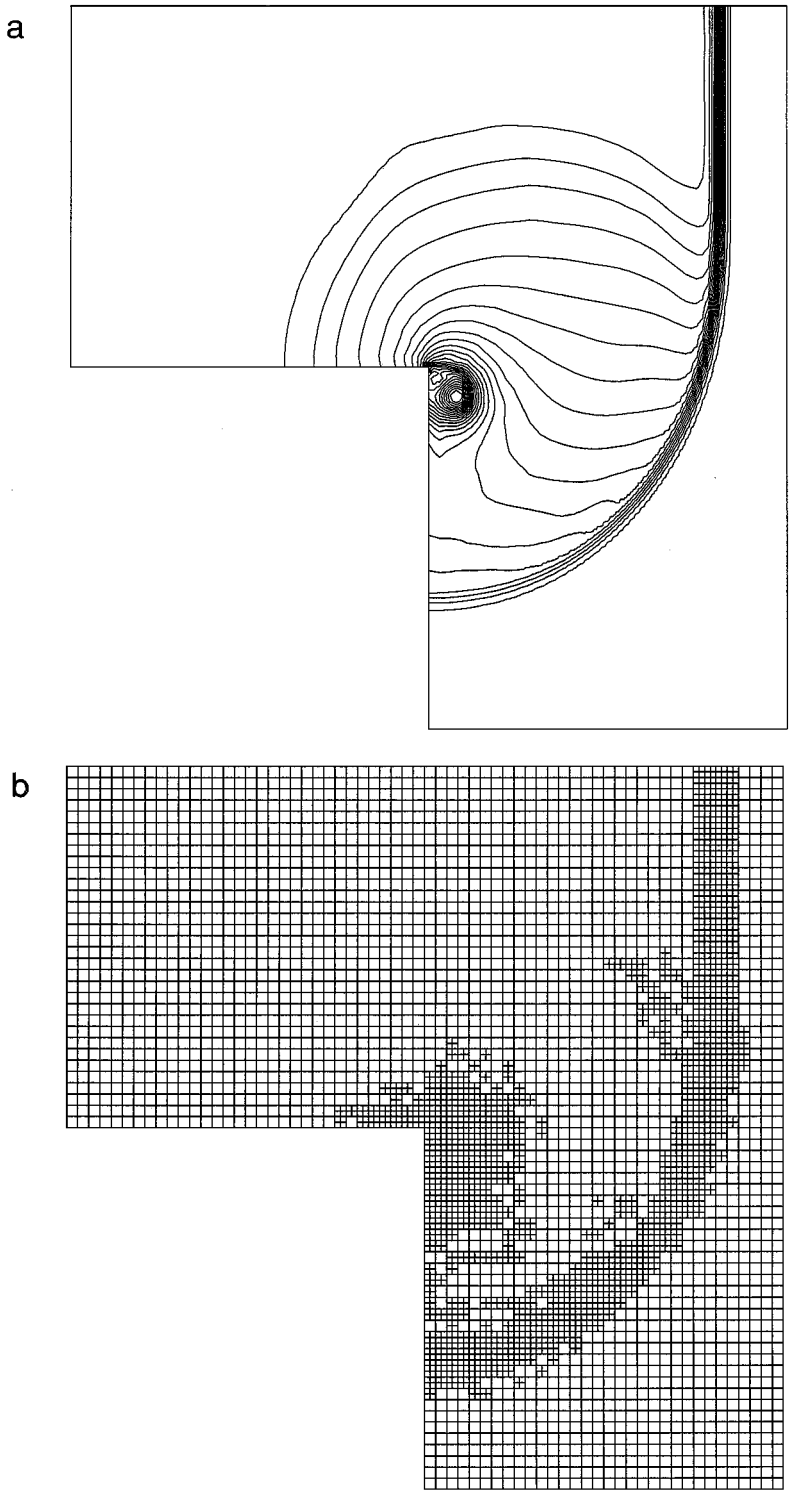

FIG. 18. Unsteady shock diffraction on adaptive grids with different levels of refinement, isopycnics and grids: $M_s = 1.3$, CFL = 0.9. (a, b) level = 1, 4,683 cells, CPU = 2.0 s; (c, d) level = 2, 6,744 cells, CPU = 6.2 s; (e, f) level = 3, 11,382 cells, CPU = 20.4 s.

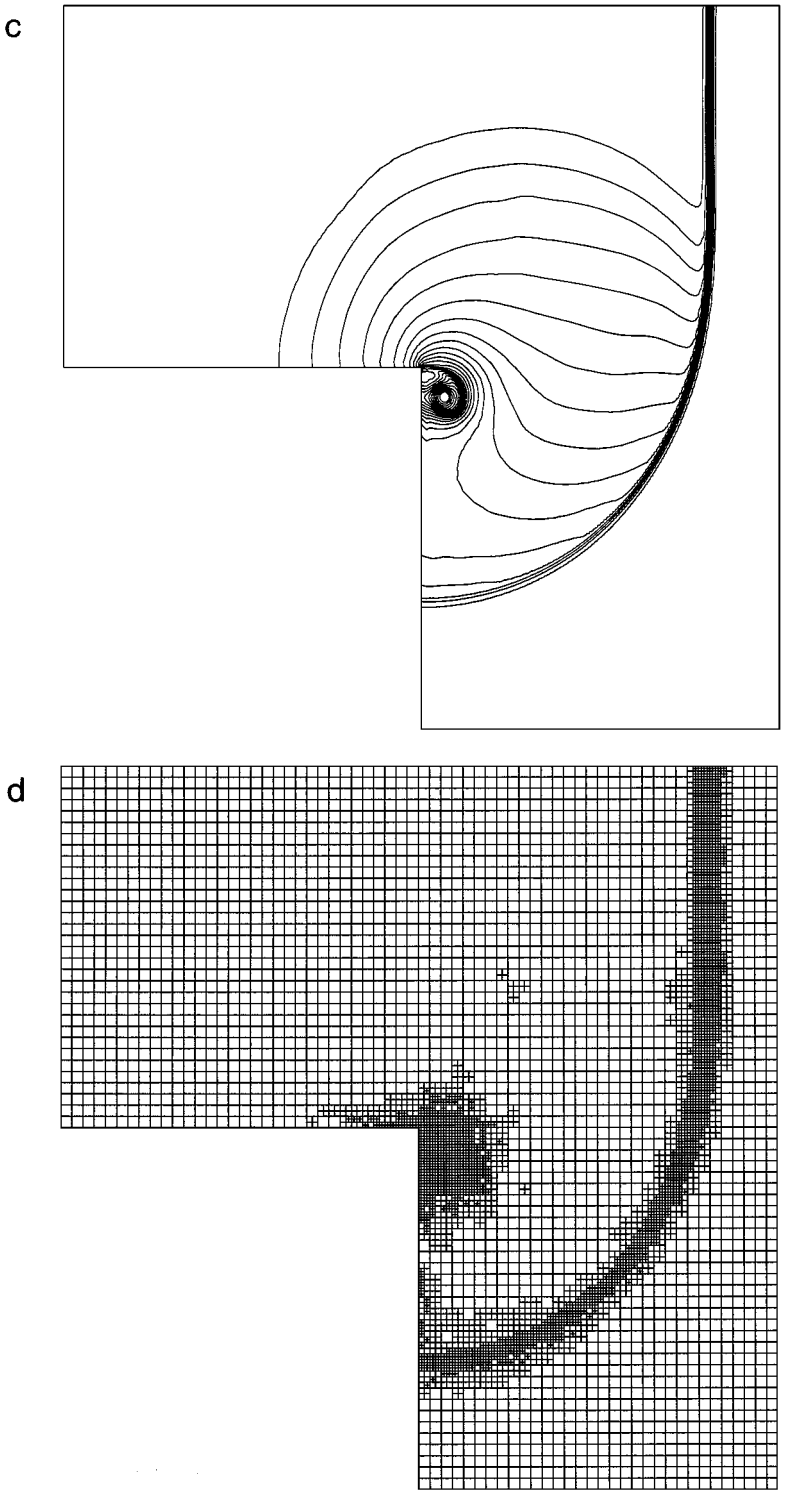

FIG. 18—*Continued*

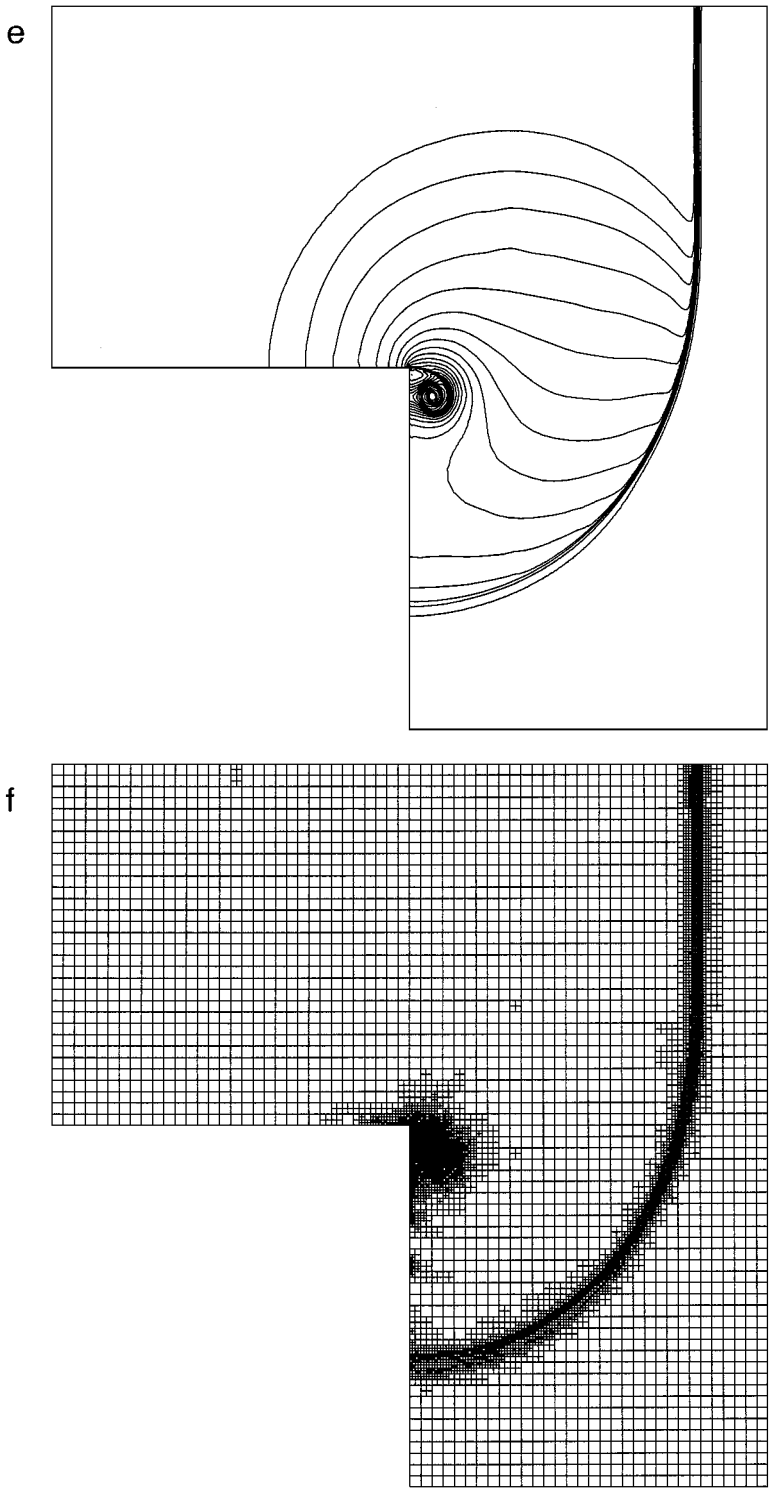

FIG. 18—*Continued*

physical phenomena. Three-level refinement is adopted, and it is $256 \times 256 \times 3$ cells if using a uniform grid. The required CPU time is no more than 35 s in each case. We emphasize here the two parameters $\varepsilon_1 = 0.7$ and $\varepsilon_2 = 1/4$ are kept constant in calculating these shock waves. Combined with the results of weak shock diffraction shown in Fig. 18, it is clear that the smoothing step efficiently cleans the oscillations behind shock waves with a wide range of strengths.

For strong shock diffraction, flow patterns close to the corner are similar. There are expansion waves, a slipstream and a secondary shock wave. These are well resolved for all shock strengths. The vortex is not observable for very strong shocks. A clear difference between different shocks lies in their reflection configurations on the downstream wall. The foot of the $M_s = 2$ shock is almost perpendicular to the wall; the $M_s = 4$ shock forms a sharp turn there, but the waves behind it are somewhat continuous. The waves at the wall for the $M_s = 8$ case can be recognized as a sharp discontinuity or a reflected shock wave, and then they form a Mach reflection and a very short Mach stem is seen; for the $M_s = 16$ shock the Mach stem is no longer visible and the configuration is a regular reflection. These changes qualitatively agree with experimental observations [2].

5.2. *Unsteady Shock Reflection*

Another basic unsteady shock phenomenon is shock reflection. A benchmark test is set for a shock wave with $M_s = 2$ moving over a 46° wedge. A variety of numerical results by different schemes and a few experimental photos are available in [19]. We repeat the computation under the same condition. Figure 20 shows our numerical results on structured and unstructured quadrilaterals.

The computation uses a 4-level refinement and its finest cells correspond to a 512×512 grid which covers the whole computational domain for the structured grid. The CFL number is 0.9, but the time step is somewhat limited by the smallest cells near the corner. Around 1000 time steps are required and the CPU time is less than 1 min.

The wave configuration consists of an incident shock, a Mach stem, a reflected shock, and a slipstream. These are clearly seen in the results on both grids. The slipstream has been captured as sharply as shock waves. Shock thickness is not constant in the figures because the background cells are of different sizes, as shown in Fig. 20b. The resolution of shocks is comparable with many other schemes [19], but the present computation requires much less computer time.

5.3. *Steady Channel Flows*

Steady flows are computed in a channel with a compression corner, followed by an expansion corner, which is similar to that in [23]. The length of computation domain is 4. Solutions are given at $t = 4$. The upper and lower surfaces of the channel are reflecting conditions, with a supersonic inflow on the left, and outflow on the right. The inflow Mach number $M = 2$. The CFL number can be chosen slightly higher than 1 in steady computation. We compared the results computed at lower CFL numbers, and found no big differences in pressure and density contours, but an increase in computer time. The grid adaptation is performed every three time steps. When using a refined grid, we calculate for $t \leq 2$ on a coarse grid to get a good initial state, then start the adaptation procedure and compute up to $t = 4$. Computing time is reduced by nearly half using this simple strategy.

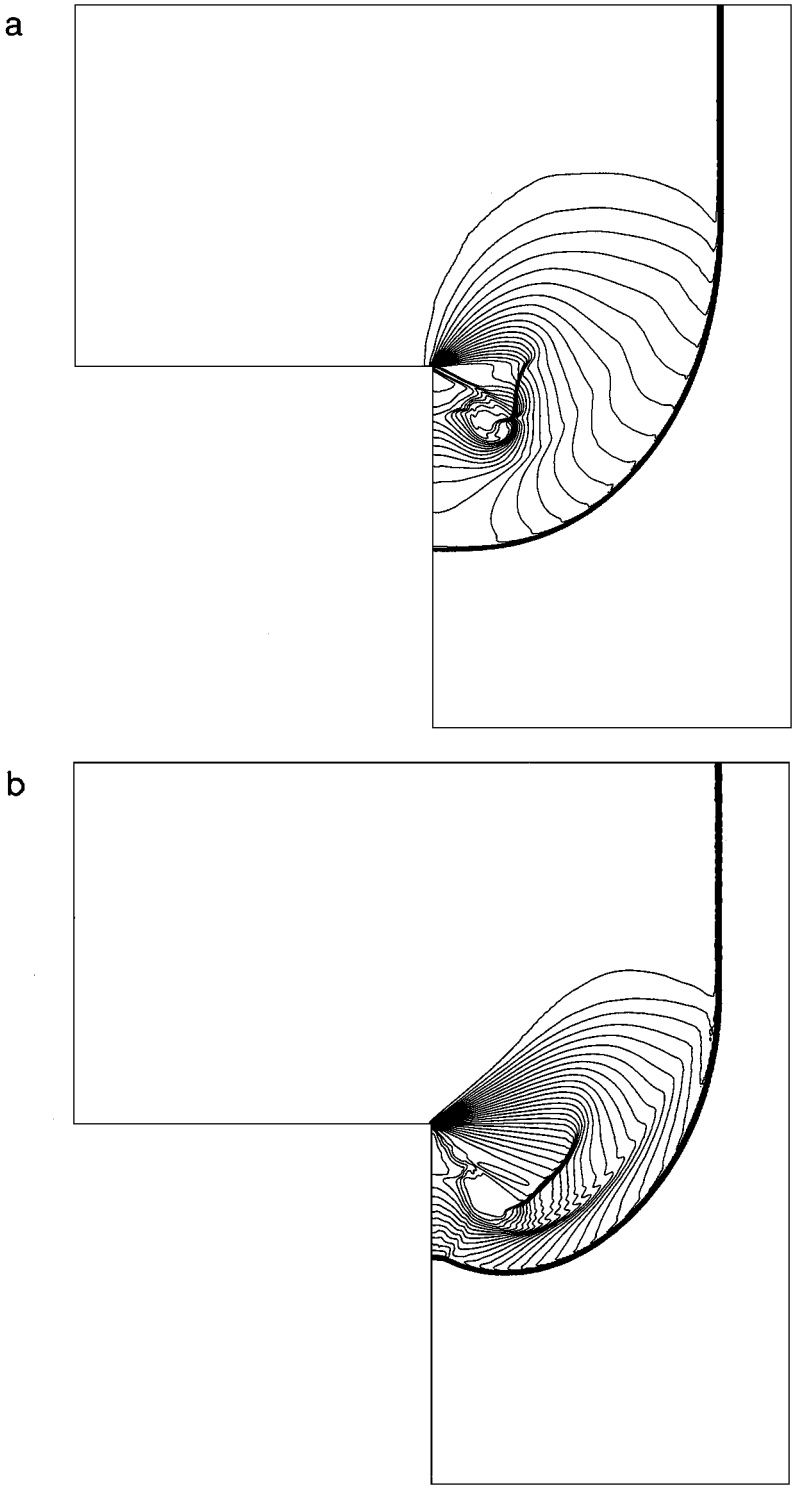

FIG. 19. Unsteady shock diffraction for strong shock waves, isopycnics: CFL = 0.9, level = 3, 20,000–24,000 cells, CPU = 30–35 s. (a) $M_s = 2.0$; (b) $M_s = 4.0$; (c) $M_s = 8.0$; (d) $M_s = 16.0$.

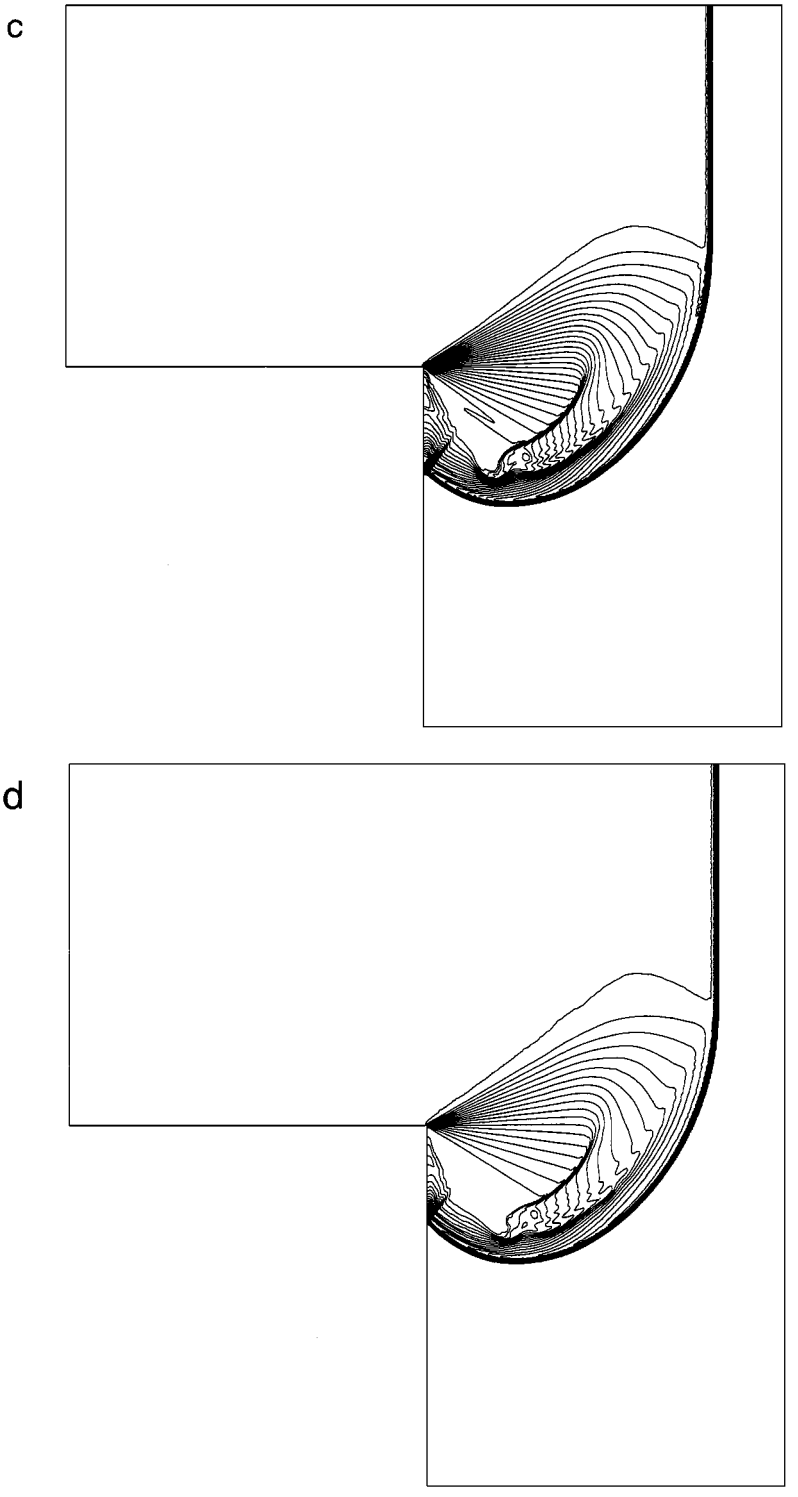

FIG. 19—*Continued*

Figure 21 shows the results using three-level refinement. There is an attached shock at the compression corner. The shock is reflecting from the upper surface and forms a Mach reflection. The reflected shock interacts with the Prandtl–Meyer fan starting from the corner and then reflects from the lower surface as well. There is a discontinuity in the isopycnics, emanating from the triple point and nearly going parallel to the upper surface, as shown in Fig. 21a. This discontinuity is interpreted as being a slipstream because the pressure across it is continuous, as shown in Fig. 21b.

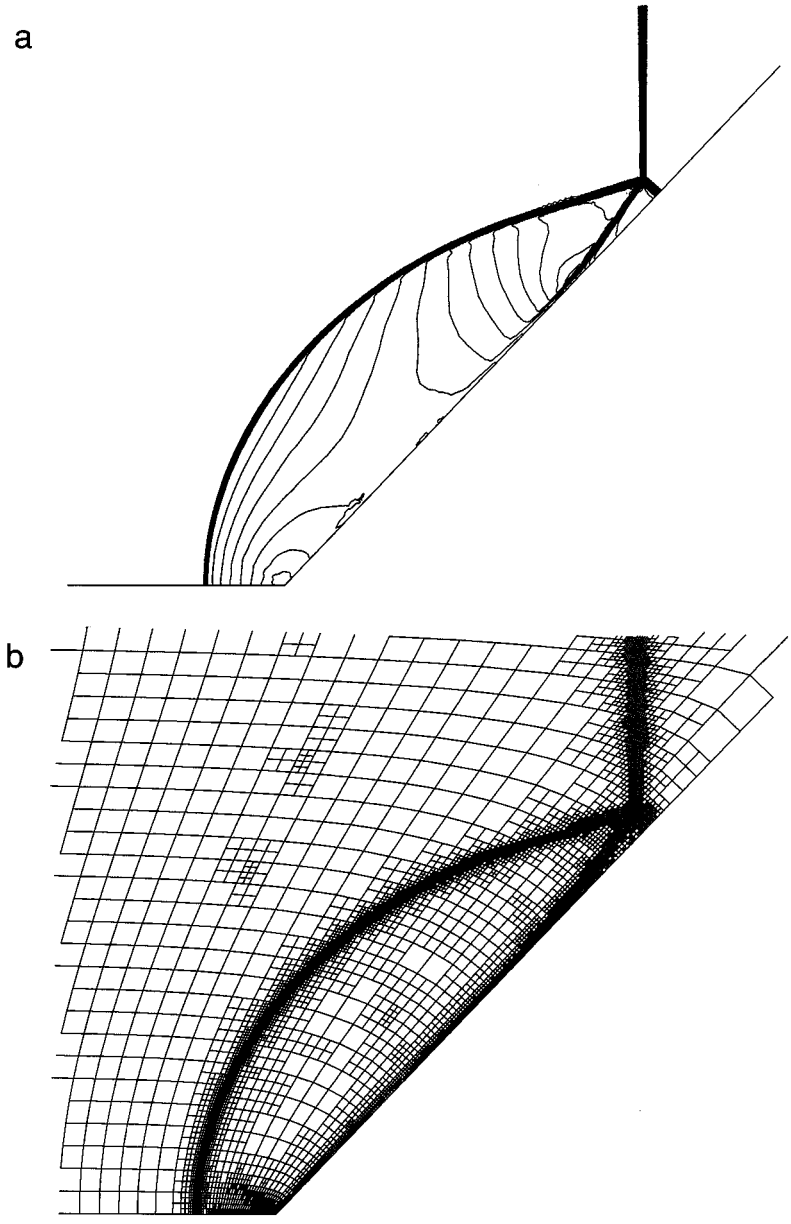

FIG. 20. Unsteady shock reflection over a 46° wedge for $M_s = 2$, isopycnics and grids: CFL = 0.9, level = 4, 22,000–23,000 cells. (a, b) on a structured grid; (c, d) on an unstructured grid.

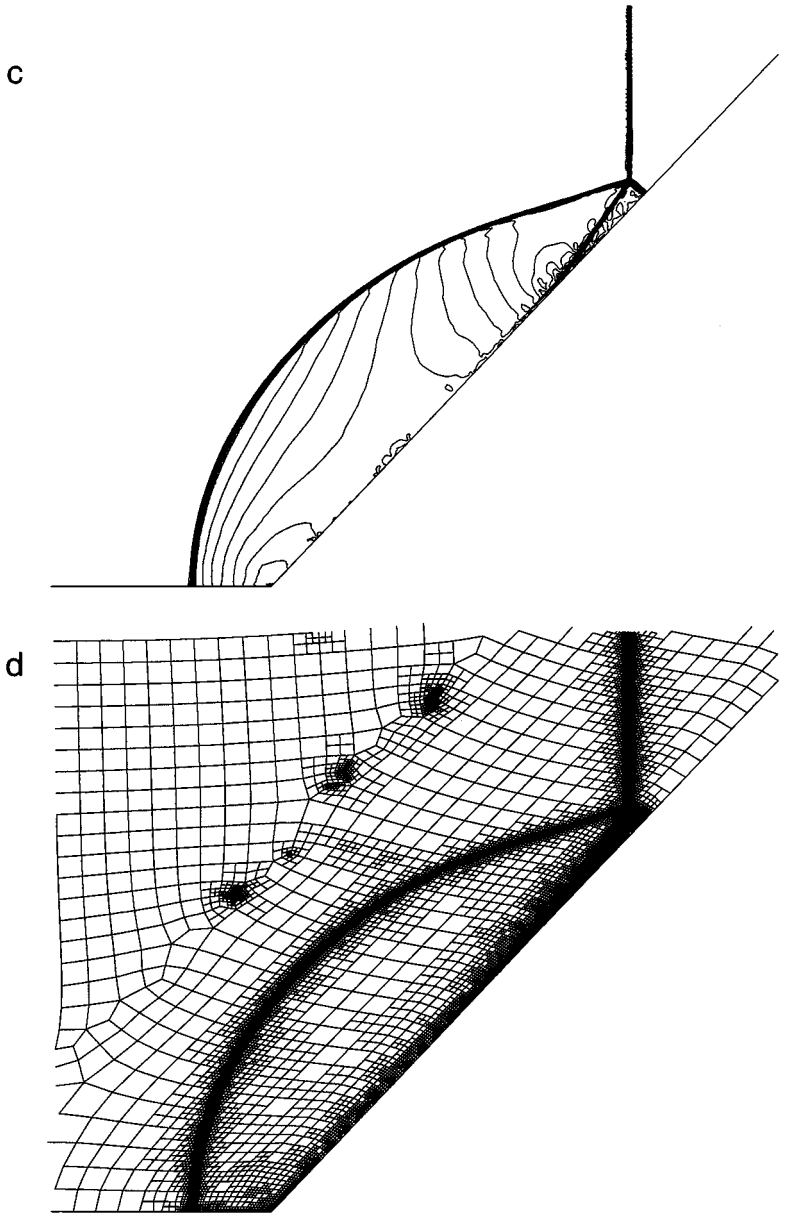

FIG. 20—*Continued*

A shortcoming of the smoothing or filter in computing unsteady [6] and steady flows [5] is the fluctuation of contours. The present approach has successfully overcome this. It is seen that both density and pressure contours are sufficiently smooth in expansion wave regions even on a grid with different sized cells.

5.4. *Shock Motion in a Circular reflector*

Previous results are either steady flows or unsteady but self-similar flows. This subsection gives a truly unsteady result of shock wave moving in a circular reflector. The geometry

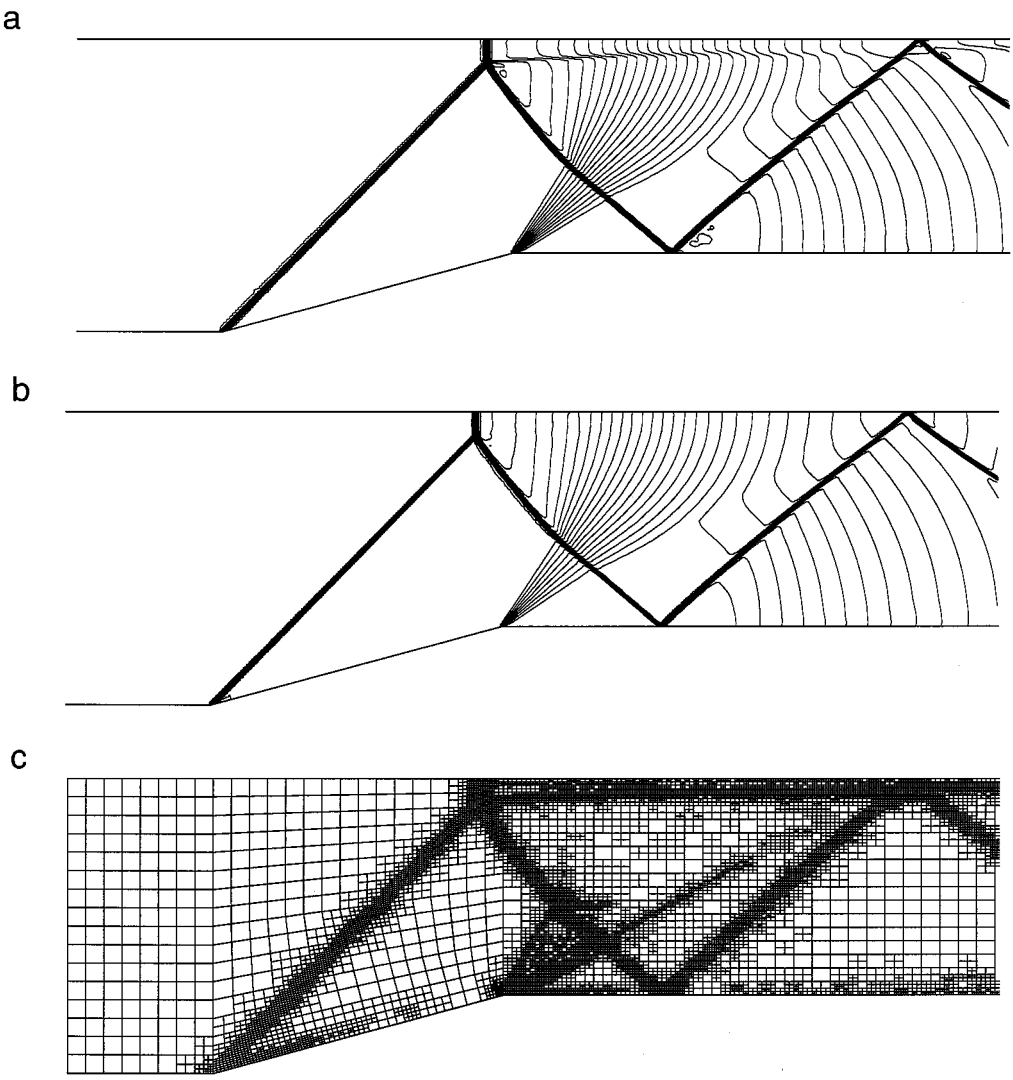

FIG. 21. Steady tunnel flow over a 15° bump on an adaptive grid: $M = 2.0$, CFL = 1.6, level = 3, 13,861 cells, 1,273 time steps, CPU = 61.7 s. (a) isopycnics; (b) isobars; (c) corresponding grid.

of the reflector and initial grid are shown in Fig. 22. Only half of the domain is computed because of symmetry.

The results are shown in Fig. 23 for incident shock Mach number $M_s = 1.5$. The initially planar shock wave enters and diffracts at the entrance, resulting in a curved shock wave, the foot of which is perpendicular to the wall, as seen in Fig. 23a. The foot of this curved shock wave tends to lean forward with its propagation and eventually becomes Mach reflection, as seen in Fig. 23b. With further propagation of the shock wave, the Mach reflection transits to a regular reflection. A fully developed regular reflection is shown in Fig. 23c. In Fig. 23d the incident shock has been completely reflected. It is seen that the adaptive grids are efficiently distributed around time-dependent sharp changing regions.

For quantitative comparison, the density distribution along the centerline is plotted in Fig. 24 at an instant when experimental data are available [18]. The numerical data are compared with those obtained by another adaptive flow solver [7], which is performed

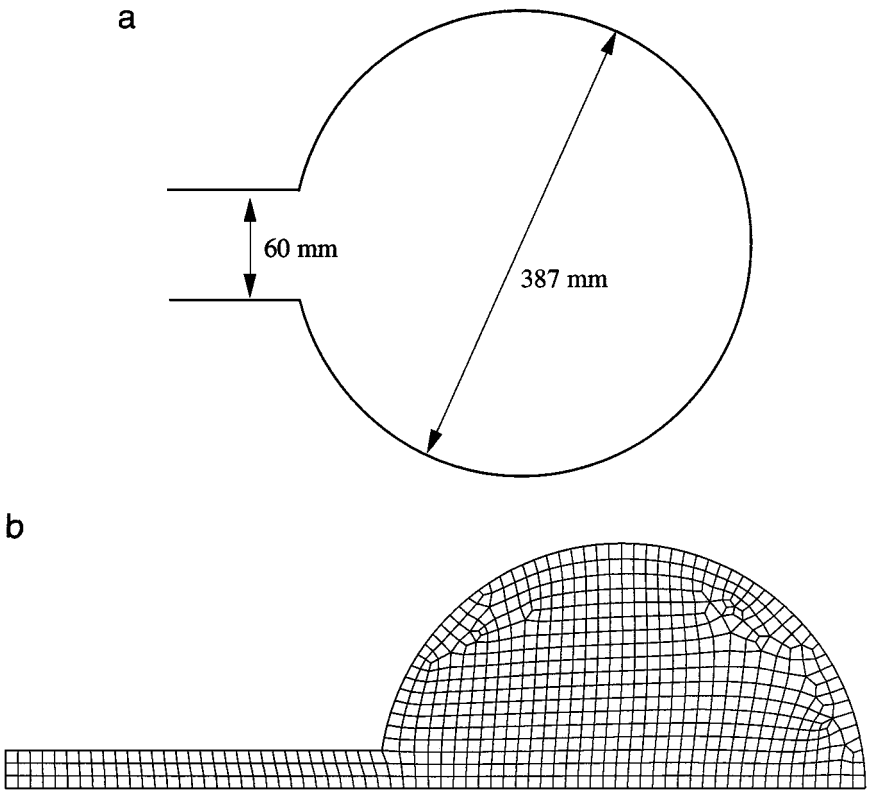

FIG. 22. Physical geometry and unstructured quadrilateral grid.

on triangular grid using a second-order Godunov scheme. It is seen that the present result agrees well with experimental data, and the accuracy is similar to the second-order Godunov scheme.

6. CONCLUDING REMARKS

This paper shows that the nonlinear limited smoothing step efficiently removes the oscillations generated by the Lax–Wendroff scheme in solving two-dimensional Euler equations. The approach has been invariably applied to simulate both unsteady and steady flows, for weak- and strong-shocked flows. The approach has been coupled with a vectorized locally adaptive algorithm using quadrilateral cells, so that the efficiency is enhanced considerably further.

Compared with the popular upwinding schemes, two merits of the smoothing approach are worth mentioning. One is that the approach can be applied easily to solve other hyperbolic systems because no Riemann solver is involved, and also it is simpler and easier to implement than the upwind schemes.

We finally remark that the approach still has a large scope for future work. First, the present convection step uses a one-dimensional approximation as done by the Riemann solvers, due to the limited information (two cells) which an interface can efficiently accesses on an unstructured grid. However, a second-order central difference scheme is actually able to be truly multidimensional. This suggests that the flux through interface can be determined without any one-dimensional approximation. A truly multidimensional solver should be

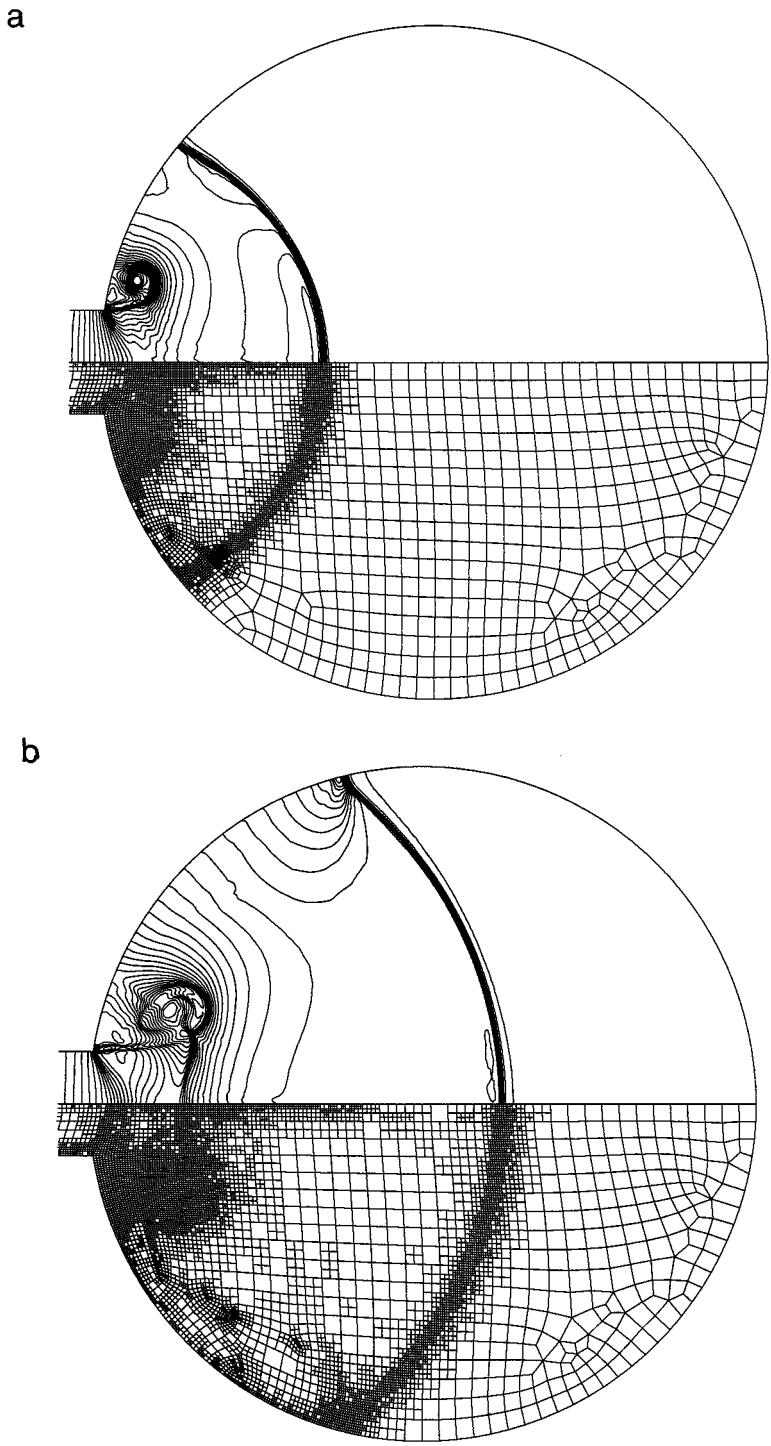

FIG. 23. Sequential numerical isopycnics and adaptive grids of shock wave propagation in the circular reflector for $M_s = 1.5$.

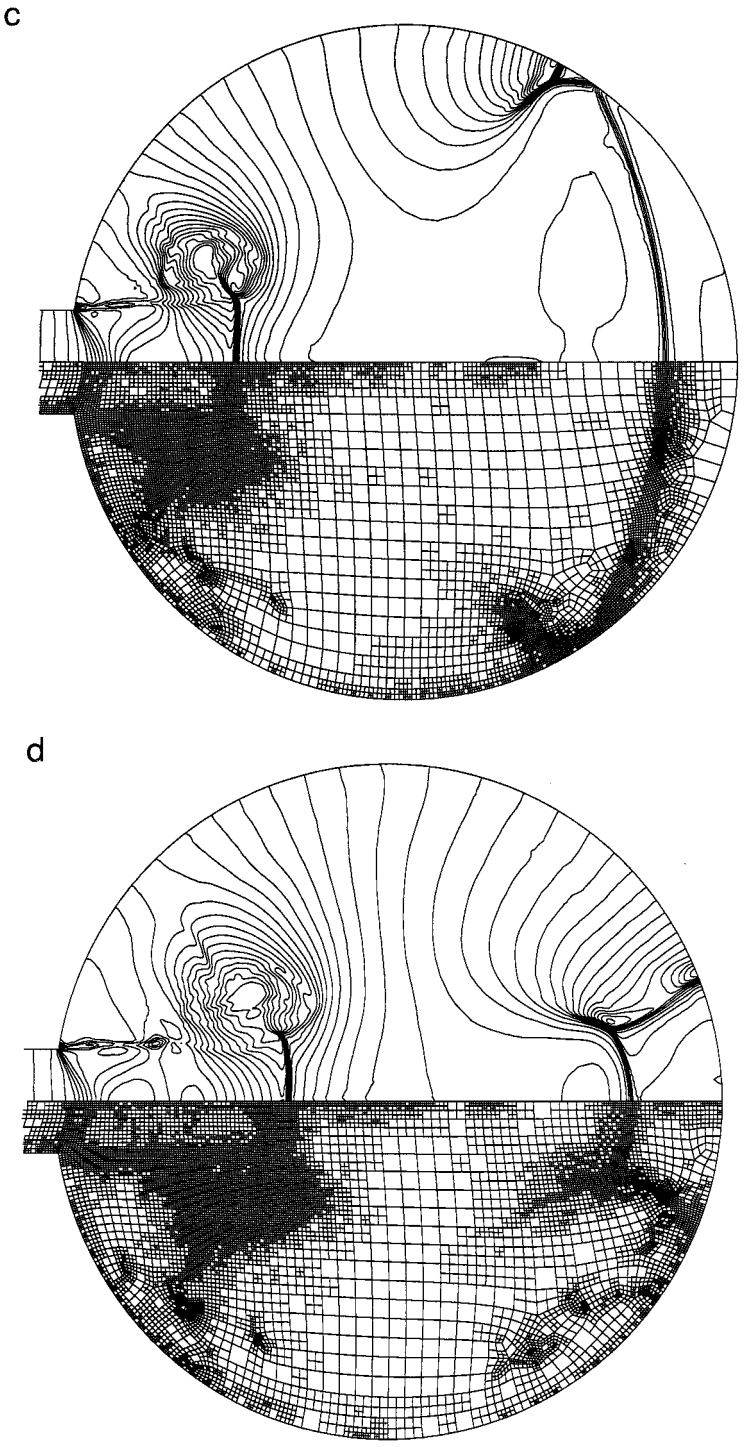

FIG. 23—*Continued*

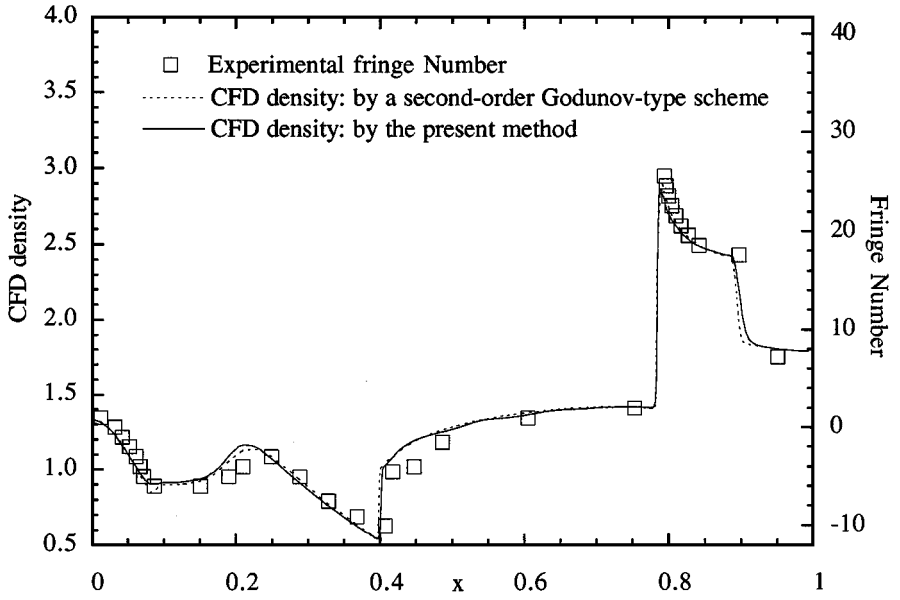

FIG. 24. Comparison of density distributions along the centerline.

constructed for the unstructured grid by using more information, for instance, gradient of flow variables.

Second, unlike most upwind schemes which incorporate implicit artificial dissipation, the present approach uses an explicit one. The explicit dissipation is relatively easily controlled for interested flow phenomena, such as slipstreams and vortices. This has been shown to be the case in the present work. It opens a promising way to switch off artificial viscosity in the boundary layer when solving the Navier–Stokes equations. The performance of the approach on the solution of the boundary layer will be investigated and compared with the popular upwind schemes in the near future.

APPENDIX

We will prove here that the sufficient and necessary condition of a monotonic sequence $u_1, u_2, \ldots, u_i, \ldots, u_n$ is that, for any $1 < i < n$,

$$\phi_i = \frac{|u_{i+1} + u_{i-1} - 2u_i|}{\varepsilon_0 + |u_{i+1} - u_{i-1}|} < 1,$$

where ε_0 is an infinitely small positive number.

Proof. Let $u_{i+1/2} = u_{i+1} - u_i$ and $u_{i-1/2} = u_i - u_{i-1}$; the statement above is identical to the sufficient and necessary condition that all members of sequence

$$u_{1/2}, \ldots, u_{i-1/2}, u_{i+1/2}, \ldots, u_{n-1/2}$$

have the same sign or are equal to 0, is that, for any $1 < i < n$,

$$\phi_i = \frac{|u_{i+1/2} - u_{i-1/2}|}{\varepsilon_0 + |u_{i+1/2} + u_{i-1/2}|} < 1,$$

where ε_0 is an infinite small positive number.

On the one hand, if the $u_{i+1/2}$ and $u_{i-1/2}$ have the same sign or are equal to 0, then

$$\phi_i = \frac{||u_{i+1/2}| - |u_{i-1/2}||}{\varepsilon_0 + |u_{i+1/2}| + |u_{i-1/2}|},$$

so

$$\phi_i < 1.//$$

On the other hand, given $\phi_i < 1$, we will show that nonzero $u_{i+1/2}, u_{i-1/2}$ with different signs is not true. If it is true,

$$\phi_i = \frac{|u_{i+1/2}| + |u_{i-1/2}|}{\varepsilon_0 + ||u_{i+1/2}| - |u_{i-1/2}||},$$

so

$$\phi_i \geq 1,$$

which violates the given condition $\phi_i < 1$. So $u_{i+1/2}, u_{i-1/2}$ have the same sign or are equal to zero. ■

ACKNOWLEDGMENTS

We express our thanks to Dr. Z. Jiang, Dr. E. Timofeev, Dr. T. Saito, Professor P. Voinovich, Professor J. Falcovitz, and J. Schumacher for their discussions and criticisms of the present work.

REFERENCES

1. A. Aoyagi and K. Abe, Runge–Kutta smoother for suppression of computational-mode instability of leapfrog scheme, *J. Comput. Phys.* **93**, 287 (1991).
2. T. V. Bazhenova, L. G. Gvozdeva, and Y. V. Zhilin, Change in the shape of the diffracting shock wave at a convex corner, *Acta Astronautica* **6**, 401 (1979).
3. J. P. Boris and D. L. Book, Flux-corrected transport. I.SHASTA, a fluid transport algorithm that works, *J. Comput. Phys.* **11**, 38 (1973).
4. M. G. Edwards, J. T. Oden, and L. Demkowicz, An h-r adaptive approximate Riemann solver for the Euler equations in two dimensions, *SIAM J. Sci. Comput.* **14**, 185 (1993).
5. B. Engquist, P. Lotstedt, and B. Sjogreen, Nonlinear filters for efficient shock computation, *Math. Comput.* **52**, 509 (1989).
6. A. A. Fursenko, D. M. Sharov, E. V. Timofeev, and P. A. Voinovich, Numerical simulation of shock wave interactions with channel bends and gas nonuniformities, *Comput. Fluids* **21**, 377 (1992).
7. A. A. Fursenko, N. P. Mende, K. Oshima, D. M. Sharov, E. V. Timofeev, and P. A. Voinovich, Numerical simulation of propagation of shock waves through channel bends, *Comput. Fluid Dyn. J.* **2**, 1 (1993).
8. L. Giraud and G. Manzini, Parallel implementations of 2D explicit Euler solvers, *J. Comput. Phys.* **123**, 111 (1996).
9. V. P. Goloviznin, A. I. Zhmakin, and A. A. Fursenko, A numerical method for the investigation of discontinuous flows of relaxing mixtures, *Sov. Phys. Dokl.* **27** (1982).
10. C. Hirsch, *Numerical Computational of Internal and External Flows, Vol. 2* (Wiley, New York, 1990).
11. Y. Kallinderis and A. Vidwans, Generic parallel adaptive-grid Navier–Stokes algorithm, *AIAA J.* **32**, 54 (1994).
12. P. D. Lax and B. Wendroff, Systems of conservation laws, *Comm. Pure Appl. Math.* **13**, 217 (1960).

13. P. L. Roe, Modern shock-capturing schemes, in *Proc. 18th ISSW, Sendai*, edited by K. Takayama (Springer-Verlag, Berlin/New York, 1991), p. 29.
14. G. H. Schmidt and F. J. Jacobs, Adaptive local grid refinement and multi-grid in numerical reservoir simulation, *J. Comput. Phys.* **77**, 140 (1988).
15. B. K. Shivamoggi, *Theoretical Fluid Dynamics* (Nijhoff, Dordrecht, 1985).
16. W. Shyy, M. H. Chen, R. Mittal, and H. S. Udaykumar, On the suppression of numerical oscillations using a nonlinear filter, *J. Comput. Phys.* **102**, 49 (1992).
17. G. S. Sod, A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws, *J. Comput. Phys.* **27**, 1 (1978).
18. M. Sun and K. Takayama, A holographic interferometric study of shock wave focusing in a circular reflector, *Shock Waves* **6**, 323 (1996).
19. K. Takayama and Z. Jiang, Shock wave reflection over wedges: A benchmark test for CFD and experiments, *Shock Wave* **7**, 191 (1997).
20. V. Venkatakrishnan, A perspective on unstructured grid flow solvers, AIAA paper 95-0667, 1995.
21. J. von Neumann and R. D. Richtmyer, A method for the numerical calculations of hydrodynamic shocks, *J. Appl. Phys.* **21**, 232 (1950).
22. D. P. Young, R. G. Melvin, and M. B. Bieterman, A locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics, *J. Comput. Phys.* **92**, 1 (1991).
23. D. D. Zeeuw and K. G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, *J. Comput. Phys.* **104**, 56 (1993).