Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 226 (2007) 1550–1588

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Towards front-tracking based on conservation in two space dimensions II, tracking discontinuities in capturing fashion

Mao De-kang *,1

Department of Mathematics, Shanghai University, Shanghai 200444, PR China

Received 10 April 2006; received in revised form 24 May 2007; accepted 1 June 2007
Available online 14 June 2007

Abstract

In this paper, the second one in the series beginning with [D. Mao, Towards front tracking based on conservation in two space dimensions, SIAM J. Sci. Comput. 22 (1) (2000) 113–151], we study another important feature of our 2D conservative front-tracking method, i.e. discontinuity curves in two space dimensions are tracked in a 1D capturing fashion. The evolution of 2D discontinuity curves are locally described by 1D conservation laws with source terms, which are derived from the governing equations. The front-tracking in our method is then realized by numerically simulating these 1D conservation laws with source terms in a conservative fashion. In this paper, our 2D front-tracking method is described in details, which is Cartesian-grid-based, conservative and much simpler in algorithm than other 2D front-tracking methods. The discussion starts with the 1D case, which facilitates the following 2D discussion. Data structure of the numerical solutions and first- and second-order versions of our 2D front-tracking method are described. Finally, numerical examples for both scalar equations and the Euler system of gas dynamics in 2D are presented to show the efficiency and effectiveness of the method.
© 2007 Elsevier Inc. All rights reserved.

AMS subject classification: 65M05; 65M10; 35L65

Keywords: Conservation laws; Front-tracking; Discontinuity cell; Cartesian-grid-based

1. Introduction

We are concerned with the nonlinear hyperbolic systems of conservation laws

$$u_t + \sum_{i=1}^{m} f_i(u,x)_{x_i} = 0, \tag{1.1}$$

* Tel.: +86 21 66134464; fax: +86 21 66133239.
E-mail address: dkmao@staff.shu.edu.cn
1 Research is supported by China NSF Grant No. 10171063 and Shanghai Pu Jiang Program [2006] 118.

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.06.004

where u and $f_i(u, x)$ can be either scalars or vectors, with $m = 1$ or 2. For many years, we have been developing a front-tracking method based on the philosophy of the Lax–Wendroff theorem [16] for the systems, i.e. tracking discontinuities by enforcing the conservation properties of solutions. The research can be traced back to [30,31]. The progress of the development and the efficiency and effectiveness of the methods in 1D and 2D were reported in [22,24,26–29,25,20,21,12,13]. At present, the 1D method has been completed. An all-purposed and robust algorithm has been built and tested on a variety of numerical examples, and the results were excellent [20,21,12,13].

The real challenge comes from the 2D and higher-dimensional cases. Beginning with [24], we planned to present a series of papers to develop the conservative front-tracking method in two space dimensions. In [24], we developed for the first time the method and tested it on certain examples to verify its efficiency. Like its 1D version, the method tracks discontinuities by enforcing the conservation properties of solutions.

The present paper is the second one in the series in which we are going to explore another important feature of our 2D method, i.e. *discontinuity curves in 2D are tracked in a 1D capturing fashion*. As a matter of fact, it is this feature that makes our 2D front-tracking method Cartesian-grid-based, i.e. the method runs on Cartesian grid without introducing adaptive grids. Thus, the method's data structures are easy to manage and its algorithm is easy to code. Based on this exploration, we give in this paper a detailed description of implementation of our method, which includes the solution structure and the algorithm.

To begin with, we briefly review the front-tracking methods developed by other people. See [3,4,8,9,7,17,40,41] and the papers cited therein. Front-tracking methods are distinguished from capturing methods by having a separate numerical description, a lower dimensional adaptive moving grid, curve in 2D and surface in 3D, for a discontinuity. In most 2D front-tracking methods, this adaptive moving curve of a discontinuity is described by a set of logically connected points and elements, called front, and is stored in a doubly linked list. The front can be either grid-based or grid-free as shown in Fig. 1.1. There is also a fixed grid, maybe modified near the front to make a grid line to follow the discontinuity, for the solution in smooth regions.

The solution in smooth regions, defined either pointwisely or cell-averagely, are computed by numerical methods obtained by direct discretization of the governing equations, the Euler equations of compressible fluids, the Navier–Stokes equations of incompressible fluids, or other sets of equations. To compute the solution near a front, solution properties on and solution relations across the discontinuity would be approximated. For example, in the Euler equations Riemann problems need to be solved and in the incompressible Navier–Stokes equations surface tension needs to be evaluated on the front [15,39,40]. The front is tracked at each point by its moving velocity, which is obtained in certain ways. Shocks in the Euler equations move with velocities constrained by the Hugoniot conditions across them and can be obtained by solving Riemann problems on the fronts, while contact discontinuities and material interfaces in either the Euler equations or the incompressible Navier–Stokes equations are simply dragged by the velocity field of the flow.

In spite of their high accuracy and lack of numerical diffusion, the main criticism of the front-tracking methods is about their complexity. This complexity is obviously caused by the fact that the solution in smooth regions and the front are defined on different grids and computed using algorithms substantially independent of each other, which makes the integration of the latter into the former very complicated. The complexity

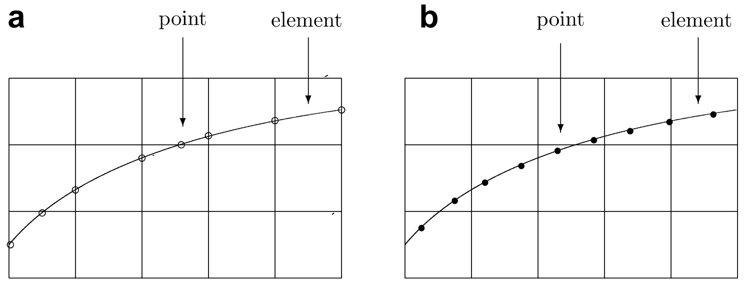


Fig. 1.1. Front and fix Cartesian grid. (a) The front is grid-based, where the small circles are the points which are connected by the curve elements. (b) The front is grid-free, where the dots are points which are connected by the curve elements.

starts with embedding a front into the fixed grid since the former cuts the latter and generates irregular grid cells varying with time near the front, see Fig. 1.1. The "one-flow" model suggested in [40] avoids the generation of irregular grid cells and thus somewhat eases the complexity, it seems that this approach has been applicable only to the incompressible Navier–Stokes equations so far. The communication between the fixed grid and the front, i.e. the data transfer from the fixed grid to the front and vise versa, adds to the complexity of the method. Finally, the management of the front itself also causes complexity. The front needs to be reconstructed from time to time to maintain a proper spacing between points and elements and prevent the formation of wiggles.

Some people think that the connectivity between points and elements on fronts is also a cause of algorithm complexity and thus design front-tracking methods without connectivity [36,39], in which indicator functions are used to describe the tracked discontinuity curve. The volume of fluid methods (VOF) [11,33,34] and the level set tracking methods [1,2,6] do not involve connectivity of points and elements either; therefore, their data structures and algorithms are easy to manage in coding. However, we do not think the connectivity will be a major cause for algorithm complexity when the tracking method is Cartesian-grid-based, such as the one presented in this paper. The structured feature and regularity of a Cartesian grid will greatly limit the cases needed to be handled in the algorithm, which will be seen in the following discussion. In addition, all the modern computer languages, such as Fortran 90 and C^{++}, well support all the fancy data structures that are necessary for describing the connectivity in either 2D or 3D, such as doubly linked lists, trees and graphs.

Our front-tracking method is based on the following fact: The discontinuities in solutions to Eq. (1.1) in 2D are moving curves; and like all 2D moving curves, their evolution can be locally described by 1D partial differential equations (PDE's). These 1D PDE's can be easily obtained by integrating Eq. (1.1) accordingly in either x or y direction; therefore, they are also conservation laws, however, in one space dimension and with source terms (see Section 3). The front-tracking is then accomplished by numerically simulating these 1D conservation laws with source terms in a conservative fashion. The key to our method is that the numerical simulation of these 1D PDE's is also carried out on the fixed Cartesian grid on which the solution in smooth region is computed. Moreover, the simulation of the 1D PDE's uses the same numerical fluxes on cell-interfaces used in the computation of solution in smooth regions. Thus, the integration of the tracking part of the algorithm into the part of the algorithm for the smooth regions becomes simple and natural. As will be seen in the following discussion, the former is just embedded into the latter. All the numerical complexities seen in other front-tracking methods are avoided. The management of a front also becomes quite simple because the discontinuity positions now move along the grid lines rather than in all directions as in other front-tracking methods. In addition, the simulation of the 1D conservation laws can enjoy many well-developed numerical methods for 1D conservation laws. The conservation of solutions is automatically preserved and enhancing the accuracy becomes easy. Although our method is developed for (1.1), a model of which is the compressible Euler equations, we believe that the methodology is applicable to other sets of equations, such as the incompressible Navier–Stokes equations.

We note the similarity of our front-tracking method to the VOF methods. Both of them reconstruct discontinuities by the constraint of local conservation and track them by evolving fluid volume forward in time with solutions to certain advection PDE's. However, the VOF methods are tracking methods without connectivity while ours is with connectivity. Moreover, the advection PDE used in VOF methods for the tracking is of 2D and is thus a global description of the flow evolution (see Section 3 in [34]), while the one used in our method is of 1D and is thus a local description of the evolution of the discontinuity curve. As will be seen in Section 3, benefited by these features the reconstruction and tracking of discontinuity curves in our front-tracking method are much simpler and more natural than that of the VOF methods.

The organization of the paper is as follows: Section 1 is the introduction. We begin our discussion with the 1D case in Section 2, in which we will show how to find a conservative ODE to describe the evolution of a discontinuity and how to discretize it on the fixed Cartesian grid to accomplish our front-tracking. The 1D discussion, though simple and of limited practical use, facilitates the following 2D discussion. The main part of the paper is Section 3, in which we present our 2D conservative front-tracking method. As in the 1D case, we first find the 1D PDE's to locally describe the evolution of a discontinuity curve, based on which a detailed description of our 2D front-tracking method, which includes the solution structure and algorithm, is presented. The tracking of a front in our method is accomplished by numerical simulation of the found 1D

PDE's, and we present in the section the first- and second-order discretization of these PDE's. A sketch on how their higher-order discretization can be accomplished is also given in the section. In Section 4 we present several numerical examples of scalar conservation laws and Euler system for γ-law gas computed with a second-order version of our front-tracking method to show the efficiency and effectiveness of the method. Finally, Section 5 is the conclusion.

2. One-dimensional method

To facilitate the development of our 2D front-tracking method we first describe in this section the 1D version of the method.

2.1. Mathematical formulation

Eq. (1.1) in one space dimension reads

$$u_t + f(u,x)_x = 0. \tag{2.1}$$

We begin with the case that both u and $f(u)$ are scalar and assume that the solution involves only one discontinuity, whose position at time t is $s(t)$. On the two sides of the discontinuity the solution, denoted by $u^-(x,t)$ and $u^+(x,t)$, respectively, is assumed to be smooth enough and can be smoothly extended to the other sides (see Fig. 2.1).

To find the equation that describes the evolution of the discontinuity in the solution, we first integrate (2.1) with respect to x from $x = a$ to $x = b$ and obtain

$$\frac{\mathrm{d}\int_a^b u(x,t)}{\mathrm{d}x}\mathrm{d}t + f(u^+(b,t),b) - f(u^-(a,t),a) = 0, \tag{2.2}$$

where a and b are two constants. Here $\int_a^b u(x,t)\mathrm{d}x$ is the total volume of the physical species u in the interval $[a,b]$ at time t, and it consists of two parts, the integrals of $u^-(x,t)$ and of $u^+(x,t)$, respectively,

$$\int_a^b u(x,t)\,\mathrm{d}x = \int_a^{s(t)} u^-(x,t)\mathrm{d}x + \int_{s(t)}^b u^+(x,t)\mathrm{d}x, \tag{2.3}$$

with $s(t)$ as the dividing point. Since we are concerned with the evolution of the discontinuity, we regard $s(t)$ as the unknown in Eq. (2.2) and view $u(x,t)$ as known. Under this consideration, the integral $\int_a^b u(x,t)\mathrm{d}x$ is a function of the discontinuity position $s(t)$, and we denote it by $U_{(a,b)}(s(t))$. With this notation, Eq. (2.2) reads

$$\frac{\mathrm{d}U_{(a,b)}(s(t))}{\mathrm{d}t} + F_{(a,b)}(t) = 0 \tag{2.4}$$

with $F_{(a,b)}(t) = f(u^+(b,t),b) - f(u^-(a,t),a)$ the flux difference between the two ends. This ordinary differential equation describes the evaluation of the discontinuity in the solution. It should be noted that $s(t)$ is not necessary to be between a and b, it can be either on the left of a or the right of b. We should bear this in mind when developing the front-tracking method in the following discussion.

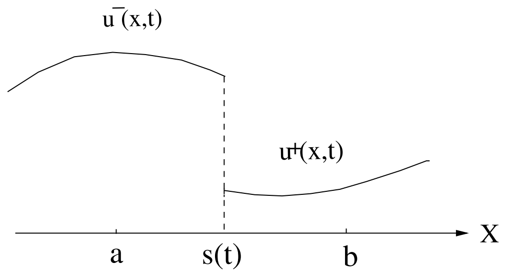


Fig. 2.1. The solution is piecewise smooth and involves only one discontinuity, whose position is $s(t)$ at time t.

The physics Eq. (2.4) describes is familiar to us, i.e. the conservation of u in the interval $[a,b]$. More precisely, the variation of the total volume of u in the interval is balanced by the flow-in and -out through the two ends of the interval, and this variation of u causes the movement of the discontinuity position in the interval as shown in Fig. 2.2.

The solution to (2.1) satisfies the Hugoniot condition across the discontinuity

$$[u]\frac{\mathrm{d}s}{\mathrm{d}t}=[f], \tag{2.5}$$

where $[u]$ and $[f]$ are jumps of the solution and flux across the discontinuity. A quick calculation shows that (2.4) and (2.5) are equivalent to each other, both describing the evolution of the discontinuity. However, Eq. (2.4) is in a conservation form and (2.5) is not.

Eq. (2.4) can be easily integrated because the flux difference $F_{a,b}(t)$ involved in it is a known function of t. Once $U_{(a,b)}(s(t))=\int_a^b u(x,t)\mathrm{d}x$ was computed, the discontinuity position $s(t)$ can then be solved out from (2.3) with $u^-(x,t)$ and $u^+(x,t)$ as known functions. If the interval $[a,b]$ is small and $u^-(x,t)$ and $u^+(x,t)$ do not vary rapidly, Eq. (2.3) admits a unique solution for $s(t)$.

When (1.1) is a system of equations we have for each physical species of u an equation of (2.3), and this situation should be counted in our front-tracking method.

2.2. Numerical implementation

The 1D front-tracking method going to be described in this subsection is developed in [26,28–31,20,21,12,13].

2.2.1. Structure of solution

Our numerical solution is defined on a fixed Cartesian grid. The grid cells harboring no discontinuity are called smooth cells, and the grid cells harboring one discontinuity are called discontinuity cells. In a smooth cell the numerical solution is a cell-average approximation to the exact solution

$$u_j^n\simeq\frac{1}{h}\int_{x_{j-1/2}}^{x_{j+1/2}}u(x,t_n)\mathrm{d}x. \tag{2.6}$$

However, in a discontinuity cell the numerical solution has three cell-average approximations, the left cell-average $u_j^{n,-}$, which is a cell-average approximation to the smooth solution and its extension on the left of the tracked discontinuity, the right cell-average $u_j^{n,+}$, which is a cell-average approximation to the smooth solution and its extension on the right of the tracked discontinuity, and the ordinary cell-average u_j^n, which is the cell-average approximation to the exact solution involving the tracked discontinuity. The left and right cell-averages are actually two "ghost" states used to describe the numerical solution and its extension on the two sides of the tracked discontinuity.

Our front-tracking method involves a ghost technique, called "stack-technique", to handle the close-to-each-other discontinuities. A grid cell may host more than one discontinuity cells and they are neighboring

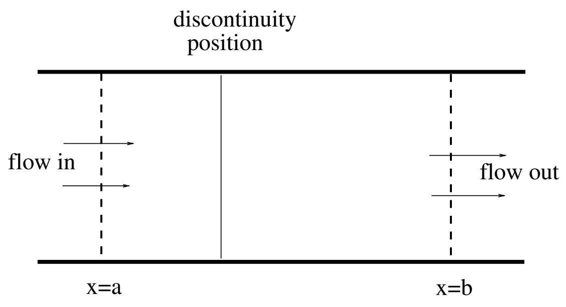


Fig. 2.2. The variation of the total volume of u in the interval is caused by the flow-in and -out through the two ends, which causes the movement of the discontinuity position.

to each other. Any two neighboring critical cells in the stack share a common cell-average, which is the left cell-average of the right discontinuity cell and the right cell-average of the left one. Fig. 2.3 shows the smooth cells and discontinuity cells, single and stacked. A stacked discontinuity cell is treated in our method as a single discontinuity cell with its left and right cell-averages as the smooth solution on its two sides.

In the program, all the discontinuity cells and their related information are stored in a doubly linked list in the order from the left to the right. There is a grid map to indicate whether a grid cell is smooth cell or hosts discontinuity cells, and if the cell hosts a stack of discontinuity cells the map also stores the addresses of the top (left) and the bottom (right) members of the stack in the doubly linked list. In this way, the solution is structured in an "I-know-only-my-neighbors" fashion; that is, in each grid cell, either smooth or discontinuity, one can find its neighbor cells by the grid map and the doubly linked list of the discontinuity cells.

2.2.2. *Computing solution in smooth regions*

Since stacked discontinuity cells are treated in our method as single discontinuity cells, in the following two sub-subsections we will only consider the case of single discontinuity cells. The stacked discontinuity cells can be treated analogously in the algorithm.

We employ a finite-volume scheme

$$u_j^{n+1} = u_j^n - \lambda(\hat{f}_{j+1/2}^n - \hat{f}_{j-1/2}^n) \tag{2.7}$$

for the computation of smooth regions, where u_j^n is the cell-average approximation to the exact solution and $\hat{f}_{j+1/2}^n$ is the flux average approximation to $f(u,x)$ on the cell-interface at $x_{j+1/2}$

$$\hat{f}_{j+1/2}^n = \hat{f}(u_{j-k+1}^n, \ldots, u_{j+k}^n) \simeq \frac{1}{\tau}\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2},t), x_{j+1/2})\mathrm{d}t, \tag{2.8}$$

which is consistent with the flux function $f(u,x)$ in the sense as described in [18, Section 12.2], and $\lambda = \tau/h$ is the mesh ratio with τ and h being the time and space increments of the grid, respectively. The computation of smooth regions is proceeded in the fashion that on each side of a discontinuity it uses information only from the same side. This can be accomplished as whenever data across the tracked discontinuity are required, smooth extension data from the same side are used. For example, if the solution has a discontinuity cell in the grid cell $[x_{j_1-1/2}, x_{j_1+1/2}]$, then the solution in the smooth cells on the two sides is computed as

$$u_j^{n+1} = u_j^n - \lambda(\hat{f}_{j+1/2}^{n,\pm} - \hat{f}_{j-1/2}^{n,\pm}), \tag{2.9}$$

with "−" for $j < j_1$ and "+" for $j > j_1$, respectively, where

$$\hat{f}_{j+1/2}^{n,-} = \hat{f}(u_{j-k+1}^n, \ldots, u_{j_1-1}^n, u_{j_1}^{n,-}, \ldots, u_{j+k}^{n,-}, x_{j+1/2}), \tag{2.10}$$

and

$$\hat{f}_{j+1/2}^{n,+} = \hat{f}(u_{j-k+1}^{n+}, \ldots, u_{j_1}^{n,+}, u_{j_1+1}^n, \ldots, u_{j+k}^n, x_{j+1/2}) \tag{2.11}$$

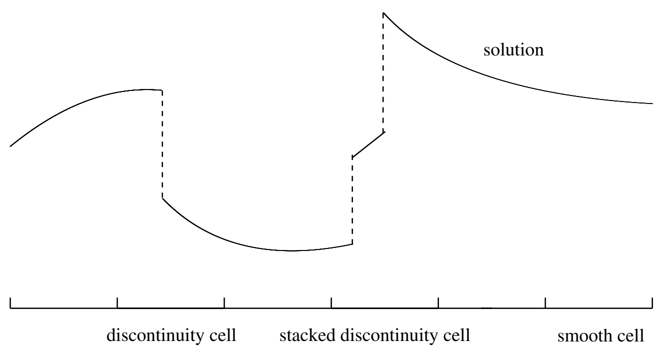


Fig. 2.3. Smooth cells and discontinuity cells in one space dimension.

with $u_i^{n,\pm}$ being the left and right cell-averages in the discontinuity cell or the smooth extension data of the cell-averages from the two sides to the other sides. The left and right cell-averages in the discontinuity cell are computed in the same fashion as above, i.e.

$$u_{j_1}^{n+1,\pm} = u_{j_1}^{n,\pm} - \lambda(\hat{f}_{j_1+1/2}^{n,\pm} - \hat{f}_{j_1-1/2}^{n,\pm}). \tag{2.12}$$

In doing this, the "I-know-only-my-neighbors" structure of the solution allows us to easily produce the smooth data used in the evaluation of (2.10) or (2.11). When discontinuity cells are close to each other or even stacked in the same grid cells, the extension data used are obtained by low-order or even zeroth-order extrapolation because there are not enough grid cells to implement high-order extrapolation.

Once the cell-averages in all smooth cells and the left and right cell-averages in all discontinuity cells are computed, the solution in each smooth region, including the smooth regions between stacked discontinuity cells, and its extension near discontinuities can then be reconstructed via interpolation and extrapolation. As a matter of fact, most finite-volume schemes, e.g. [10,5], provide this reconstruction facility. However, when discontinuity cells are closed to each other or even stacked in the same grid cell, the reconstruction can only be accomplished with low-order or even zeroth-order interpolations due to the lack of enough grid cells. Moreover, the flow fluxes across all cell-interfaces during the time step are computed. Therefore, in the following discussion the solution in smooth regions and the flow fluxes across all cell-interfaces are known. For simplicity of discussion we denote the reconstructed solution with u in the following discussion without risk of ambiguity.

2.2.3. Tracking discontinuities

We now discretize Eq. (2.4) to compute the ordinary cell-average in a discontinuity cell, and again we assume the discontinuity cell to be in the grid cell $[x_{j_1-1/2}, x_{j_1+1/2}]$. By taking $a = x_{j_1-1/2}$ and $b = x_{j_1+1/2}$ in (2.3) we have

$$\frac{U_{(a,b)}(s(t))}{h} = \frac{1}{h}\int_{x_{j_1-1/2}}^{x_{j_1+1/2}} u(x,t)\mathrm{d}x = u_{j_1}(t). \tag{2.13}$$

Integrating (2.4) with respect to t over the interval $[t_n, t_{n+1}]$ and noticing (2.13) we arrive at

$$u_{j_1}^{n+1} = u_{j_1}^{n} - \lambda\left(\frac{1}{\tau}\int_{t_n}^{t_{n+1}} f(u^+(x_{j_1+1/2},t), x_{j_1+1/2})\mathrm{d}t - \frac{1}{\tau}\int_{t_n}^{t_{n+1}} f(u^-(x_{j_1-1/2},t), x_{j_1-1/2})\mathrm{d}t\right), \tag{2.14}$$

where $u_{j_1}^n$ and $u_{j_1}^{n+1}$ are the ordinary cell-averages in the discontinuity cell at time t_n and t_{n+1}, respectively. By approximating the fluxes $\frac{1}{\tau}\int_{t_n}^{t_{n+1}} f(u^\pm(x_{j_1\pm1/2},t), x_{j_1\pm1})\mathrm{d}t$ with the numerical fluxes $\hat{f}_{j_1\pm1/2}^{n,\pm}$ used in (2.12) we obtain

$$u_{j_1}^{n+1} = u_{j_1}^{n} - \lambda\left(\hat{f}_{j_1+1/2}^{n,+} - \hat{f}_{j-1/2}^{n,-}\right). \tag{2.15}$$

Once the cell-average $u_{j_1}^{n+1}$ is computed, the discontinuity position $s^{n+1} \simeq s(t_{n+1})$ is solved from (2.3) with $a = x_{j_1-1/2}$ and $b = x_{j_1+1/2}$. When (2.1) is scalar, there is only one equation of (2.3) for solving s^{n+1}. In this case, if the smooth solution is reconstructed in a piecewise constant fashion, the discontinuity position is simply solved as

$$s^{n+1} = x_{j_1} + \frac{u_{j_1}^{n+1} - u_{j_1}^{n+1,-}}{u_{j_1}^{n+1,+} - u_{j_1}^{n+1,-}}. \tag{2.16}$$

When (2.1) is a system, we will have more than one equation of (2.3) for solving s^{n+1}. In this case, the final discontinuity position should be taken as an average of all the solved s^{n+1}'s. However, when the governing equations are the Euler system and the tracked discontinuity is a contact discontinuity, the discontinuity position is taken as the solution of (2.3) for the density since the contact discontinuity is a discontinuity of density.

2.2.4. Moving and collisions of discontinuities

The discontinuity position s^{n+1} at the next time level may move out of the original discontinuity cell. Under the CFL restriction on the mesh ratio τ/h, s^{n+1} may move only to either the left or the right neighbor cell. In this case, the old discontinuity cell should be deleted. If the neighbor grid cell was a smooth cell, it now hosts a discontinuity cell, and if the neighbor grid cell already hosted a stack of discontinuity cells, now one more discontinuity cell is added on the stack. It is important to maintain the conservation of the solution in updating the cell-averages. For example, if the grid cell $[x_{j_1-1/2}, x_{j_1+1/2}]$ hosted a discontinuity cell at t_n, and s^{n+1} moved into the left neighbor grid cell at t_{n+1}. In this case, cell $[x_{j_1-1/2}, x_{j_1+1/2}]$ becomes smooth cell and the cell $[x_{j_1-3/2}, x_{j_1-1/2}]$ hosts a discontinuity cell at t_{n+1}, and the cell-averages are updated as

$$\begin{cases} u_{j_1-1}^{n+1,-} := u_{j_1-1}^{n+1}, \\ u_{j_1-1}^{n+1} := u_{j_1-1}^{n+1} + u_{j_1}^{n+1} - u_{j_1}^{n+1,+}, \\ u_{j_1}^{n+1} := u_{j_1}^{n+1,+}, \\ u_{j_1-1}^{n+1,+} := \text{extension data from the left.} \end{cases} \tag{2.17}$$

In doing so the sum $u_{j_1-1}^{n+1} + u_{j_1}^{n+1}$ is unchanged; therefore, the conservation of the solution is preserved. The cases of moving to the right and of stacked discontinuity cells are treated analogously.

The discontinuity positions of two neighboring discontinuity cells in a stack may move across each other, which means that the two tracked discontinuities must collide somewhen during the time step. In this case, the old discontinuity cells should be deleted and new discontinuity cells corresponding to the discontinuities emanating from the collision should be set, and all this is implemented at t_{n+1} without the knowledge of the exact collision time during the time step. The Riemann problem with the left cell-average of the left discontinuity cell and the right cell-average of the right discontinuity cell as the left and right states is solved to find the values for the left and right cell-averages of the new discontinuity cells. The ordinary cell-averages of the new discontinuity cells are computed in accordance with the characteristic waves emanating from the Riemann problem and maintaining the conservation of the solution. Since the method is Cartesian-grid-based, all these can be implemented easily and naturally. The detailed description of the algorithm for moving and collisions of discontinuity cells can be found in [21,26].

2.2.5. *Propagation of waves in other characteristic fields*

If (2.1) is a system of equations, there will be several characteristic fields and thus several different kinds of discontinuities. When tracking a discontinuity of a certain kind, waves in other characteristic fields may propagate across it. This situation should be treated properly. Principally speaking, the Riemann problem is solved at the discontinuity, and based on the solution to the Riemann problem information related to the other characteristic fields are separated from the cell-averages in the discontinuity cell and then transferred to the corresponding sides to update the left or right cell-averages there. Certain integrals of the type $\int(u^+ - u^-)\mathrm{d}x$ will be evaluated; nevertheless, they are all defined on the fixed Cartesian grid and thus can be easily evaluated by numerical quadratures. The detailed algorithm for the wave propagation in other fields can be found in [26,20,21].

2.3. *Summary*

Thus, we completed the description of our 1D conservative front-tracking method. As is seen in the description, the method has the following features: (1) Based on the philosophy of the Lax–Wendroff theorem, the method tracks discontinuities by enforcing the conservation properties of solutions. This conservation feature of the method, together with the accuracy of the underlying scheme (2.7), guarantees the accuracy of the tracking and the robustness of the treatment of collisions of discontinuities and spontaneous shocks, which has been verified by the numerical examples presented in [20,26]. A strict analysis of truncation error of the method is presented in [20] (see the Theorem 3.1 in that paper). (2) The method is Cartesian-grid-based. Actually, it is formulated as consisting of a 1D capturing and an ODE differencing, the former computing the solution in smooth regions and the latter tracking discontinuities. Both the 1D capturing and ODE differencing are implemented on the fixed Cartesian grid; therefore, the data structures involved are much simpler and algorithm is much easier to code compared with those of other 1D front-tracking methods. Moreover, the method completely gets rid of the small-cell problem that bothers most front-tracking methods. We have built a code for the 1D method in Fortran90, which involves about 3500 instructions with thirty plus modules on different levels and has the facilities of tracking, handling collisions of discontinuities, and capturing spontaneous shocks and then tracking them in the following computation.

3. Two-dimensional method

Now we are going to develop our 2D front-tracking method, and we will still proceed our discussion as in the previous section for the 1D method.

3.1. Mathematical formulation

Eq. (1.1) in two space dimensions reads

$$u_t + f(u,x,y)_x + g(u,x,y)_y = 0. \tag{3.1}$$

We still begin with the scalar case and assume that the solution involves only one discontinuity, on the two sides of which the solution, denoted by $u^-(x,y,t)$ and $u^+(x,y,t)$, respectively, is smooth enough and can be smoothly extended to the other sides. An orientation is defined on the discontinuity curve, and in the following discussion the orientation is always so defined that u^- is on the left and u^+ is on the right of the curve.

In the 2D case, the discontinuity is a moving curve in the (x,y)-plane. Let us begin with a special case that the discontinuity curve is described by a smooth function $y = s(x,t)$ at time t, above which is the right region and below which is the left region, as shown in Fig. 3.1. As in the 1D case, we first integrate (3.1) with respect to y from $y = a$ to $y = b$ and obtain

$$\frac{\partial \int_a^b u(x,y,t)\mathrm{d}y}{\partial t} + \frac{\partial \int_a^b f(u(x,y,t),x,y)\mathrm{d}y}{\partial x} + g(u^+(x,b,t),x,b) - g(u^-(x,a,t),x,a) = 0, \tag{3.2}$$

where a and b are again two constants. There are two integrals in (3.2), i.e. $\int_a^b u(x,t)\mathrm{d}y$ and $\int_a^b f(u(x,y,t),x,y)\,\mathrm{d}y$, and each of them consists of two parts, the integrals involving $u^-(x,t)$ and involving $u^+(x,t)$, respectively. That is,

$$\int_a^b u(x,y,t)\mathrm{d}y = \int_a^{s(x,t)} u^-(x,y,t)\mathrm{d}y + \int_{s(x,t)}^b u^+(x,y,t)\mathrm{d}y \tag{3.3}$$

and

$$\int_a^b f(u(x,y,t),x,y)\mathrm{d}y = \int_a^{s(x,t)} f(u^-(x,y,t),x,y)\mathrm{d}y + \int_{s(x,t)}^b f(u^+(x,y,t),x,y)\mathrm{d}y. \tag{3.4}$$

As in the 1D case, we regard $s(x,t)$ as the unknown in Eq. (3.2) and view $u(x,y,t)$ as known. Under this consideration, both the integrals in (3.3) and (3.4) are functions of the discontinuity position $s(x,t)$ and x and t, and we denote them by $U_{(a,b)}(s(x,t))$ and $F_{(a,b)}(s(x,t),x,t)$, respectively. With this notation, Eq. (3.2) reads

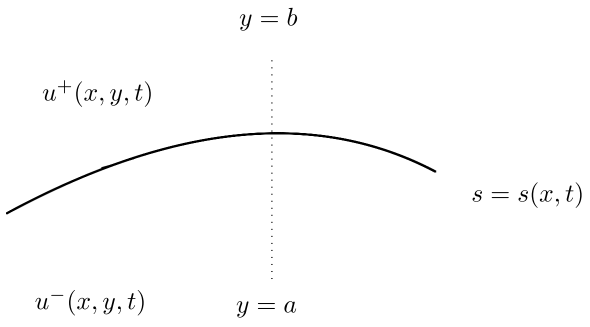


Fig. 3.1. The solution is piecewise smooth and involves only one discontinuity curve, which is described by a function $y = s(x,t)$ at time t.

$$\frac{\partial U_{(a,b)}(s(x,t))}{\partial t}+\frac{\partial F_{(a,b)}(s(x,t),x,t)}{\partial x}+G_{(a,b)}(x,t)=0 \tag{3.5}$$

with $G_{(a,b)}(x,t)=g(u^+(x,b,t),x,b)-g(u^-(x,a,t),x,a)$ the vertical flux difference between a and b. This partial differential equation describes the evaluation of the discontinuity curve in the solution region with $s(x,t)$ as the curve point at x.

The form of Eq. (3.5) looks familiar to us, a 1D conservation law with a source term $G_{(a,b)}(x,t)$, which is the y-flux difference between $y=a$ and $y=b$. The only difference is that the unknown $s(x,t)$ appears in a (non-linear) function $U_{(a,b)}$. The physics Eq. (3.5) describes is still the conservation of u. More precisely, if we draw a rectangle with $y=a$ the bottom, $y=b$ the top, $x=x$ the left edge and $x=x+\Delta x$ the right edge with $\Delta x\to 0$, see Fig. 3.2(a), then what (3.5) says is that the variation of the total volume of u in the rectangle is balanced by the flow-in and -out through the boundary of the rectangle. This variation of u causes the movement of the discontinuity curve in the rectangle, driving the curve either up or down.

The solution to (3.1) satisfies the Hugoniot condition across the discontinuity curve, which reads

$$[u]\frac{\partial s}{\partial t}+[f]\frac{\partial s}{\partial x}=[g] \tag{3.6}$$

with x and t viewed as the independent variables, where $[u]$, $[f]$ and $[g]$ are the jumps of the solution and the two fluxes across the discontinuity (see [23]). A quick calculation also shows that (3.5) and (3.6) are equivalent to each other; both describing the evolution of the discontinuity curve in the solution. However, Eq. (3.5) is in a conservation form and (3.6) is not. It is seen that Eq. (3.6) is hyperbolic and it has an eigenvalue $[f]/[u]$, which is just the x-component of the propagation velocity of the discontinuity. Because of the equivalency between the two equations, Eq. (3.5) is also hyperbolic and $[f]/[u]$ is its eigenvalue.

As in the 1D case, the discontinuity curve $s(x,t)$ is not necessary between $y=a$ and $y=b$. It can be in the situation as shown in Fig. 3.2(b) or even totally out of the rectangle. We should bear this in mind when developing our 2D front-tracking method in the following discussion.

Unlike the 1D case, Eq. (3.5) can not be straightforwardly integrated. Nevertheless, it can be numerically simulated and its hyperbolicity and conservation feature allow its numerical solution to enjoy many well-developed numerical methodologies, which will be seen in the following discussion.

When a discontinuity curve is described by a smooth function $s=s(y,t)$, the analogous discussion can be proceeded as well and the result is the same, but with a shift of the two space dimensions. As a matter of fact, a general discontinuity curve can always be divided into finitely many parts each of which can be either

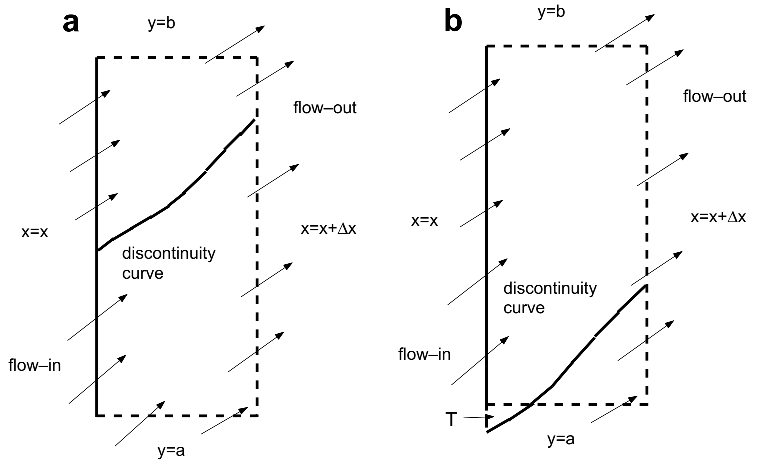


Fig. 3.2. (a) The variation of the total volume of u in the rectangle is balanced by the flow-in and -out through the boundary of the rectangle, which causes the movement of the discontinuity curve in it. (b) The physical explanation in (a) still holds even when part of the curve segment is out of the rectangle.

described as $s = s(y, t)$, called an xx-part, or $s = s(x, t)$, called a yy-part. It will be seen in the following discussion that our 2D front-tracking is developed on the basis of this fact.

3.2. Numerical implementation

3.2.1. Structure of solution

As in the 1D method, our numerical solution is defined on a fixed Cartesian grid. The grid cells harboring no discontinuity curve are called smooth cells, and the grid cells harboring a segment of discontinuity curve are called discontinuity cells. In a smooth cell the numerical solution is a cell-average approximation to the exact solution

$$u^n_{i,j} \simeq \frac{1}{h^2}\int_{x_{i-1/2}}^{x_{i+1/2}}\int_{y_{j-1/2}}^{y_{j+1/2}} u(x, y, t_n)\mathrm{d}x. \tag{3.7}$$

However, in a discontinuity cell the numerical solution has three cell-average approximations, the left cell-average $u^{n,-}_{i,j}$, which is a cell-average approximation to the smooth solution and its extension on the left of the tracked discontinuity curve, the right cell-average $u^{n,+}_{i,j}$, which is a cell-average approximation to the smooth solution and its extension on the right of the tracked discontinuity curve, and the ordinary cell-average $u^n_{i,j}$, which is the cell-average approximation to the exact solution involving the tracked discontinuity curve. The left and right cell-averages are actually two "ghost" states used to describe the numerical solution and its extension on the two sides of the tracked discontinuity curve.

What is different from the 1D method is that we now have three different types of discontinuity cells defined by how they are crossed by the discontinuity curve. We postulate that the discontinuity curve intersects only two of the cell-edges when it crosses a discontinuity cell. The discontinuity cells are then classified into xx-type, of which the two x-edges (horizontal edges) of a cell are intersected, the yy-type, of which the two y-edges (vertical edges) of a cell are intersected, and the xy-type, of which an x-edge and a y-edge of a cell are intersected. Fig. 3.3 shows a situation of smooth cells and discontinuity cells. All the discontinuity cells and their related information of a tracked discontinuity curve are stored in a doubly linked list, which is called a front.

Our 2D method also involves a "stack-technique" to handle the close-to-each-other discontinuity cells. A grid cell may host more than one discontinuity cells and they are neighboring to each other. Any two neighboring critical cells in the stack share a common cell-average, which is either the left or the right cell-average of

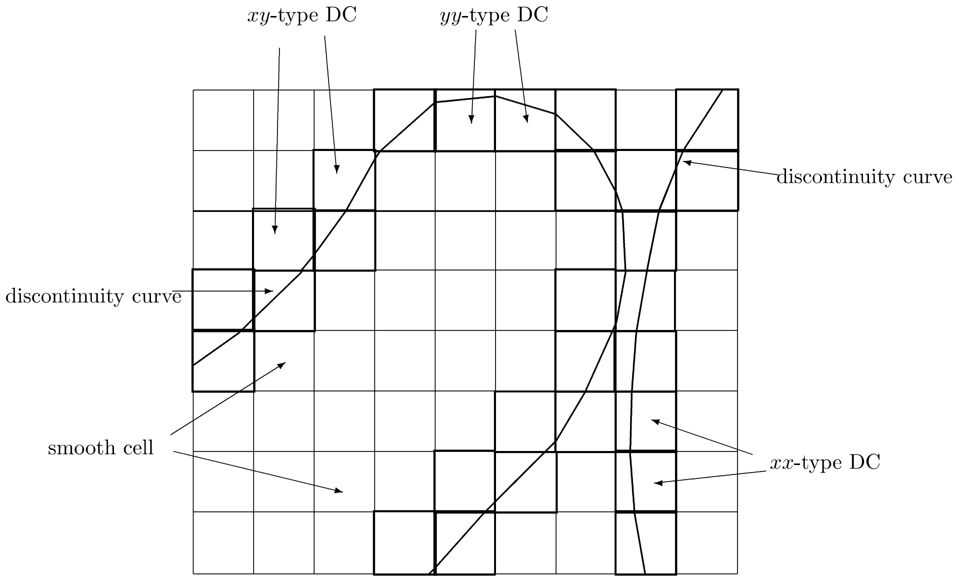


Fig. 3.3. Smooth cells and discontinuity cells in two space dimensions. Here DC stands for discontinuity cell.

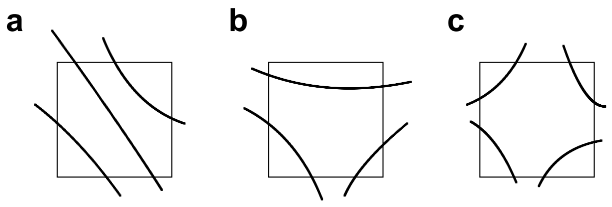


Fig. 3.4. The stack situation in (a) is allowed and the stack situations in (b) and (c) are not allowed.

each of the two neighboring discontinuity cells. This means that we allow only the stack situation as shown in Fig. 3.4(a) and do not allow the stack situation as shown in Fig. 3.4(b) and (c). In a stack of discontinuity cells each member has two pointers pointing to its left and right neighbors in the stack. A stacked discontinuity cell is treated in our method as a single discontinuity cell with its left and right cell-averages as the smooth solution on its two sides.

In each step of computation we need to produce an auxiliary front for every front in which each xy-type discontinuity cell, if possible, should be reshaped with one of its neighbor xy-type discontinuity cells on the front to produce an xx- or yy-type discontinuity cell, which is called a reshaped discontinuity cell. For example, the two neighboring xy-type discontinuity cells $\Delta_{i,j}$ and $\Delta_{i,j-1}$ as shown in Fig. 3.5(a) will be reshaped as a pair of yy-type discontinuity cells as shown in Fig. 3.5(b) and (c). In the reshaping, each discontinuity cell drops its discontinuity position on the x-edge and gets its neighbor's discontinuity position on the y-edge as one of its discontinuity position. The reshaped yy-type discontinuity cells are different from the ordinary yy-type discontinuity cells in that parts of their volumes are pulled out by their out-of-cell discontinuity positions on the y-edges. For the pair of reshaped yy-type discontinuity cells, the left and right cell-averages in them are still the ones in the original xy-type cells; however, the ordinary cell-averages over them should be recomputed as

$$u^n_{i,j} := u^n_{i,j} + u^n_{i,j-1} - u^{n,-}_{i,j-1} \tag{3.8}$$

and

$$u^n_{i,j-1} := u^n_{i,j} + u^n_{i,j-1} - u^{n,+}_{i,j}, \tag{3.9}$$

respectively, which comes from the observation that the out-of-cell discontinuity positions on the y-edges pull parts of the volumes of the discontinuity cells out of the cells.

The reshaping to produce a pair of xx-type discontinuity cells on a front is analogous. When the tracked discontinuity curve is smooth and the grid is fine enough, in most cases an xy-type discontinuity cell has at least one of its neighbor cells on the front being of xy-type with which it can be reshaped. However, very often

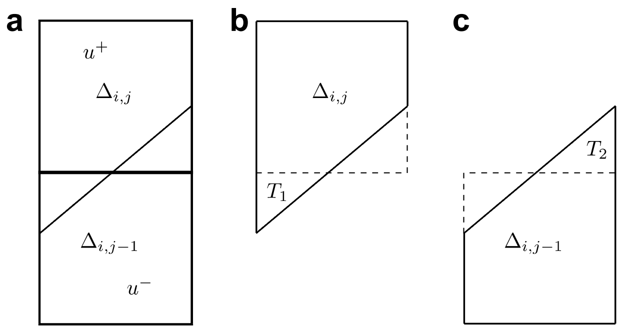


Fig. 3.5. Two neighboring xy-type discontinuity cells shown in (a) are reshaped to produce a pair of yy-type discontinuity cells as shown in (b) and (c). Each discontinuity cell drops its discontinuity position on the x-edge and gets its neighbor discontinuity cell's discontinuity position on the y-edge. Part of the volume of each reshaped discontinuity cell is pulled out by its out-of-cell discontinuity positions on the y-edges.

an xy-type discontinuity cell has both of its neighbor discontinuity cells on the front being of xy-type with which it can be reshaped. To avoid this dilemma we produce the auxiliary front for each front alternatively favoring the x- and y-directions in time, that is, in one time step we reshape the discontinuity cell in xx-type when the situation occurs and in the next time step we reshape the cell in yy-type. Therefore, for a front with a given favored direction the auxiliary front is uniquely determined. The production of an auxiliary front is actually the numerical division of the front into xx- and yy-parts. Fig. 3.6 shows an auxiliary front produced favoring the y-direction. As we can see from the figure that not all xy-type discontinuity cells are reshaped in an auxiliary front even for a smooth discontinuity curve; there are still single xy-type discontinuity cells and they occur at joints of xx-parts and yy-parts of the front.

All the discontinuity cells and their related information of an auxiliary front are also stored in a doubly linked list. The reshaped discontinuity cells on an auxiliary front inherit the "stack" relations on the original front.

As in the 1D method, there is also a grid map to indicate whether a grid cell is a smooth cell or hosts discontinuity cells, and if the cell hosts a stack of discontinuity cells the map also stores the addresses of the top and bottom members of the stack in the doubly linked lists of the fronts or the auxiliary fronts they belong to. In this way, the solution is structured in an "I-know-only-my-neighbors" fashion, that is, in each grid cell, either smooth or discontinuity, one can find its neighbor cells by the grid map and the doubly linked list of the front and the doubly linked list of the stack in the grid cell.

3.2.2. Computing solution in smooth regions

Since stacked discontinuity cells are treated as single discontinuity cells, in the following discussion we consider only the case without stack.

We employ a finite-volume scheme

$$u_{i,j}^{n+1} = u_{i,j}^n - \lambda(\hat{f}_{i+1/2,j}^n - \hat{f}_{i-1/2,j}^n) - \lambda(\hat{g}_{i,j+1/2}^n - \hat{g}_{i,j-1/2}^n) \tag{3.10}$$

defined on the fixed Cartesian grid for the computation of smooth regions, where $u_{i,j}^n$ is the cell-average approximation to the exact solution and $\hat{f}_{i+1/2,j}^n$ and $\hat{g}_{i,j+1/2}^n$ are the flux average approximations to $f(u,x,y)$ and $g(u,x,y)$ on the y- and x-cell-interfaces $\partial\Delta_{i+1/2,j}$ and $\partial\Delta_{i,j+1/2}$, respectively,

$$\hat{f}_{i+1/2,j}^n \simeq \frac{1}{h\tau}\int_{t_n}^{t_{n+1}}\int_{y_{j-1/2}}^{y_{j+1/2}} f(u(x_{i+1/2},y,t),x_{i+1/2},y)\mathrm{d}y\,\mathrm{d}t \tag{3.11}$$

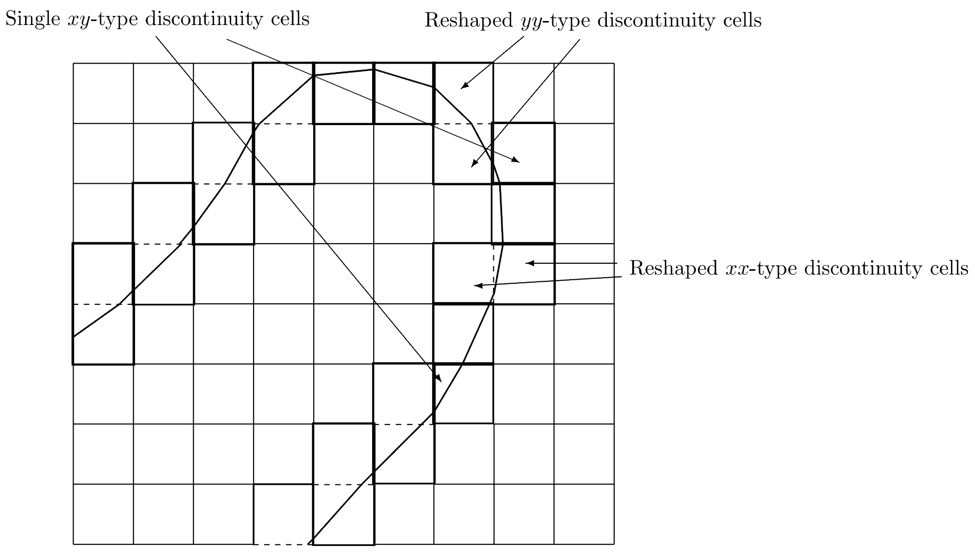

Single xy-type discontinuity cells
Reshaped yy-type discontinuity cells
Reshaped xx-type discontinuity cells

Fig. 3.6. An auxiliary front produced favoring the y-direction, on which every xy-type discontinuity cell, if possible, is reshaped with one of its neighbor cells to form an xx- or yy-type discontinuity cell. There are still some single xy-type discontinuity cells on the auxiliary front.

and

$$\hat{g}^n_{i,j+1/2} \simeq \frac{1}{h\tau}\int_{t_n}^{t_{n+1}}\int_{x_{i-1/2}}^{x_{i+1/2}} g(u(x,y_{j+1/2},t),x,y_{j+1/2})\mathrm{d}x\,\mathrm{d}t, \tag{3.12}$$

which are consistent with the flux functions $f(u,x,y)$ and $g(u,x,y)$ as in the one-dimensional case, and $\lambda = \tau/h$ is the mesh ratio with τ and h being the time and space increments of the grid, respectively.

As in the 1D case, the computation of each smooth region is carried out using the information only from the same region. Thus, it proceeds in the following fashion in the left and right regions of a front, respectively:

$$u^{n+1}_{i,j} = u^n_{i,j} - \lambda(\hat{f}^{n,\pm}_{i+1/2,j} - \hat{f}^{n,\pm}_{i-1/2,j}) - \lambda(\hat{g}^{n,\pm}_{i,j+1/2} - \hat{g}^{n,\pm}_{i,j-1/2}) \tag{3.13}$$

if the grid cell $\Delta_{i,j}$ is a smooth cell, "−" for the left region and "+" for the right region, and

$$u^{n+1,\pm}_{i,j} = u^{n,\pm}_{i,j} - \lambda(\hat{f}^{n,\pm}_{i+1/2,j} - \hat{f}^{n,\pm}_{i-1/2,j}) - \lambda(\hat{g}^{n,\pm}_{i,j+1/2} - \hat{g}^{n,\pm}_{i,j-1/2}) \tag{3.14}$$

if $\Delta_{i,j}$ is a discontinuity cell. Here $\hat{f}^{n,\pm}$ and $\hat{g}_{n,\pm}$ are evaluated in the way that when data across the discontinuity curve are required the smooth extension data from the same sides are provided. If the scheme (3.10) is constructed from a semi-discretization of (3.1) with the temporal derivatives being discretized in a Runge–Kutta fashion, the evaluation of $\hat{f}^{n,\pm}$ and $\hat{g}_{n,\pm}$ needs only the extension data in the horizontal and vertical directions. No data in diagonal directions is required (see [24,27]). In doing this, the "I-know-only-my-neighbors" structure of the solution allows us to easily produce the smooth data used in evaluating $\hat{f}^{n,\pm}$ and $\hat{g}_{n,\pm}$ in (3.13) or (3.14). When discontinuity cells are close to each other or even stacked in the same grid cells, the extension data used are obtained by low-order or even zeroth-order extrapolation because there are no enough grid cells to implement high-order extrapolation.

Once the cell-averages in all smooth cells and the left and right cell-averages in all discontinuity cells are computed, the solution in all smooth regions, including the smooth regions between stacked discontinuity cells, and its extension near discontinuities can then be reconstructed via interpolation and extrapolation. Since the solution is defined on a fixed Cartesian grid, the reconstruction can be accomplished dimension-by-dimensionly using 1D reconstruction. However, when discontinuity cells are closed to each other or even stacked in the same grid cell, the reconstruction can only be accomplished with low-order or even zeroth-order interpolations due to the lack of enough grid cells. Now we can reconstruct the solution in smooth regions at t_n and t_{n+1}, respectively, using the cell-averages in the regions at the two time levels, and in the following discussion we denote the reconstructed solution in smooth regions by u without risk of ambiguity.

Moreover, the flow fluxes across all cell-interfaces in all smooth regions during the time step can also be reconstructed from the information obtained in evaluating numerical fluxes on cell-interfaces. For example, if the underlying scheme is constructed from a semi-discretization of (3.1) with the temporal derivatives being discretized in a Runge–Kutta fashion, the prediction values of $f(u^{\pm},x,y)$ in all the time stages in the Runge–Kutta procedure are the necessary information for the reconstruction of flow fluxes on cell-interfaces. In the following discussion we denote by $f_{i+1/2}(y,t)$ and $g_{j+1/2}(x,t)$ the flow fluxes on the vertical cell-interfaces at $x_{i+1/2}$ and on the horizontal cell-interfaces at $y_{j+1/2}$, respectively.

3.2.3. Tracking discontinuity curves

3.2.3.1. Tracking on a yy-part auxiliary front without reshaped discontinuity cells. Now we are going to compute the ordinary cell-averages in discontinuity cells and then track the discontinuity curve with the reconstructed solution in smooth regions u at t_n and t_{n+1} and the reconstructed flow fluxes $f_{i+1/2}(y,t)$'s at $x_{i+1/2}$'s and $g_{j+1/2}(x,t)$'s at $y_{j+1/2}$'s in smooth regions. This is the main ingredient of our 2D front-tracking method. The tracking is implemented on auxiliary fronts. For the clarity of discussion, we are first concerned with a yy-part of front consisting of only ordinary yy-type discontinuity cells between horizontal grid lines $y = y_{j_1-1/2}$ and $y = y_{j_1+1/2}$ as shown in Fig. 3.7, and no reshaped discontinuity cells are involved. Without loss of generality, we still assume that u^- is below the curve and u^+ above the curve. We now take $a = y_{j_1-1/2}$ and $b = y_{j_1+1/2}$ in (3.3)–(3.5), and for the simplicity of discussion we drop the dependency of U, F and G on $y_{j_1-1/2}$ and $y_{j_1+1/2}$ in

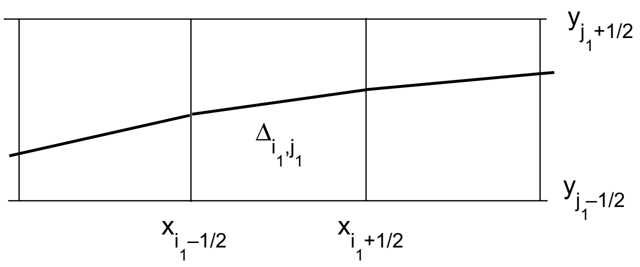


Fig. 3.7. The tracked discontinuity curve is of *yy*-part consisting of only ordinary *yy*-type discontinuity cells between horizontal grid lines $y=y_{j_1-1/2}$ and $y=y_{j_1+1/2}$.

the following discussion. It is easy to see from (3.3) that the cell-averages of $U(s(x,t))$ over the cells $(x_{i-1/2},x_{i+1/2})$'s on the x-sub-grid at t_n and t_{n+1} are the ordinary cell-averages in the discontinuity cells

$$\frac{1}{h}\int_{x_{i-1/2}}^{x_{i+1/2}} U(s(x,t_k))\mathrm{d}x = hu^k_{i,j_1}, \quad k=n,n+1,\ \ i=i_1,i_1\pm 1,\ldots; \tag{3.15}$$

therefore, computing the ordinary cell-averages and tracking the discontinuity curve then become a job of numerical simulation of (3.5) on the x-sub-grid. Integrating (3.5) over $(x_{i-1/2},x_{i+1/2})\times(t_n,t_{n+1})$ and dividing it by h, we obtain

$$hu^{n+1}_{i,j_1} = hu^n_{i,j} - \lambda(\widehat{F}^n_{i+1/2}-\widehat{F}^n_{i-1/2}) + \frac{1}{h}\int_{t_n}^{t_{n+1}}\mathrm{d}t\int_{x_{i-1/2}}^{x_{i+1/2}} G(x,t)\mathrm{d}x, \tag{3.16}$$

where according to (3.2)

$$\frac{1}{h}\int_{t_n}^{t_{n+1}}\mathrm{d}t\int_{x_{i-1/2}}^{x_{i+1/2}} G(x,t)\mathrm{d}x = \tau\left(\hat{g}^{n,+}_{i,j_1+1/2}-\hat{g}^{n,-}_{i,j_1-1/2}\right), \tag{3.17}$$

which is known from the previous computation of smooth regions, and

$$\widehat{F}^n_{i+1/2} = \frac{1}{\tau}\int_{t_n}^{t_{n+1}}\left\{\int_{y_{j_1-1/2}}^{s(x_{i+1/2},t)} f^-_{i+1/2}(y,t)\mathrm{d}y + \int_{s(x_{i+1/2},t)}^{y_{j_1+1/2}} f^+_{i+1/2}(y,t)\mathrm{d}y\right\}\mathrm{d}t, \tag{3.18}$$

which involves the unknown discontinuity positions s. Thus, the main task in the numerical simulation of (3.5) is the numerical evaluation of the flux function (3.18). As we have pointed out in the introduction, the job can enjoy many well-developed numerical methodologies for 1D conservation laws.

We are going to implement our numerical simulation of (3.5) in a Godunov fashion, and it proceeds as usual in the following three steps:

Step-R. Reconstruct $U(s(x,t_n))$ and then the discontinuity curve at t_n.
Step-E. Evaluate the fluxes (3.18) on the cell-interfaces of the x-sub-grid using the reconstructed discontinuity curve.
Step-A. Compute the ordinary cell-averages of $U(s(x,t_{n+1}))$ on the x-sub-grid at t_{n+1} and then reconstruct the discontinuity curve at t_{n+1}.

This can be implemented in many different ways with different accuracies.
First-order discretization
Step-R. The function $U(s(x,t_n))$ at t_n is reconstructed as a piecewise constant function

$$U(s(x,t)) \simeq R^n(x;U) = hu^n_{i,j_1}, \quad i=i_1,i_1\pm 1,\ldots. \tag{3.19}$$

in each cell on the x-sub-grid. Once $U(s(x,t_n))$ is reconstructed the discontinuity curve $s(x,t_n)$ can then be reconstructed by solving (3.3), in which $u^{\pm}(x,y,t_n)$, the reconstructed solution in the smooth regions on the two sides,

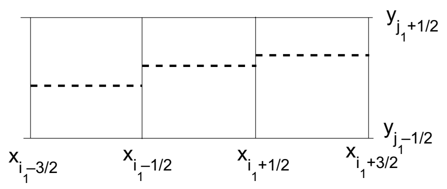


Fig. 3.8. In the first-order discretization the discontinuity curve is reconstructed as a set of horizontal line segments in all the *yy*-type discontinuity cells if the solution in smooth regions is reconstructed piecewise-constantly.

are known. As we have mentioned in the 1D case, the Eq. (3.3) admits a unique solution if h is small and $u^{\pm}$ do not vary rapidly on the two sides. In particular, if the solution in smooth regions is reconstructed piecewise-constantly, the reconstructed discontinuity curve in each discontinuity cell is a segment of horizontal line

$$s(x,t_n) = s_i^n = y_{j_1} + h\frac{u_{i,j_1}^n - u_{i,j_1}^{n,-}}{u_{i,j_1}^{n,+} - u_{i,j_1}^{n,-}}, \quad i = i_1, i_1 \pm 1, \ldots \tag{3.20}$$

as shown in Fig. 3.8. Also if (3.1) is a system and we then have three of Eq. (3.3), the reconstructed discontinuity curve in the cell should be taken as an average of certain kind of the three curve segments solved from (3.3) as in the 1D case.

Step-E. The fluxes $F(s(x,t),x,t)$ in (3.18) on cell-interfaces of the x-sub-grid are evaluated in the Euler-forward fashion in time using the flow fluxes reconstructed on the vertical cell-interfaces in smooth regions and the reconstructed discontinuity curve. That is, the fluxes on the two cell-interfaces of the cell $(x_{i-1/2}, x_{i+1/2})$ are evaluated as

$$\widehat{F}_{i\pm1/2}^n = \int_{y_{j_1-1/2}}^{s_{i\pm1/2}^n} f_{i\pm1/2}^-(y,t_n)\mathrm{d}y + \int_{s_{i\pm1/2}^n}^{y_{j_1+1/2}} f_{i\pm1/2}^+(y,t_n)\mathrm{d}y \tag{3.21}$$

with $f^{\pm}$ being the reconstructed flow fluxes on the cell-interfaces in the left and right smooth regions. In particular, if the flow fluxes are also reconstructed piecewise-constantly, we simply have

$$\widehat{F}_{i\pm1/2}^n = (s_{i\pm1/2}^n - y_{j_1-1/2})f_{i\pm1/2}^{n,-} + (y_{j_1+1/2} - s_{i\pm1/2}^n)f_{i\pm1/2}^{n,+}. \tag{3.22}$$

The discontinuity positions $s_{i\pm1/2}^n$ in (3.21) and (3.22) are approximations to $s(x_{i\pm1/2}, t_n)$, and they can be computed, for example, as

$$s_{i\pm1/2}^n = \frac{1}{2}(s_i^n + s_{i\pm1}^n). \tag{3.23}$$

However, a better way is to compute them in the following up-wind fashion. To compute $s_{i+1/2}^n$, we first compute an approximate local eigenvalue $[f]/[u]$ of Eq. (3.5). For example, it can be computed as

$$eig = \frac{1}{2}\left\{\frac{f(u_{i,j_1}^{n,+}) - f(u_{i,j_1}^{n,-})}{u_{i,j_1}^{n,+} - u_{i,j_1}^{n,-}} + \frac{f(u_{i+1,j_1}^{n,+}) - f(u_{i+1,j_1}^{n,-})}{u_{i+1,j_1}^{n,+} - u_{i+1,j_1}^{n,-}}\right\}. \tag{3.24}$$

Then $s_{i+1/2}^n$ is computed as

$$s_{i+1/2}^n = \begin{cases} s_i^n, & eig > 0, \\ s_{i+1}^n, & eig \leqslant 0. \end{cases} \tag{3.25}$$

Step-A. Once the flux functions $\widehat{F}$ are evaluated, the ordinary cell-averages in the discontinuity cells at t_{n+1} are computed as

$$u_{i,j_1}^{n+1} = u_{i,j_1}^n - \frac{\lambda}{h}(\widehat{F}_{i+1/2}^n - \widehat{F}_{i-1/2}^n) - \lambda(\hat{g}_{i,j_1+1/2}^{n,+} - \hat{g}_{i,j_1-1/2}^{n,-}), \quad i = i_1, i_1 \pm 1, \ldots \tag{3.26}$$

with $\hat{g}^{n,+}_{i_1,j_1+1/2}$ and $\hat{g}^{n,-}_{i_1,j_1-1/2}$ the two vertical numerical fluxes used in computing the smooth solution u^+ and u^-, respectively. After the ordinary cell-averages at t_{n+1} are computed in all the discontinuity cells, the discontinuity curve at t_{n+1} can then be reconstructed as in *Step-R* using the reconstructed solution in smooth regions at t_{n+1}.

Second-order discretization.

Step-R. The function $U(s(x,t_n))$ at t_n is now reconstructed as a piecewise linear function

$$U(s(x,t)) \simeq R^n(U;x) = hu^n_{i,j_1} + \xi^n_i(x-x_i), \quad i = i_1, i_1 \pm 1, \ldots. \tag{3.27}$$

in each cell of the x-sub-grid. Here ξ^n_i is the slope which approximates $\frac{\partial U(s(x,t))}{\partial x}|_{x_i}$ and can be computed as a convex combination of $\Delta^+_i u^n_{i,j_1}$ and $\Delta^-_i u^n_{i,j_1}$. Slope limiters, such as that of TVD or ENO may be used in the reconstruction for the purpose of stability. Once $U(s(x,t_n))$ is reconstructed the discontinuity curve $s(x,t_n)$ can also be reconstructed by solving (3.3) as in the first-order case. Generally, the reconstructed discontinuity curve segment will not be linear. A situation of this is shown in Fig. 3.9.

Step-E. Once the discontinuity curve is reconstructed in each cell of the x-sub-grid, we have at each cell-edge $x_{i+1/2}$ at t_n two discontinuity positions, one from the left discontinuity cell and the other from the right discontinuity cell. The discontinuity position $s^n_{i+1/2}$ can then be computed either in an average fashion as in (3.23) or in an up-wind fashion as in (3.24) and (3.25).

In the following we are going to evaluate the flux (3.18) on cell-interfaces of the x-sub-grid using midpoint integral, and to this end we need to compute $s^{n+1/2}_{i\pm1/2}$, the approximations to $s(x_{i\pm1/2}, t_{n+1/2})$. They can be computed by the so-called local Cauchy–Kowalevski procedure using (3.5) [10]. That is, by the Taylor expansion and from Eq. (3.5) we have

$$s(x,t_{n+1/2}) = s(x,t_n) + \frac{\tau}{2}\frac{\partial s}{\partial t}|_{t_n} + O(\tau^2) = s(x,t_n) - \frac{\tau}{2}\left\{\left(\frac{[f]}{[u]}\frac{\partial s}{\partial x}\right)|_{t_n} + \left(\frac{[g]}{[u]}\right)|_{t_n}\right\} + O(\tau^2). \tag{3.28}$$

Then $s^{n+1/2}_{i\pm1/2}$ can be computed by replacing the terms on the RHS of (3.28) with the reconstructed ones and truncating the $O(\tau^2)$ error. Once $s^{n+1/2}_{i\pm1/2}$ are computed, the fluxes (3.18) on cell-interfaces of the x-sub-grid are computed with a midpoint integral, that is

$$\widehat{F}^n_{i+1/2} = \left\{\int_{y_{j_1-1/2}}^{s^{n+1/2}_{i+1/2}} f^-_{i+1/2}(y,t_{n+1/2})\mathrm{d}y + \int_{s^{n+1/2}_{i+1/2}}^{y_{j_1+1/2}} f^+_{i+1/2}(y,t_{n+1/2})\mathrm{d}y. \tag{3.29}$$

Step-A. Again we can compute the cell-averages of $U(s(x,t_n))$ at t_n as in (3.26) and then reconstruct the discontinuity curve as in *Step-R* using the reconstructed solution in smooth regions at t_{n+1}.

Remark 3.1. The temporal evolution, i.e. *Step-E* and *Step-A* can also be implemented in the manner of method of lines using a Runge–Kutta discritization in time.

Remark 3.2. The front-tracking algorithm tested in the following section is a partially second-order discretization, in which the discontinuity curve is reconstructed in the first-order fashion. Nevertheless, the numerical results are quite promising.

Higher-order discretization

Step-R. We reconstruct the function $U(s(x,t_n))$ as a piecewise rth-order polynomial, and this can be accomplished by the reconstruction via primitive functions. A detailed description of the procedure can be found in

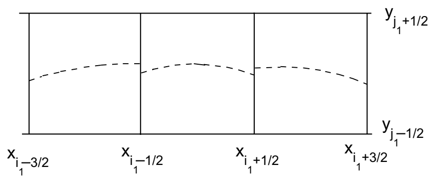


Fig. 3.9. In the second-order discretization the discontinuity curve is reconstructed as a set of curve segments in all the yy-type discontinuity cells.

[24]. Once $U(s(x,t_n))$ is reconstructed, the discontinuity curve segment in each discontinuity cell can be reconstructed by solving (3.3).

Step-E. The numerical fluxes are evaluated by using a numerical quadrature with the prediction values of $s(x_{i+1/2},t)$ at fraction time steps on cell-interfaces of the x-sub-grid. The prediction values of $s(x_{i+1/2},t)$ can still be obtained by the Taylor expansion and using Eq. (3.5), the so-called local Cauchy–Kowalevski procedure. However, to achieve the higher-order accuracy, the Taylor expansion must include higher-order terms in t, which are then evaluated by replacing the t-derivatives with the x-derivatives using Eq. (3.5) and truncating the $O(\tau^r)$ error.

Step-A. It proceeds just as in the first- and second-order discretizations.

Remark 3.3. As in the previous case, the temporal evolution, i.e. *Step-E* and *Step-A* can also be implemented in the manner of method of lines using a Runge–Kutta discretization in time.

Remark 3.4. One may use the Hugoniot condition (3.6) to do the front-tracking, discretizing it on the fixed Cartesian grid and embedding it into the computation of smooth regions as is done in the above discussion, and the resulting method is also Cartesian-grid-based. The paper [27] described a front-tracking method in 2D built in such a way and tested it on numerical examples of piecewise smooth solutions. However, the method built on the basis of Eq. (3.6) is not conservative.

3.2.3.2. Tracking on a general auxiliary front. Now we are going to discuss the implementation of tracking on a general auxiliary front. Principally speaking, if we bear in our minds Fig. 3.2(b), i.e. the discontinuity curve can be partially or even totally out of the rectangle, the tracking of a yy-part front with reshaped discontinuity cells is almost the same as that of the front without reshaped discontinuity cells. The only difference is that Eq. (3.15) may be revised in some of the cells on the x-sub-grid. For example, if we have an auxiliary front as shown in Fig. 3.10, then the cell-average of U in the grid cell $(x_{i_1+1/2}, x_{i_1+3/2})$ is

$$\frac{1}{h}\int_{x_{i_1+1/2}}^{x_{i_1+3/2}} U(s(x,t_n))\mathrm{d}x = h(u^n_{i_1+1,j_1-1} - u^{n,-}_{i_1+1,j_1-1} + u^n_{i_1+1,j_1}) \tag{3.30}$$

instead of (3.15), where $u^n_{i_1+1,j_1-1}$ is the ordinary cell-average in the yy-discontinuity cell Δ_{i_1+1,j_1-1} and $u^n_{i_1+1,j_1}$ is the cell-average in the smooth cell Δ_{i_1+1,j_1} that belongs to the right smooth region.

The computation and tracking of an xx-part of a front is analogous. Now the remaining thing is the computation and tracking in single xy-type discontinuity cells on auxiliary fronts. Principally speaking, the reconstruction in an xy-type discontinuity cell is proceeded using information extrapolated from its neighbor cells of xx- or yy-type, either ordinary or reshaped; however, certain slope limiters are involved for maintaining the stability of reconstruction. Once the discontinuity curve is reconstructed in the cell, flow fluxes across the cell-interfaces can also be evaluated, and then the ordinary cell-average can be updated in time in a finite-volume fashion, and then the discontinuity curve at t_{n+1} can be reconstructed as well. However, we

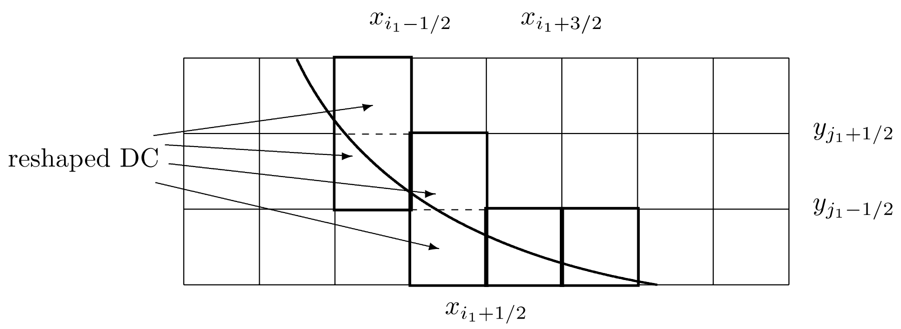


Fig. 3.10. A yy-part of front involving reshaped yy-type discontinuity cells. In this case, (3.15) should be revised as (3.30) in the grid cell $(x_{i_1+1/2}, x_{i_1+3/2})$.

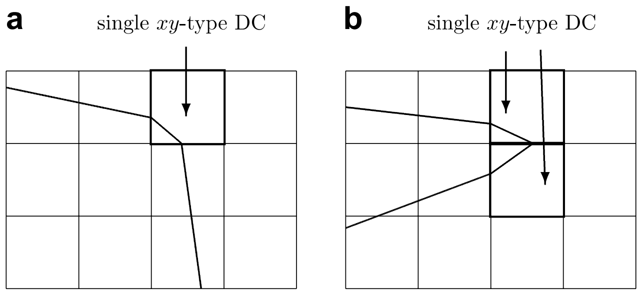


Fig. 3.11. When the tracked discontinuity curve involves corners, we may have single xy-type discontinuity cells on an auxiliary front as shown in (a) and (b).

may have situations of single xy-type discontinuity cells as shown in Fig. 3.11(a) and (b), where obviously the tracked discontinuity curve has corners in these cells. At these points, the accuracy of the reconstruction with the slope limiters degenerates, which will be clearly seen in the numerical examples in the following section. As a matter of fact, these are the only points where the current version of our front-tracking method loses accuracy.

3.2.4. Moving and collisions of discontinuities

The discontinuity curve at t_{n+1} moves to a new position, which causes some discontinuity cells to disappear and some to be generated. In our 2D method, this is accomplished by transforming the auxiliary fronts at t_{n+1} back to ordinary fronts based on the reconstructed discontinuity curves and the solution in smooth regions at this time level.

We need first to reconstruct a continuous discontinuity curve for each front. To this end, we first note that any pair of reshaped discontinuity cells share a common segment of discontinuity curve (see Fig. 3.5), and their reconstructions in the two discontinuity cells are usually different. To cope with this situation, we take the final segment as the average of the two reconstructed ones. Secondly, the discontinuity curve thus reconstructed is not continuous and it breaks at some cell-edges (see Figs. 3.8 and 3.9). However, this can be fixeded. For example, if the first-order discretization is used in the previous computation and the reconstructed segments of the curve are as in Fig. 3.8, we can then compute the discontinuity positions at cell-edges as (3.23) and link them with line segments to form a continuous and piecewise linear reconstruction for the discontinuity curve. High-order continuous reconstruction can be accomplished as well.

The transformation of an auxiliary front back to an ordinary front will involve integrals of the type $\int\int(u^+ - u^-)\mathrm{d}x\,\mathrm{d}y$ over many triangles cut off by the discontinuity curves from the fixed Cartesian grid. For example, if we have two reshaped yy-type discontinuity cells as shown in Fig. 3.5(b) and (c) at t_{n+1}, then they will be transformed back to two xy-type discontinuity cells as shown in Fig. 3.5(a). The discontinuity position on the common horizontal cell-edge of the two transformed discontinuity cells is computed as the intersection point of the discontinuity curve with the cell-edge. The ordinary cell-averages over the two transformed xy-type discontinuity cells can be computed, from the view of $\Delta_{i,j}$, as

$$\begin{aligned} u_{i,j}^{n+1} &:= u_{i,j}^{n+1} - I_1, \\ u_{i,j}^{n+1} &:= u_{i,j-1}^{n+1,-} + I_1, \end{aligned} \tag{3.31}$$

where

$$I_1 = \int\int_{T_1} (u^+(x,y,t_{n+1}) - u^-(x,y,t_{n+1}))\mathrm{d}x\,\mathrm{d}y \tag{3.32}$$

with the triangle T_1 as shown in Fig. 3.5(b). The integral I_1 can be evaluated by numerical quadratures. In doing so, the conservation properties of the solution are preserved. However, the ordinary cell-averages over the two transformed xy-type discontinuity cells can also be computed, from the view of $\Delta_{i,j-1}$, as

$$\begin{aligned} u_{i,j}^{n+1} &:= u_{i,j}^{n+1,+} + I_2, \\ u_{i,j-1}^{n+1} &:= u_{i,j-1}^{n+1} - I_2, \end{aligned} \tag{3.33}$$

where

$$I_2 = \int\int_{T_2} (u^-(x,y,t_{n+1}) - u^+(x,y,t_{n+1}))\mathrm{d}x\,\mathrm{d}y \tag{3.34}$$

with the triangle T_2 as shown in Fig. 3.5(c). To avoid favoring any one of the two discontinuity cells, in our algorithm we take the average of (3.31) and (3.33) as the final ordinary cell-averages in the two transformed xy-type discontinuity cells.

There will be other cases, such as that either or both of the discontinuity positions of an ordinary discontinuity cell move out of the grid cell, either in the same or different directions. With certain restriction on the mesh ratio $\lambda = \tau/h$, the algorithm allows the discontinuity positions move less than one cell width in each direction in a time step. The transformation can be accomplished likewise for all these cases. For example, if $\Delta_{i,j}$ is an ordinary yy-type discontinuity at t_n and one of its discontinuity position moves into the grid cell $\Delta_{i,j-1}$ at t_{n+1} as shown in Fig. 3.5(b), then both of the cells become discontinuity cells of xy-type at the late time. In this case,

$$\begin{cases} u_{i,j-1}^{n+1,-} := u_{i,j-1}^{n+1} \\ u_{i,j-1}^{n+1,+} := \text{extension data from the right region on top} \\ u_{i,j}^{n+1} \quad \text{and} \quad u_{i,j-1}^{n+1} := \text{computed from (3.31) and (3.32).} \end{cases} \tag{3.35}$$

Single xy-type discontinuity cells on auxiliary fronts may also move or split. Two situations of this kind are shown in Fig. 3.12. Nevertheless, the transformation can be accomplished likewise for these cases in the same spirit. Thanks to the Cartesian-grid-based feature of our front-tracking method, the implementation of the transformation is not quite complex.

Collisions of discontinuity curves are not addressed in this paper. As is well known, handling collisions of discontinuity curves is really a hard nut for all front-tracking methods, especially in 2D and 3D due to lack of sufficient knowledge about 2D or 3D Riemann problems. We are trying to cope with this problem also by enforcing the conservation properties of the solution, without using the knowledge of 2D Riemann problems. The research has achieved success in some simple examples, two of which were presented in [24], one is for the 2D Burger's equation and the other is for the Euler system. However, this part of the algorithm has not been well matured enough yet to be presented in this paper, and we wish to address this issue in our future papers in this series after further development of the method.

3.2.5. Propagation of waves in other characteristic fields

As in the 1D method, if (3.1) is a system of equations, there will be several characteristic fields and thus several different kinds of discontinuities. When tracking a discontinuity of a certain kind, waves in other char-

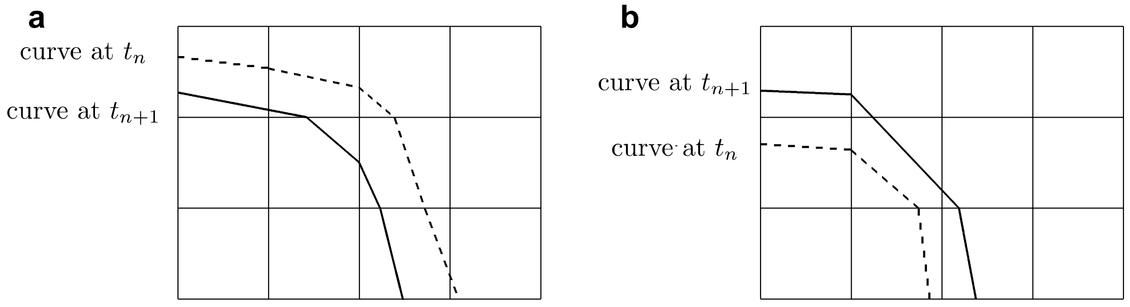


Fig. 3.12. (a) A single xy-type discontinuity cell on an auxiliary front moves and (b) a single xy-type discontinuity cell on an auxiliary front splits.

acteristic fields may propagate across it. This situation should also be treated properly in our 2D method. Principally speaking, 1D Riemann problems are solved at the discontinuity curve in the normal directions, and based on which informations related to the other characteristic fields are separated from the cell-averages in the discontinuity cell and then transferred to the corresponding sides. As in the 1D method, certain integrals should be evaluated numerically in the treatment; however, they are now over 2D areas, triangles, trapezoid and pentagons, and they can be easily computed thanks to the Cartesian-grid-based feature of the method.

3.3. Summary

Thus, we completed the description of our 2D conservative front-tracking method. Like the 1D method, the 2D method also has the following features: (1) Based on the philosophy of the Lax–Wendroff theorem, the method tracks discontinuities by enforcing the conservation properties of the solutions. This conservation feature of the method, together with the accuracies of the underlying scheme (3.10) and the numerical discretization of (3.5), guarantees the accuracy of the tracking on discontinuity curves away from corners, which will be seen in the numerical examples presented in the following section. A strict analysis of truncation error of the method under the assumption that the tracked discontinuity curve is smooth is underway and will be addressed in our future paper in the series. (2) The method is also Cartesian-grid-based. Actually, it is formulated as consisting of a 2D capturing and a 1D capturing, the former computing the solution in smooth regions and the latter tracking discontinuities. Both the 1D and 2D capturings are implemented on the same fixed Cartesian grid; therefore, the data structures involved are much simpler and the algorithm is much easier to code compared with those of other 2D front-tracking methods. Moreover, the method completely gets rid of the small-cell problem that bothers most 2D front-tracking methods. Development of an all-purposed code of the 2D method, written in Fortran90 in an object-oriented fashion and including the treatment of collisions of discontinuity curves and spontaneous shocks, is now in progress.

Finally, we would like to make a comparison of our conservative front-tracking method with the VOF methods, which track fluid interfaces also by enforcing the conservation properties of solution in some way. The difference of our method from the VOF methods is that our method tracks discontinuity curves by numerically simulating (3.5) for a *yy*-part of a discontinuity curve or its counterpart for an *xx*-part. In computation, by producing an auxiliary front for a front it divides a tracked discontinuity curves into *xx*- and *yy*-parts and then implements the simulation on each of them on the *x*- or *y*-sub-grid. In this way, the reconstruction and evolution of a tracked discontinuity curve in our method are much simpler than that of the VOF methods. We believe that this approach is applicable to the VOF methods to simplify their reconstructions and evolutions of fluid interfaces, though the VOF methods do not have connectivity between discontinuity cells.

4. Numerical experiments

In this section we present several numerical examples to show the efficiency and effectiveness of our 2D conservative front-tracking method. A second-order finite-volume scheme is employed in our method for the computation of solutions in smooth regions, an old version of TVD scheme developed in [32] for the scalar equations and a WENO scheme [14] for the Euler system, and then the solutions in smooth regions are reconstructed also with second-order accuracy. As is stated in Remark 3.2, the method tested here is partially of second-order accuracy in that discontinuity curves are reconstructed in a first-order fashion with (3.23) for the computation of their positions. However, the numerical fluxes (3.18) is evaluated second-orderly as in (3.29) with the predicted values $s_{i+1/2}^{n+1/2}$'s being computed with the Cauchy–Kowalevski procedure as described by (3.28). The continuous discontinuity curve is reconstructed as described in the second paragraph in Section 3.2.4. The transformation of auxiliary fronts back to ordinary fronts is second-order accurate with all the integrals evaluated numerically with second-order accuracy. The wave propagation in other fields is treated also second-orderly in that all the integrals are evaluated numerically with second-order accuracy. To ensure the movements of discontinuity positions less than one cell width in a time step the CFL number is taken to be 0.157 in all the examples.

Contours of the numerical solutions and the tracked discontinuity curves are displayed in some examples. Since the solutions in smooth regions can actually be reconstructed second-order-accurately, our method has much higher resolution of solution than that of capturing methods. To illustrate this, we reconstruct the solutions smooth-region-by-smooth-regionly, project the reconstructed solution onto grids 5–10 times finer than the original ones, and then display the contours of these projected solutions.

We start with the scalar linear case and are concerned with the partial differential equation of the type

$$u_t + v(x,y)u_x + w(x,y)u_y = 0, \tag{4.1}$$

where the velocity field is defined as

$$v(x,y) = -\phi_y \quad \text{and} \quad w(x,y) = \phi_x \tag{4.2}$$

with $\phi(x,y)$ the stream function of the flow field. Since

$$v_x + w_y = 0, \tag{4.3}$$

the flow is irrotational, and Eq. (4.1) can be written in the conservation form

$$u_t + (v(x,y)u)_x + (w(x,y)u)_y = 0. \tag{4.4}$$

Four problems, simple translation, solid body rotation, single vortex and deformation field as suggested in [33,34], are tested to assess the integrity and capability of our conservative front-tracking method in interface tracking. The detailed descriptions of the problems can be found in [34]. In all the test problems the solution region is the square $(0,1)\times(0,1)$ with periodic boundary conditions. As is stated at the beginning of this section, the underlying finite-volume scheme (3.10) is a second-order TVD scheme of the type developed in [32] with the temporal derivatives discretized in a predictor–corrector fashion. This is also the underlying scheme used in [24,25,27].

Example 1. *Simple translation.* The stream function is chosen to be

$$\phi(x,y) = x - y, \tag{4.5}$$

which results in a velocity field

$$u(x,y) = 1, \quad v(x,y) = 1. \tag{4.6}$$

There is a cycle C centered at (0.5,0.5) with radius 0.25. The initial value is set to be

$$u(x,y,0) = \begin{cases} 1, & (x,y) \in C, \\ 0, & (x,y) \notin C. \end{cases} \tag{4.7}$$

The circular body translates diagonally across the mesh and returns to its initial position after a time unit. The computations are carried out on 32^2, 64^2, 128^2 and 256^2 grids, respectively, and the L_1 numerical errors and convergence rate are displayed in Table 4.1.

Example 2. *Solid body rotation.* The stream function is chosen to be

$$\phi(x,y) = \frac{1}{2}(x^2 + y^2), \tag{4.8}$$

which results in a velocity field

$$u(x,y) = -y, \quad v(x,y) = x, \tag{4.9}$$

Table 4.1
L_1 numerical errors and convergence rate for the simple translation problem

Case	32^2	64^2	128^2	256^2
Error	4.738×10^{-3}	1.762×10^{-3}	5.087×10^{-4}	1.607×10^{-4}
Rate	–	1.427	1.792	1.663

Table 4.2
L_1 numerical errors and convergence rate for the solid body rotation problem

Case	32^2	64^2	128^2	256^2
Error	6.054×10^{-3}	1.989×10^{-3}	5.198×10^{-4}	1.318×10^{-4}
Rate	–	2.018	1.936	1.980

a constant-vorticity velocity field. A cycle C is centered at $(0.5, 0.75)$ with radius 0.15. The initial value is set to be

$$u(x,y,0) = \begin{cases} 1, & (x,y) \in C, \\ 0, & (x,y) \notin C. \end{cases} \tag{4.10}$$

The circular body rotates around the center of the vorticity and returns to its initial position after a π-time unit. The computations are carried out again on 32^2, 64^2, 128^2 and 256^2 grids, respectively, and the L_1 numerical errors and convergence rate are displayed in Table 4.2.

Example 3. *Single vortex.* The stream function is chosen to be

$$\phi(x,y) = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y), \tag{4.11}$$

which results in a velocity field

$$u(x,y) = -\sin^2(\pi x)\sin(2\pi y), \quad v(x,y) = \sin(2\pi x)\sin^2(\pi y), \tag{4.12}$$

a single vorticity. A cycle C is centered at $(0.5, 0.75)$ with radius 0.15. The initial value is set to be

$$u(x,y,0) = \begin{cases} 1, & (x,y) \in C, \\ 0, & (x,y) \notin C. \end{cases} \tag{4.13}$$

The circular body is deformed and stretched by the velocity field. The computations are carried out on 32^2, 64^2, 128^2 and 256^2 grids, respectively, first to the times 1, 2 and 3, respectively, and then back to the initial value by setting the two components of the velocity negative. The L_1 numerical errors and convergence rate are displayed in Table 4.3. In Fig. 4.1 we display the tracked interface at the time 3 and its return to initial data at 6 on the 128^2 grid. The computation keeps the numerical solution to be 1 inside the region surrounded by the interface and 0 outside the region.

Example 4. *Deformation field.* The stream function is chosen to be

$$\phi(x,y) = \frac{1}{4\pi}\sin\left(4\pi\left(x+\frac{1}{2}\right)\right)\cos\left(4\pi\left(y+\frac{1}{2}\right)\right), \tag{4.14}$$

which results in a velocity field

$$u(x,y) = \sin\left(4\pi\left(x+\frac{1}{2}\right)\right)\sin\left(4\pi\left(y+\frac{1}{2}\right)\right), \quad v(x,y) = \cos\left(4\pi\left(x+\frac{1}{2}\right)\right)\cos\left(4\pi\left(y+\frac{1}{2}\right)\right), \tag{4.15}$$

Table 4.3
L_1 numerical errors and convergence rate for the single vortex problem

Case	32^2	64^2	128^2	256^2
$T = 2$	5.667×10^{-3}	1.773×10^{-3}	7.427×10^{-4}	1.915×10^{-4}
Rate	–	1.676	1.494	1.956
$T = 4$	1.311×10^{-2}	4.320×10^{-3}	1.786×10^{-3}	5.714×10^{-4}
Rate	–	1.602	1.274	1.644
$T = 6$	1.669×10^{-2}	6.167×10^{-3}	2.384×10^{-3}	8.349×10^{-4}
Rate	–	1.436	1.371	1.4919

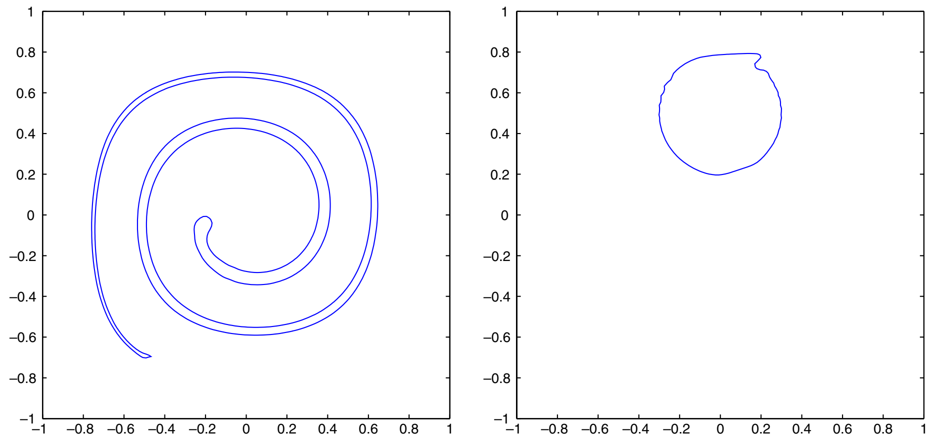


Fig. 4.1. Numerical results for single vortex problem. Left is the stretched interface at time = 3 and right is its return to the initial shape.

a quite complex velocity field. A cycle C is centered at $(0.5, 0.75)$ with radius 0.15. The initial value is set to be

$$u(x,y,0) = \begin{cases} 1, & (x,y) \in C, \\ 0, & (x,y) \notin C. \end{cases} \tag{4.16}$$

The circular body is largely deformed and stretched by the velocity field. The computations are carried out on 64^2, 128^2 and 256^2 grids, respectively, first to the time 1 and then back to the initial data as is done in the previous example to assess the numerical errors and convergence rate. In the computation on 32^2 gird a situation that three discontinuities are stacked in a discontinuity cell as shown in Fig. 3.4(b) occurs, which is still a missing case in our algorithm; thus, the run of the computation fails. The tracked interface and its return to the initial data on 128^2 grid are displayed in Fig. 4.2 and the L_1 numerical errors and convergence rates are displayed in Table 4.4.

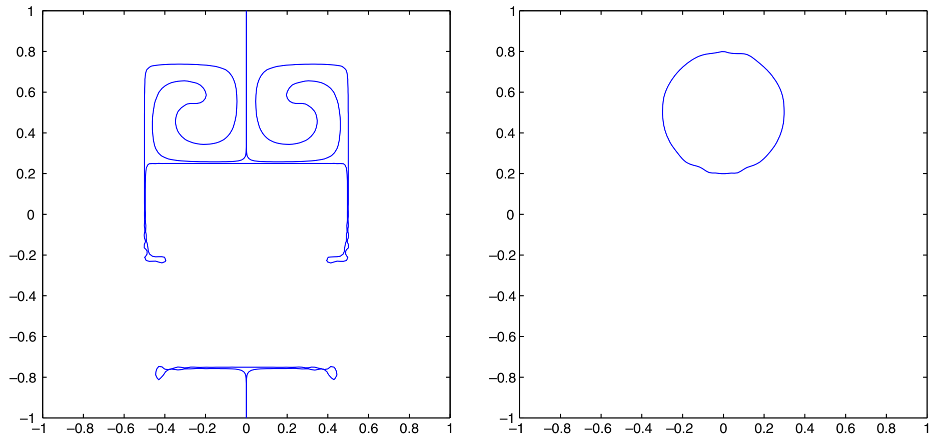


Fig. 4.2. Numerical results for deformation field problem. Left the stretched interface and right its return.

Table 4.4
L_1 numerical errors and convergence rate for deformation field problem

Case	64^2	128^2	256^2
Error	2.661×10^{-3}	8.338×10^{-4}	2.563×10^{-4}
Rate	–	1.674	1.702

It is seen that our numerical results are superior to most of the results presented in [33]. However, our results of the last two examples are not perfect yet especially compared with the ones presented in [6,7], though the methods presented in those two papers are much more expensive than ours and are not conservative. The numerical errors of our method come mainly from the spikes of the stretched and deformed discontinuity curves; the sharp corners are blunted there and sometimes wiggles occur near the corners. This is because the fronts there involve single *xy*-type discontinuity cells of the types as shown in Fig. 3.11. As is stated in the previous discussion, our front-tracking is based on Eq. (3.5) for evolution of smooth discontinuity curves; however, near corners of discontinuity curves Eq. (3.5) loses its sense. To solve the problem, it seems that we need to go back to the traditional Lagrangian front-tracking in these single *xy*-type discontinuity cells, i.e. advecting discontinuities with fluid velocities, while still maintaining the conservation property of the solutions. Modification of the method based on this idea is now underway.

We should also note in the last two examples that although our method loses accuracy near corners of the fronts, it does not severely loses structures there as the pure level-set methods [6,33], and grid-based front-tracking method [8] do. This should greatly thank to the conservation feature of the method. In our method the tracked discontinuity is reconstructed by enforcing the conservation property of solutions, which keeps pushing and dragging the discontinuity curves even near corners.

The following two examples are the Riemann problems for the 2D Burger's equation:

$$u_t + \left(\frac{1}{2}u^2\right)_x + \left(\frac{1}{2}u^2\right)_y = 0. \tag{4.17}$$

These two problems were first studied in [42] and later have been used by many authors to test their numerical methods [37]. The underlying finite-volume scheme (3.10) is again a second-order TVD scheme of the type developed in [32] with the temporal derivatives discretized in a predictor–corrector fashion as in the previous examples.

Example 5. We solve the Riemann problem with the initial data being −0.2, −1.0, 0.5 and 0.8 in the first, second, third and fourth quadrants, respectively. The solution to the problem has a shock interacting with two rarefaction waves. This shock-rarefaction interactions cause several corners, discontinuities of first derivative, on the shock curve. The shock is tracked. We compute the problem on a grid of 80^2 ($h = 0.025$) and the numerical solution at $t = 1$ is displayed in Fig. 4.3. Fig. 4.3(a) and (b) shows the contour of the projected solution on the finer grid and the tracked shock curve, respectively. For verification and comparison we compute the same problem with the underlying scheme without the tracking, and the contour of the solution is displayed in Fig. 4.3(c). It is seen that the result with the tracking is much superior to that without tracking. The shock is sharpened and the corners are better resolved in the tracking result.

Example 6. We solve the Riemann problem with the initial data being −1.0, 0.5, −0.2 and 0.8 in the first, second, third and fourth quadrants, respectively. The solution to the problem is also of a shock interacting with two rarefaction waves. However, the shock curve in this example involves a very sharp corner. Again the shock is tracked and the problem is computed on a grid of 80^2 ($h = 0.025$). The numerical solution at $t = 1$ is displayed in Fig. 4.4. Fig. 4.4(a) and (b) shows the contour of the projected solution on the finer grid and the tracked shock curve, respectively. Also for the verification and comparison we compute the same problem with the underlying scheme without the tracking, and the contour of the solution is displayed in Fig. 4.4(c). As in the previous example, the result with the tracking is much superior to that without tracking.

However, the sharp corner of the shock curve is blunted out, which is also due to the single *xy*-type discontinuity cells of the kind shown in Fig. 3.11 near the corner. Also there are two kinks on the two sides of the corner, which we believe is caused by the reason that our second-order front-tracking method is designed in a Lax–Wendroff fashion, see Section 3.2.3. Introducing certain TVD or ENO limiters in the *R-Step* may improve the result.

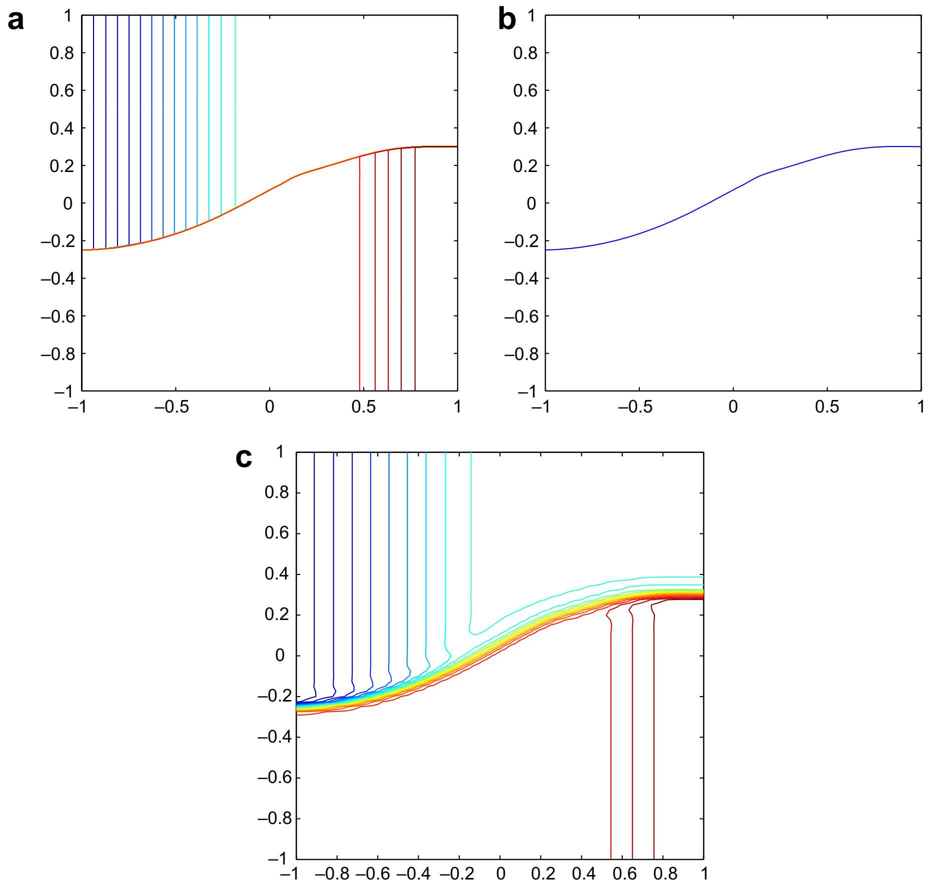


Fig. 4.3. Numerical solution for Example 5 at $t = 1$, (a) the contour of the projected solution on the finer grid computed with the tracking, (b) the tracked shock curve and (c) the contour of the solution computed without tracking.

The following five examples are for the Euler system of gas dynamics:

$$u_t + f(u)_x + g(u)_y = 0, \tag{4.18a}$$

$$u = (\rho, m^x, m^y, E)^{\mathrm{T}} \tag{4.18b}$$

$$f(u) = q^x u + (0, p, 0, q^x p)^{\mathrm{T}} \tag{4.18c}$$

$$g(u) = q^y u + (0, 0, p, q^y p)^{\mathrm{T}} \tag{4.18d}$$

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho q^2\right), \quad q^2 = (q^x)^2 + (q^y)^2, \tag{4.18e}$$

where ρ, q^x, q^y, p and E are the density, x-component and y-component of velocity, pressure and total energy, respectively, $m^x = \rho q^x$ is the x-component of momentum, $m^y = \rho q^y$ is the y-component of momentum and γ is the ratio of specific heats. In our tests $\gamma = 1.4$. The underlying scheme is a second-order WENO scheme of type developed in [14].

Example 7. This example is designed to assess the accuracy and convergence rate of our front-tracking method. A cycle C with radius 0.6 is centered at the origin. The initial data are set to be

$$u(x, y, 0) = \begin{cases} u(\rho_0, q_0, p_0), & (x, y) \notin C, \\ u(\rho_1, q_1, p_1), & (x, y) \in C, \end{cases} \tag{4.19a}$$

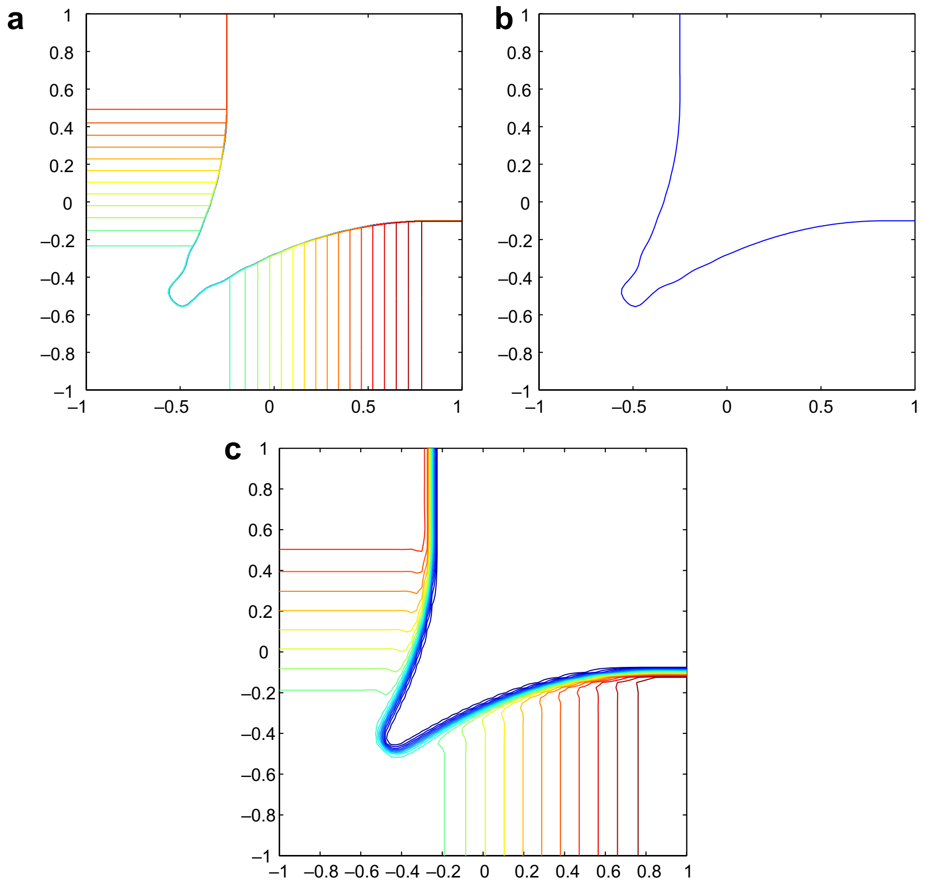


Fig. 4.4. Numerical solution for Example 6 at $t = 1$, (a) the contour of the projected solution on the finer grid computed with the tracking, (b) the tracked shock curve and (c) the contour of the solution computed without tracking.

where $\rho_0 = 1$, $q_0 = (0,0)$ and $p_0 = 1$ and $\rho_1 = 2.8803$,

$$q_x(x,y) = 1.6596\cos(\alpha)r/0.6, \quad q_y = 1.6596\sin(\alpha)r/0.6 \tag{4.19b}$$

with

$$r = \sqrt{x^2+y^2}, \quad \alpha = \tan^{-1}\left(\frac{x}{y}\right) \tag{4.19c}$$

and $p_1 = 5.2191$. Initially there is a shock on the cycle and it goes outward as the time evolves and leaves a circular weak discontinuity, discontinuity in first derivatives, behind it. We track the shock and let the weak discontinuity be computed by the underlying scheme. The so-called flow-through boundary conditions are implemented on the boundary of the square region $[-1,1]^2$ in that whenever data outside the region are required extrapolation data are provided.

We do not have the exact solution to the problem; thus, we compute a numerical solution with our front-tracking method on a grid of 1024^2 to $t = 0.12$, which is the capacity limit of our desktop used for the numerical simulation, and regard it as the exact solution for comparison. We then compute numerical solutions with our front-tracking method on a list of coarse grids of 16^2, 32^2, 64^2, 128^2 and 256^2, and compare them against the exact one to assess the numerical errors and convergence rate. The L_1 numerical errors of density and the convergence rate are displayed in Table 4.5. To see the improvement of our front-tracking method over capturing methods we compute the problem on the same list of grids with a forth-order WENO scheme without tracking, and the numerical errors and convergence rate are also displayed in the Table. It is

Table 4.5
L_1 numerical errors of density and convergence rate for circular shock problem

Case	Grid	L_1 errors	Rate
Tracking	16^2	8.670×10^{-2}	–
	32^2	2.138×10^{-2}	2.020
	64^2	9.205×10^{-3}	1.216
	128^2	2.990×10^{-3}	1.622
	256^2	1.021×10^{-3}	1.550
WENO4	16^2	7.248×10^{-1}	–
	32^2	4.209×10^{-1}	0.7840
	64^2	2.227×10^{-1}	0.9183
	128^2	1.140×10^{-1}	0.9666
	256^2	5.540×10^{-2}	1.041

seen that the results with tracking are much superior to that of the WENO4 scheme. Since there is a circular weak discontinuity which is computed by the underlying scheme, the 1.5th-order convergence rate is the best one can expect for our front-tracking method on this problem. We note that the results of the WENO4 scheme have only first-order convergence rate; besides, the numerical errors of the results of the WENOs scheme are much greater than ours.

Example 8. This problem is also designed to assess the accuracy and convergence rate of our front-tracking method. We now have two cycles, C with radius 0.6 and C_1 with radius 0.4, and both of them are centered at the origin. The initial data are set again as (4.19a) with the same ρ_0, q_0, p_0, q_1 and p_1 as in the previous example; however, ρ_1 is set to be 2.8803 outside C_1 and 0.62847 inside C_1. Thus, the solution has an outward-going contact discontinuity initially at C_1 in addition to an outward-going circular shock. Also the so-called flow-through boundary conditions are implemented on the boundary of the solution region. We do not have the exact solution to this problem either; therefore, we conduct the numerical experiment as the previous one, compute a numerical solution on a grid of 1024^2 to $t = 0.12$ and regard it as the exact solution, and then compute numerical solutions on a list of coarse grids and compare them against the exact one. The L_1 numerical errors of density and the convergence rate are displayed in Table 4.6. To see the improvement of our front-tracking method over capturing method we compute numerical solutions on the same coarse grids with WENO4 scheme without tracking, and make the comparisons against the exact one. Again, the convergence rate of our front-tracking method is about 1.5th-order and the numerical errors of our method are much smaller than that of the WENO4 scheme.

Example 9. A cycle with radius r is centered at the origin. Initially, the density of the gas is 1.0, velocity is 0.0 and the pressure is 1.0 outside the cycle and 10.0 inside the cycle. Thus, initially there are a shock and

Table 4.6
L_1 numerical errors of density and convergence rate for the circular shock and contact discontinuity problem

Case	Grid	L_1 error	Rate
Tracking	16^2	8.670×10^{-2}	–
	32^2	2.614×10^{-2}	1.707
	64^2	9.908×10^{-3}	1.400
	128^2	2.990×10^{-3}	1.668
	256^2	1.190×10^{-3}	1.390
WENO4	16^2	1.127	–
	32^2	6.412×10^{-1}	0.8134
	64^2	3.722×10^{-1}	0.7845
	128^2	2.029×10^{-1}	0.8753
	256^2	1.056×10^{-1}	0.9416

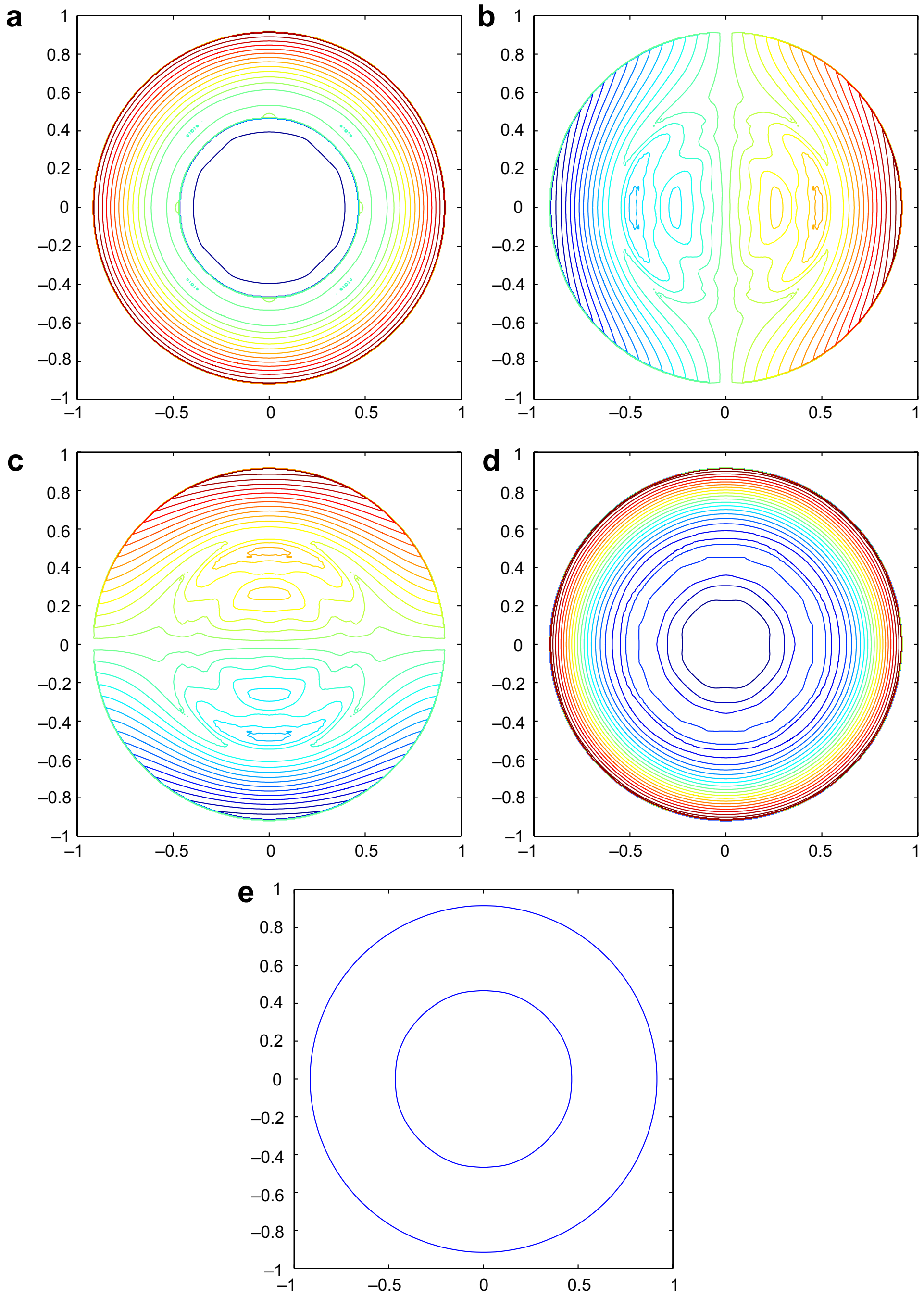
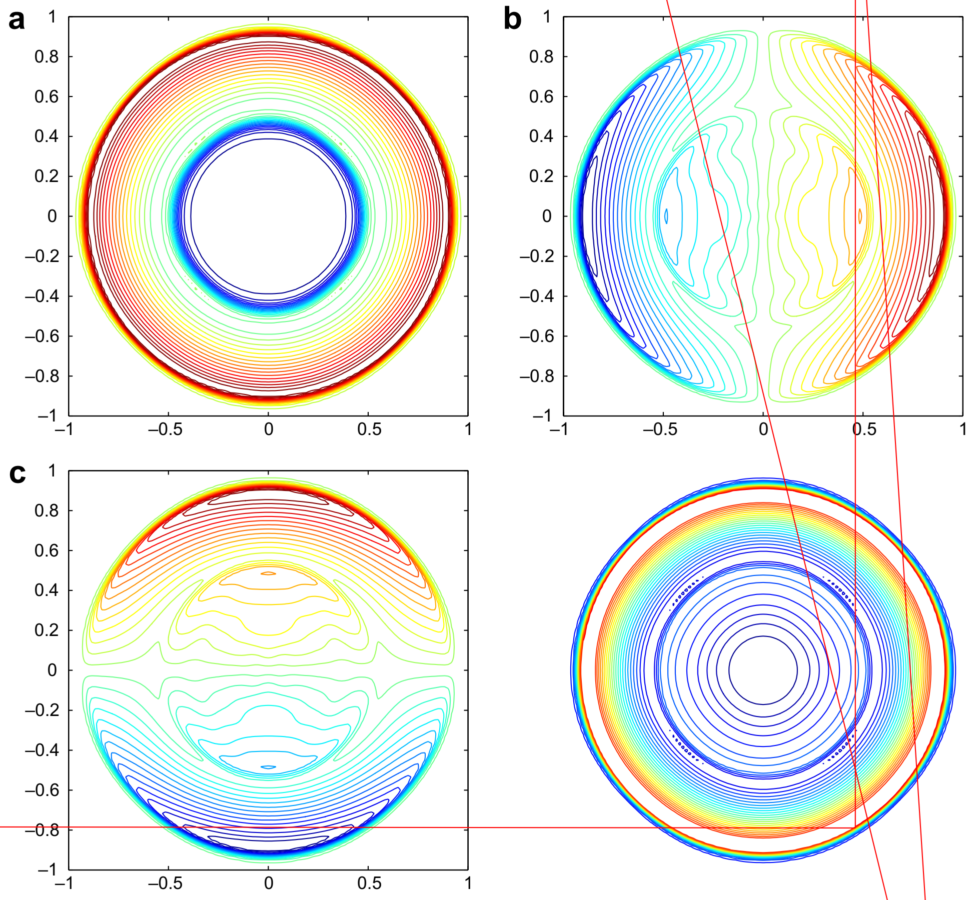
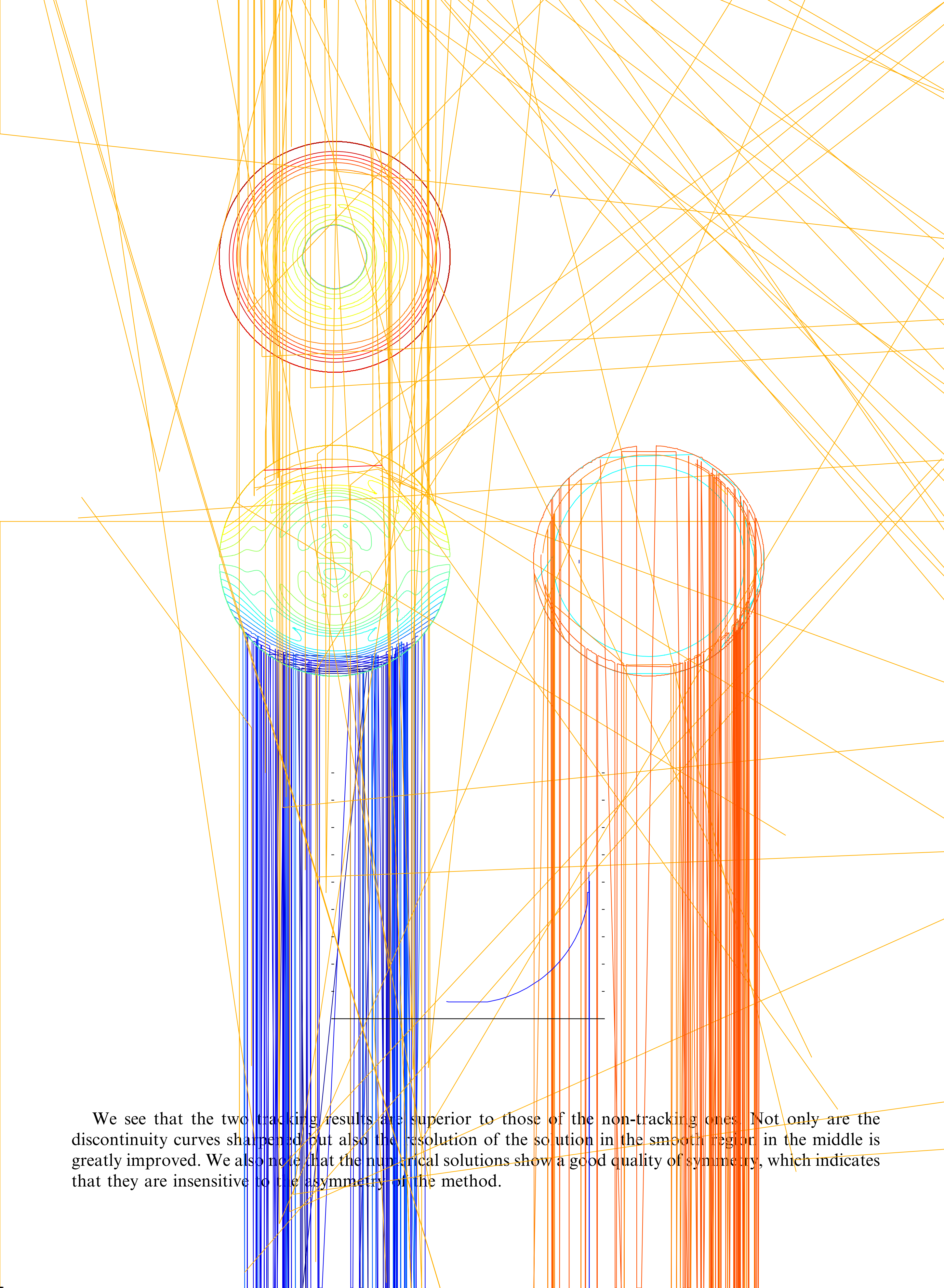


Fig. 4.5. Numerical solution for Example 7 with the front-tracking on grid of 80^2 at $t = 0.35$, $r = 0.2$; (a) density, (b) velocity, (c) pressure and (d) the tracked discontinuity curves.

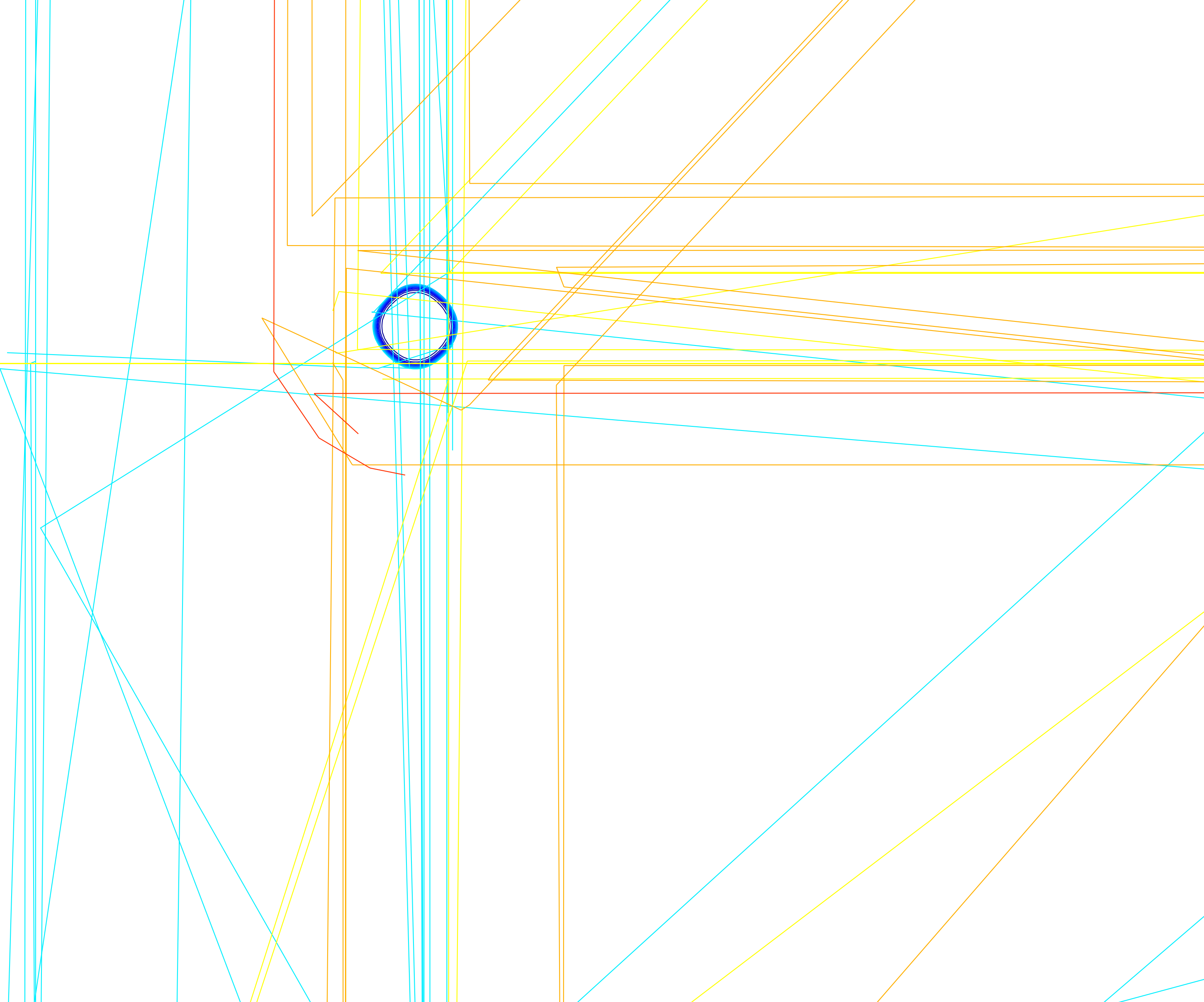
a contact discontinuity, both of which are going outwards, and a rarefaction wave, which is going inwards, on the cycle. The so-called flow-through boundary conditions are implemented on the boundary of the square region $[-1,1]^2$. This example is designed not only to assess the accuracy and resolution but also to test the efficiency of the "stack-technique" and the symmetric property of our front-tracking method. Since initially the two tracked discontinuity curves are on the same cycle, there are many stacked discontinuity cells involved in the computation at the beginning stage. Also, we should note that our front-tracking method is not symmetrical with respect to the two spatial directions. This is because the auxiliary fronts in each time step are produced alternatively in favoring x- or y-direction; therefore, beginning initially with different spatial directions favored in producing auxiliary fronts will come to different numerical results, and we would like to see if the numerical results are sensitive to the asymmetry of our method.

Two different r's are experimented. Firstly, we take $r = 0.2$ and compute the problem on a grid of 80^2 ($h = 0.025$). The numerical results at $t = 0.35$, projected on a finer grid, are displayed in Fig. 4.5. Here pictures (a), (b), (c) and (d) are contours of the density, x-velocity, y-velocity and pressure, respectively, and picture (e) is the tracked discontinuity curves, the inner one is the contact discontinuity and outer one is the shock. For verification and comparison we compute the same problem on a grid of 120^2, finer than the previous one, with the WENO4 scheme without the front-tracking, and the numerical results are displayed in Fig. 4.6 with picture (a) the density, (b) x-velocity, (c) y-velocity and (d) pressure, respectively. Secondly, we take $r = 0.1$ and compute the problem on a grid of 120^2 ($h = 0.016666$) to $t = 0.45$. The numerical results are displayed in Fig. 4.7 and the results on a grid of 160^2 without tracking for verification and comparison are displayed in Fig. 4.8.



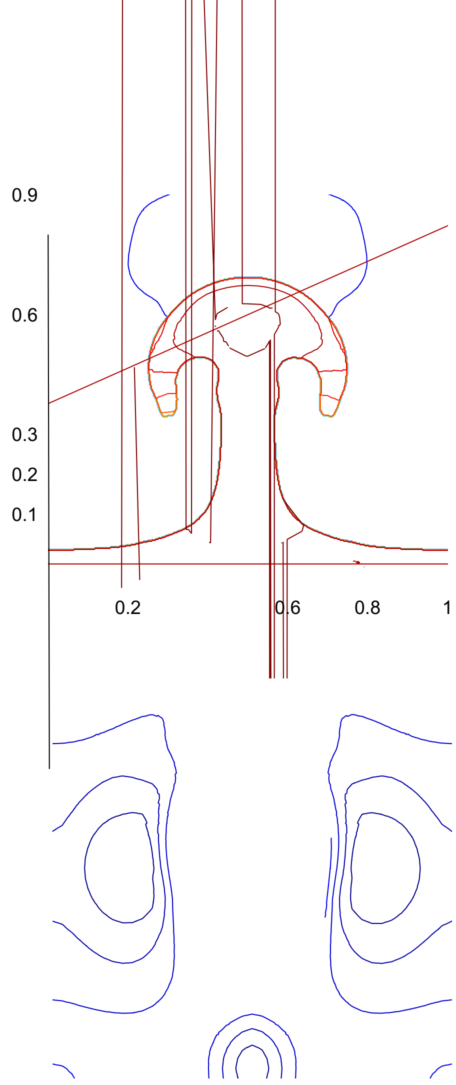


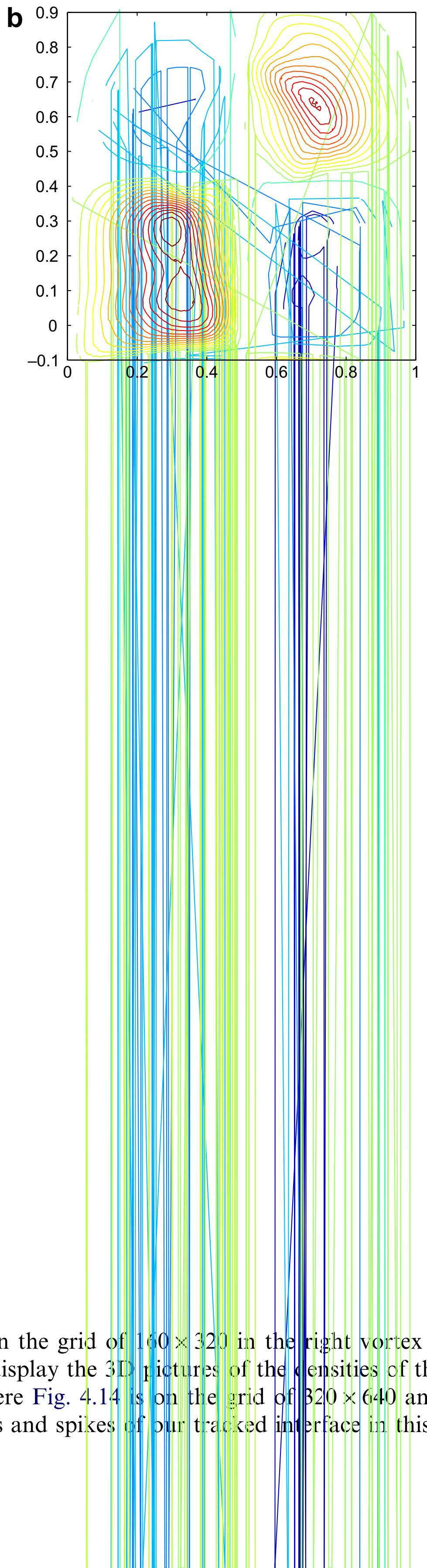
We see that the two tracking results are superior to those of the non-tracking ones. Not only are the discontinuity curves sharpened but also the resolution of the solution in the smooth region in the middle is greatly improved. We also note that the numerical solutions show a good quality of symmetry, which indicates that they are insensitive to the asymmetry of the method.



Example 10. A Mach 5.0 shock in a polytropic gas (with unshocked density 1.0) striking an interface separating two polytropic gases (both have $\gamma = 1.4$). The pre-shock contact density ratio is 1:5. The interface is sinusoidally perturbed with wavelength 1.0 and amplitude 0.1. The solution region is $(0,1) \times (-1,1)$ with flow-through boundary conditions on the top and bottom and periodic boundary conditions on the left and right. This problem is suggested in [8] and used by the authors of that paper to test their conservative and nonconservative front-tracking methods. In our test the interface is tracked and the shock is not. As in [8], We conduct the numerical experiment on 40×80, 80×160 and 160×320 grids and the numerical results, projected on a finer grid, in the region $[0,1] \times [-0.1, 0.9]$ at time $= 1.38$ are presented in Figs. 4.9–4.11, respectively. Readers are referred to the above mentioned paper for numerical comparison.

To show the improvement on resolution of our conservation front-tracking method over capturing methods, we compute the same problem with the WENO4 scheme over grids of 320×640 and of 640×1280 and the contours of the densities are displayed in Fig. 4.12, where (a) is on the grid 320×640 and (b) is on the grid of 640×1280. It is seen that the tracked interface of our method on the grid of 160×320 has almost the same structure as that of the captured one of the WENO4 scheme on the grid of 320×640 away from the vortex regions; moreover, our tracked interface has much better resolution than that of the captured one. The captured interface on the grid of 640×1280 presents Kelvin–Helmhlotz-like instability, and we believe that this KH instability is not physically real due to the reason stated in [38]. However, the tracked and the captured interfaces have somehow different structures in the vortex regions on the two sides of the mushroom. To better illustrate the situation, we display the 3D





picture of the density of our front-tracking method on the grid of 160×320 in the right vortex region, the projected one on the finer grid, in Fig. 4.13 and also display the 3D pictures of the densities of the WENO4 scheme in the same region in Figs. 4.14 and 4.15, where Fig. 4.14 is on the grid of 320×640 and Fig. 4.15 is on the grid of 640×1280. It is seen that the bubbles and spikes of our tracked interface in this region are highly resolved, while that of the captured ones, on either grids, are severely smeared and lose most of their structures in the region.

We note that the tracked interface, as well as the density, on the grid of 160 ×

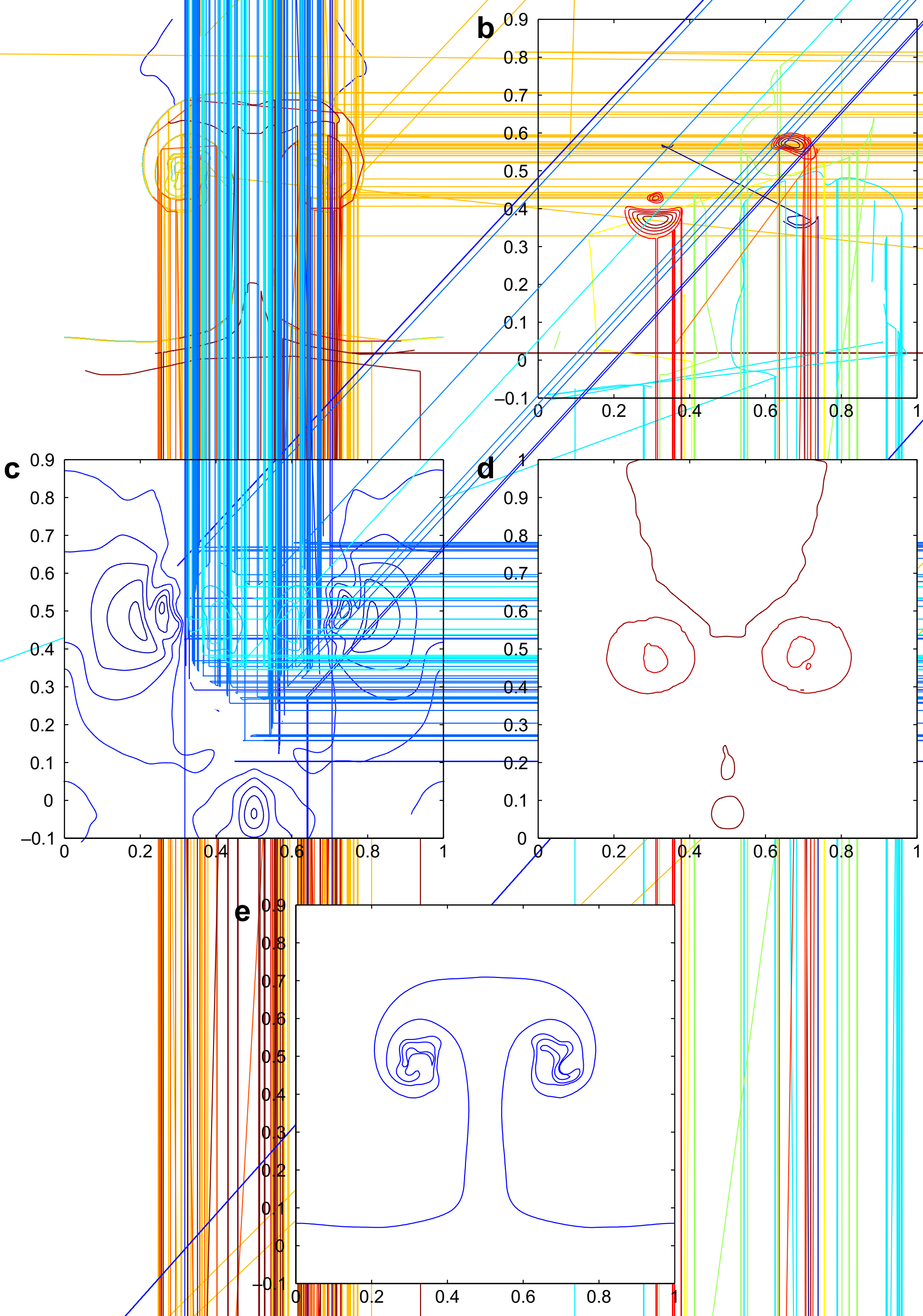
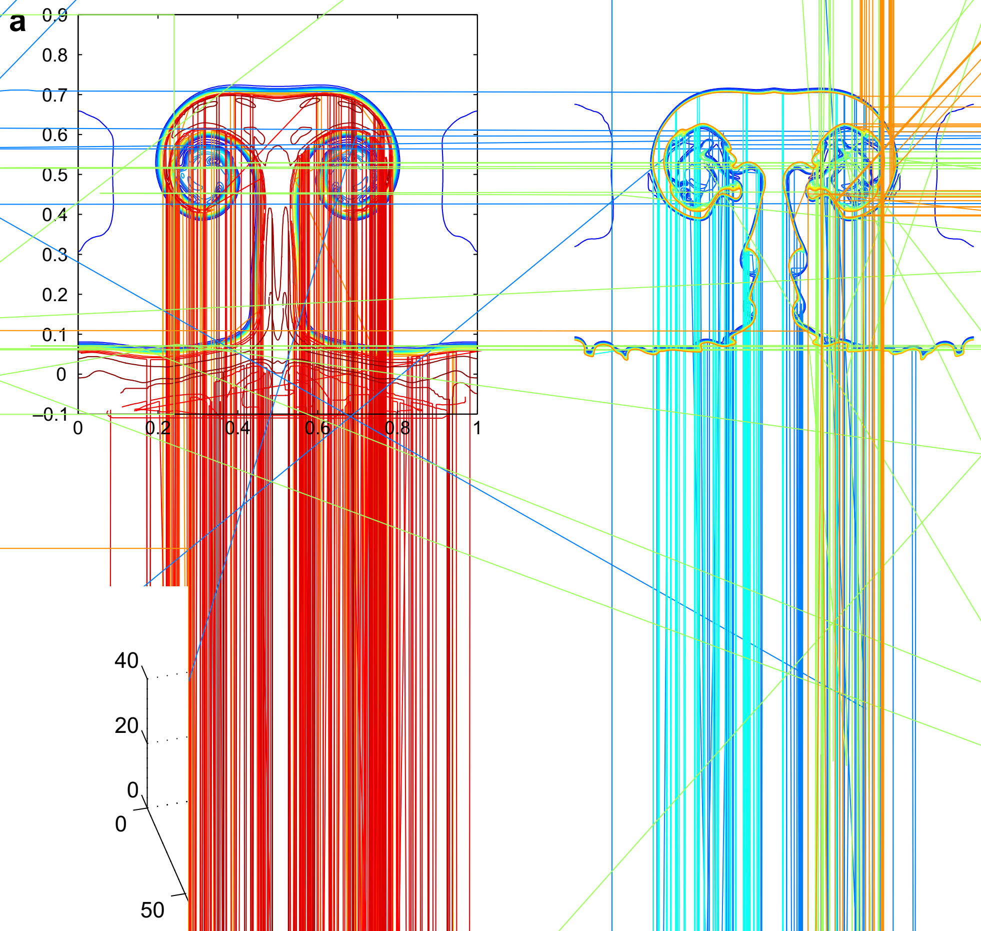


Fig. 4.11. Numerical solution for Example 10 on grid of 160 × 320 in the region [0,1] × [– 0.1,0.9] at time = 1.38; (a) density, (b) *x*-velocity, (c) *y*-velocity, (d) pressure and (e) tracked interface plot.

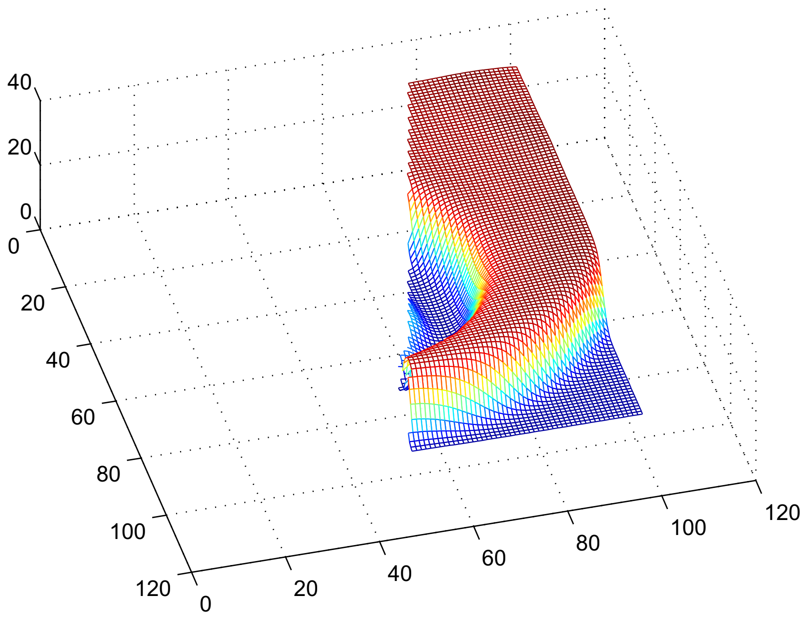
before, our front-tracking method is not symmetrical with respect to the two spatial directions. Besides, the interface in the two regions is not quite smooth due to the stretching and deformation by the velocity field; thus, there are single *xy*-type discontinuity cells of the kind as shown in Fig. 3.11 involved in the auxiliary front near the points of spikes, where our front-tracking method loses accuracy. Therefore, the tracked front becomes sensitive to the asymmetry of the method in the two regions. We wish that the future modification of



our algorithm in those single xy-type discontinuity cells on auxiliary fronts mentioned before in this section will ease this problem and improve the quality of the tracked interface there.

5. Conclusion

We have studied another important feature of our 2D conservative front-tracking method, i.e. 2D discontinuity curves are tracked in a 1D capturing fashion. The evolution of 2D discontinuity curves are locally described by 1D conservative PDE's of (3.5). Then the tracking is realized in our method by numerically simulating these 1D PDE's in a conservation fashion. Developed in such a way, our front-tracking method is Cartesian-grid-based, conservative and numerically easy to implement. As is seen in the numerical experiments, our 2D front-tracking method has a smooth geometrical description of discontinuity curves.



However, it does not have quite satisfactory resolution near corners of discontinuity curves, which is obviously due to the failure of 1D PDE's of (3.5) in describing the evolution of these corners of discontinuity curves. We believe that the traditional Lagrangian tracking should be employed near these corners while still maintaining the conservation properties of solutions. Modification of the method based on this idea is now underway, and we expect to enhance the resolution of the corners of tracked discontinuity curves with the modified method.

Acknowledgements

The author is grateful for fruitful discussions with Prof. J. Glimm and Prof. X.L. Li. Some of the numerical examples presented in Section 4 were suggested by them and the tests were done when the author was visiting the Department of Applied Mathematics and Statistics at SUNY at Stony Brook. Thanks also go to the referees for their valuable comments and suggestions, which have helped to improve the clarity of this work, and to Prof. Zhang Xijiu with School of Foreign Languages, Shanghai University for his valuable help in improving and smoothing the English writing. Finally, this work was supported by the China National Science Foundation Grant No. 10171063 and Shanghai Pu Jiang Program [2006] 118.

References

[1] T.D. Aslam, A level set algorithm for tracking discontinuities in hyperbolic conservation laws II: system of equations, J. Sci. Comput. 19 (2003) 37–62.
[2] T.D. Aslam, A level set algorithm for tracking discontinuities in hyperbolic conservation laws I: scalar equations, J. Comput. Phys. 167 (2) (2001) 413–438.
[3] I.-L. Chern, J. Glimm, O. McBryan, B. Plohr, S. Yaniv, Front tracking for gas dynamics, J. Comput. Phys. 62 (1986) 83–110.
[4] I.-L. Chern, P. Colella, A conservative front tracking method for hyperbolic system of conservation laws, LLNL Rep. No. UCRL-97200, 1987.
[5] P. Colella, P.R. Woodward, The numerical simulation of two-dimensional fluid flow with strong shocks, J. Comput. Phys. 54 (1984) 115–173.
[6] D. Enright, R. Fedwik, J. Ferziger, I. Mitchell, A hybird particle level set method for improved interface capturing, J. Comput. Phys. 183 (2002) 83–116.
[7] J. Du, F. Brain, J. Glimm, X. Jia, X. Liu, Y. Liu, L. Wu, A simple package for front-tracking, J. Comput. Phys. 213 (2006) 613–628.
[8] J. Glimm, X.L. Li, Y.J. Liu, Z.L. Xu, N. Zhao, Conservative front-tracking with improved accuracy, SIAM J. Numer. Anal. 41 (5) (2003) 1926–1947.
[9] J. Glimm, M.J. Graham, J. Grove, X.L. Li, T.M. Smith, D. Tan, F. Tangerman, Q. Zhang, Front tracking in two and three dimensions, Comput. Math. Appl. 35 (1998) 1–11.
[10] A. Harten, B. Engquist, S. Osher, S.R. Chakravarthy, Uniformly high order accurate essentially non-oscillatory scheme III, J. Comput. Phys. 71 (1987) 231–303.
[11] C. Hirt, B. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201.
[12] J. Hu, D. Mao, Realization of a front-tracking method based on conservation for one-dimensional scalar conservation law with nonconvex flux, Comm. Appl. Math. Comput. 20 (2006) 1–9 (in Chinese).
[13] J. Hu, Realization of a front-tracking method based on conservation for one-dimensional scalar conservation law with nonconvex flux, Master's thesis, No. 11903-20720648, Shanghai University (in Chinese).
[14] G.S. Jiang, C.W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[15] M. Kang, R. Fedkiw, X. Liu, A boundary condition capturing method for multiphase incompressible flow, J. Sci. Comp. 15 (2000) 323–360.
[16] P.D. Lax, B. Wendroff, Systems of conservation laws, Commun. Pure Appl. Math. 10 (1957) 217–237.
[17] R.J. LeVeque, K.M. Shyue, Two dimensional front tracking based on high resolution wave progation methods, J. Comput. Phys. 123 (1996) 354–368.
[18] R.J. LeVeque, Numerical Methods for Conservation Laws, Birkhauser-verlag, Basel, Boston, Berlin, 1990.
[20] Y. Liu, D. Mao, Further development of a conservative front-tracking methods for systems of conservation laws in one space dimensions, J. Sci. Comput. 28 (2006) 85–119.
[21] Y. Liu, A robust conservative front-tracking method in one space dimension, Shanghai University Doctoral dissertation No. 11903-01810003 (in Chinese).
[22] Y. Liu, D. Mao, Program realization of a conservation front tracking method on scalar conservation law, Appl. Math. Comput. Math. 15 (1) (2001) 10–18 (in Chinese).
[23] A. Majda, Compressible fluid flow and systems of conservation laws in several space variables, Appl. Math. Sci. 53 (1984).
[24] D. Mao, Towards front tracking based on conservation in two space dimensions, SIAM J. Sci. Comput. 22 (1) (2000) 113–151.
[25] D. Mao, Toward robust front-tracking tracking, based on conservation, in: N. Mastorakis (Ed.), Problems in Modern Applied Mathematics, World Scientific and Engineering Society press, 2000, pp. 265–269.
[26] D. Mao, A shock tracking technique based on conservation in one space dimension, SIAM J. Numer. Anal. 32 (1995) 1677–1703.
[27] D. Mao, A treatment of discontinuities for finite difference method in the two dimensional case, J. Comput. Phys. 104 (1993) 377–397.
[28] D. Mao, A treatment of discontinuities for finite difference methods, J. Comput. Phys. 103 (1992) 359–369.
[29] D. Mao, A treatment of discontinuities in shock-capturing finite difference methods, J. Comput. Phys. 92 (1991) 422–455.
[30] D. Mao, A treatment for discontinuities, in: B. Engquist, B. Gustafsson (Eds.), Proceedings of the Third International Conference on Hyperbolic Problem, Uppsala, Swedem, 1990, Studentlitteratur, Sweden, 1991.
[31] D. Mao, A difference scheme for shock calculation, J. Comput. Math. 3 (1985) 356–382 (in Chinese).

[32] S. Osher, S. Chakravarthy, Very high order accurate TVD schemes, IMA Volume in Mathematics and Its Applications, 2, Springer-Verlag, New York/Berlin, 1986, pp. 229–274.

[33] W.J. Rider, D.B. Kothe, Strecthing and tearing interface tracking methods, AIAA-95-1717, Presented at the 12th CFD Conference, San Diago, June 20th, 1995.

[34] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (1997) 112–152.

[36] S. Shin, D. Juric, Modeling three-dimensional multiphase flows using a level contour reconstruction method for front tracking without connectivity, J. Comput. Phys. 180 (2002) 427–470.

[37] C.W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes II, J. Comput. Phys. 83 (1989) 32–78.

[38] J. Shi, Y. Zhang, C.W. Shu, Resolution of high order WENO schemes for complicated flow structures, J. Comput. Phys. 186 (2003) 690–696.

[39] D.J. Torres, J.U. Brackbill, The point-set method: front-tracking without connectivity, J. Comput. Phys. 165 (2000) 620–644.

[40] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Yan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2001) 708–759.

[41] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible multi-fluid flows, J. Comput. Phys. 100 (1992) 25.

[42] D.H. Wagner, The Riemann problem in two space dimensions for a single conservation law, SIAM J. Math. Anal. 14 (1983) 534–559.