Numerical and Experimental Studies of Shock Wave Interaction with Bodies

by
Mingyu Sun

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Engineering)
in Tohoku University
1998

Doctoral Committee:

Professor Kazuyoshi Takayama
Professor Kazuhiro Nakahashi
Professor Keisuke Sawada
Associate Professor Satoru Yamamoto
Associate Professor Zonglin Jiang

Contents

1 Introduction **1**

References 7

2 On the Lax-Wendroff scheme with conservative smoothing **9**

2.1 Introduction 9

2.1.1 Total variation diminishing (TVD) scheme 10

2.1.2 The Lax-Wendroff scheme 11

2.2 The scalar case 12

2.2.1 The TVD Lax-Wendroff scheme via limiter 12

2.2.2 The TVD Lax-Wendroff scheme via nonlinear artificial viscosity 13

2.2.3 Numerical experiments 28

2.3 Extension to system of one-dimensional conservation laws 29

2.3.1 Extension via the local characteristic approach 32

2.3.2 Extension via scalar artificial viscosity 34

2.3.3 Numerical experiments 38

2.4 Extension to system of multi-dimensional conservation laws . . . 40

2.5 Summary 49

References 52

3 An essentially vectorizable data structure for some unstructured grids **54**

3.1 Introduction 54

3.1.1 Structured and unstructured grids and data structures . 54

3.1.2 Mesh adaptation 57

3.1.3 Vector and parallel processing 58

3.1.4 The essentially vectorizable data structure 60

3.2 The cell-face data structure 60

3.2.1 Optimization of global operations 61

3.2.2 Optimization of local operations 63

3.3 Mesh adaptation under the cell-face data structure 68

3.4 Summary 71

References 74

4 Two-dimensional vectorizable adaptive solver (VAS2D) **76**

4.1 Introduction 76

4.2 Automated quadrilateral mesh generation 79

4.2.1 Outline of the mesh generation algorithm 79

4.2.2 Grid examples 82

4.3 Data structure 82

4.4 Adaptation procedure 85

4.4.1 Criteria for adaptation 85

4.4.2 Strategy for adaptation 87

4.4.3 The conservative variables at new cells 92

4.4.4 Efficiency of vector processing of the adaptation 95

4.5 Unstructured flow solver 95

4.5.1 The two-dimensional conservation laws 95

4.5.2 Finite volume method 98

4.5.3 Least square method 101
4.5.4 Evaluation of fluxes through cell faces 102
4.5.5 Boundary conditions 106
4.5.6 Flow solver under the cell-edge data structure 110
4.6 Evaluation of the VAS2D 111
4.6.1 Efficiency of vector and parallel processing 111
4.6.2 CPU time distribution on subroutines 113
4.6.3 Cartesian mesh *vs.* adaptive mesh 114
4.7 Numerical results 117
4.7.1 Shock diffraction 117
4.7.2 Steady supersonic channel flows 123
4.7.3 Shock / circular-cylinder interaction 125
4.7.4 Shock / square-cylinder interaction 125
4.7.5 Shock / boundary layer interaction 128
4.7.6 Shock wave moving in a nozzle 133
4.8 Remarks on three-dimensional extension 133
4.9 Summary 137
References 139

5 Experimental and numerical study of shock wave focusing 142
5.1 Introduction 142
5.2 Experiment and computation 144
5.2.1 Experimental setup 144
5.2.2 Numerical methods 146
5.2.3 Quantitative image data analyses of interferograms 147
5.3 Formation of reflected shock wave 150
5.4 Shock wave focusing and collision of triple points 151
5.5 Summary 158

References 160

6 Conclusions **162**

Acknowledgement **164**

List of Figures

2.1 Geometrical interpretation of scheme (2.21). The coefficient μ is a distance from the initial profile to the one after smoothing. The distance is normalized by the local jump between two neighboring nodes, say c and d. 17

2.2 Five representative stages of ϕ 20

2.3 Sketch of the transition region between the smoothing scheme and the Lax-Wendroff scheme. F^S and F^{LW} denote the fluxes of the two schemes, respectively. 22

2.4 Convection of a square wave, $\sigma = 0.5$, 200 steps. **a.** results of the first-order upwind scheme and the Lax-Wendroff scheme; **b.** results of smoothing TVD scheme and Roe's symmetric TVD scheme using limiter (2.19). 30

2.5 Convection of a square wave under different CFL numbers by the smoothing TVD scheme **a.** smoothing TVD A; **b.** smoothing TVD B. 1000, 200 and 112 steps for $\sigma = 0.1, 0.5, 0.9$ respectively. . 31

2.6 Numerical results obtained by using two scalar coefficients: **a.** scalar coefficient (2.83), denoted by $\text{Max} - \mu$; **b.** scalar coefficient (2.84), denoted by $\text{Min} - \mu$. 35

2.7 TVD region of artificial viscosity coefficient μ as a function of σ. **a.** The proposed artificial viscosity coefficient μ defined in (2.21), or more clearly in (2.23); **b.** The artificial viscosity coefficient μ defined by (2.87), a similar coefficient was used by Davis. In figure **a**, the minimum coefficient is a monotone function of the CFL number $|\sigma|$; while there is an extreme value for $|\sigma| = 1/2$ in figure **b**. A trouble of the existence of the extreme value is that defining μ according to a fast wave or larger $|\sigma|$ is not sufficient for slow waves in solving the system of the conservation laws. 37

2.8 Sod's problem solved by scheme "Roe", $\alpha = 1.0$: **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$. 39

2.9 Sod's problem solved by scheme STVD A, $\alpha = 1.0$: **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$. 41

2.10 Sod's problem solved by scheme STVD B, $\alpha = 1.0$: **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$. 42

2.11 Sod's problem solved by Harten's TVD scheme. **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$. 43

2.12 Sod's problem solved by scheme STVD A, $\alpha = 1.2$, $\sigma = 0.8$. .. 44

2.13 Computational stencil for the proposed scheme on the Cartesian grid: **a.** stencil for evaluating the flux through interface AB; **b.** stencil for the control volume (i, j), bounded by ABCD. 44

2.14 A locally one-dimensional wave lying on a two dimensional control volume. 47

2.15 Decomposition of the artificial viscosity coefficient according to the wave direction. 49

2.16 A shock wave with $M_s = 2$ diffracting over a 90° corner: $32 \times 32 \times 3$ cells, CFL=0.7, $\alpha = 1.2$. **a.** the result obtained by using the conservative variables; **b.** the result obtained by using the primitive variables. 50

3.1 Illustration of structured grids: **a.** rectangular structured grid; **b.** triangular structured grid. 55

3.2 The ratio of the number of cells and vertices for different cells: **a.** triangular grid, **b.** quadrilateral grid, **c.** hexagonal grid. One new vertex introduces two triangular cells, or one quadrilateral, but one new hexagonal cell requires the addition of two vertices, as shown by dashed lines. 56

3.3 Illustration of local mesh adaptation of a cell. The adaptation is always dividing a cell or merging a few subcells, then a cell-centered data structure is preferred. 63

3.4 Illustration of generating a cell-face data structure for a quadrilateral grid: **a.** an arbitrarily unstructured quadrilateral grid; **b.** roads inside all quadrilateral cells; **c.** roads outside of the grid and a path to trace all cells shown by arrows; **d.** a definition of face directions according to the road direction. 64

3.5 Ordering adjacent connectivity by the direction of the road: **a.** labeling two neighboring cells of a face *left cell* and *right cell*; **b.** numbering all faces of a cell. 66

3.6 Possible order of three dimensional cells: **a.** for a tetrahedron; **b.** for a hexahedron. 66

3.7 Illustration of orders of geometry required by some numerical methods: **a.** contour integral along outside neighboring cells, the order of these cells is useful; **b.** surface integral over a surface consisting of six neighboring cells, the information of which three cells that form a valid triangular surface is useful. 69

3.8 The order of face for quadrilaterals 69

3.9 Illustration of hanging nodes: **a.** hanging node H on an adaptive quadrilateral grid; **b.** hanging node H on an adaptive triangular grid. Dashed lines indicate possible connections to remove the hanging nodes. The present cell-face data structure will leave the hanging nodes intact. 71

3.10 Illustration of the direction of the outside subfaces: **a.** face directions of a cell; **b.** directions of the outside subfaces. The directions of subfaces are the same as these of their unsplit faces. 71

3.11 A vectorizable adaptation strategy for a hexahedral cell 72

3.12 Illustration of four possible adaptation strategies under the cell-face data structure. 72

4.1 The overview of the algorithm VAS2D 78

4.2 The overview of the quadrilateral mesh generation algorithm (QuadMesh) 80

4.3 Illustration of joining-front **a.** Joining a multiply connected front; **b.** Joining a simply connected front. A new cell is created between front(s); the new cell changes the connectivity of front. 81

4.4 Unstructured quadrilateral mesh generated by the QuadMesh: a 30° wedge, 640 cells 82

4.5 Unstructured quadrilateral mesh generated by the QuadMesh: a half-circle, 2112 cells 83

4.6 Unstructured quadrilateral mesh generated by the QuadMesh: a nozzle, 2169 cells 83

4.7 Unstructured quadrilateral mesh generated by the QuadMesh: an oblique square in a tube, 3316 cells 83

4.8 Cell-edge data structure: **a.** Cell information, a cell points to four edges; **b.** Edge information, an edge points to two neighboring cells. 83

4.9 Sketch of an interface: interface AB and its two neighboring cells i and j; $i-1$ and $j+1$ are ghost cells. 86

4.10 Refinement strategy: father cell ABCD is divided into four sons EBFO, OFCG, HOGD and AEOH. 88

4.11 The levels of refinement. Suppose cell ABCD is one of the coarsest cells; the level of it is then 0. The refinement level of a son equals to that of his father plus 1. In the figure the level of cell 1, 2, 3 and 4 is 1; the level of cell 5, 6, 7 and 8 is 2, and so on. It is not allowed that the difference between levels of two neighboring cells is more than 1. Refining cell 5 is then not allowed. 89

4.12 Directional refinement inside the boundary layer. A highly stretched mesh can be generated by repeatedly refining cells along the viscous boundary. 90

4.13 Illustration of pre-refinement. Cells in column c cannot be refined to capture the coming shock wave, because the level of refinement in column d is lower. The cells in column d has to be pre-refined. 90

4.14 Memory organization 91

4.15 Illustration of the active mother edge. Active mother edge AB is referred by cell i, then information of it should be updated. .. 92

4.16 Flow variables at new cells. **a**. in refinement procedure, flow variables of new sons are interpolated from these of their father; **b**. in coarsening procedure, flow variables of a father cell are volume-weighted average of these of its "dead" sons. Both the interpolation and the average preserve conservation. 94

4.17 Vector efficiency of refinement and coarsening procedure 94

4.18 General form of the conservation laws 96

4.19 Control volume 100

4.20 Stencil of the local gradient. **a.** the averaged gradient (4.35); **b.** the modified gradient (4.37). The modified gradient uses a more compact stencil. 105

4.21 Schematic of the treatment of boundary conditions 108

4.22 Vector efficiency of the VAS2D with the Euler solver using different refinement levels 113

4.23 Parallel efficiency measured by Cray performance tool *atexpert* 114

4.24 CPU time consumption by main subroutines: **a.** the VAS2D with the Euler solver; **b.** the VAS2D with the Navier-Stokes solver. The data is gathered by Cray performance tool *flowtrace*. 115

4.25 Shock diffraction using different levels of refinement: $M_s = 1.6$, CFL = 0.7, : **a.** No adaptation, 3072 cells, CPU time = 0.592s; **b.** 1-level refinement, 4773 cells, CPU time = 1.98s; **c.** 2-level refinement, 7695 cells, CPU time = 6.09s..... 118

4.26 Shock diffraction, 3-level refinement, CFL = 0.7: **a.b.** $M_s = 1.3$, 12174 cells, CPU time = 18.9s, $\alpha_f = 0.02$; **c.d.** $M_s = 1.6$, 13077 cells, CPU time = 20.5s; **e.f.** $M_s = 2.0$, 16026 cells, CPU time = 22.3s. 120

4.27 Shock diffraction, 3-level refinement, CFL = 0.7: **a.b.** $M_s = 4.0$, 20595 cells, CPU time = 23.0s; **c.d.** $M_s = 8.0$, 20817 cells, CPU time = 22.7s; **e.f.** $M_s = 16.0$, 20349 cells, CPU time = 22.4s. 121

4.28 Shock wave diffraction for $M_s = 1.475$: **a.** experimental photo obtained by holographic interferometry; **b.** numerical isopycnics obtained by the VAS2D with the Euler solver. 122

4.29 Steady flow in a tunnel with 15° bump: **a.** $M = 2$, $t = 4$, with 1-level uniform refinement; **b.c.** $M = 2$, $t = 4$, 3-level refinement, 11338 cells, 1670 steps, CPU time = 47.8s; **d.e.** $M = 4$, $t = 2$, 3-level refinement, 11386 cells, 1381 steps, CPU time = 47.4s. 124

4.30 Shock / circular-cylinder interaction, isopycnics and corresponding adaptive meshes. Incident shock Mach number $M_s = 1.7$, 4-level refinement, CFL = 0.7. **a.b.** $t = 0.0$, 9735 cells, CPU time = 3.0s; **c.d.** $t = 1.0$, 12471 cells, CPU time = 18.6s. 126

4.31 Shock / circular-cylinder interaction, isopycnics and corresponding adaptive meshes (Continued). **e.f.** $t = 2.0$, 20253 cells, CPU time = 47.4s; **g.h.** $t = 3.0$, 29790 cells, CPU time = 101.5s, 1800 steps. 127

4.32 Shock / square-cylinder interaction, isopycnics and corresponding adaptive meshes. Incident shock Mach number $M_s = 2$, 3-level refinement, CFL = 0.7. **a.b.** $t = 0.0$, 8149 cells, CPU time = 4.3s; **c.d.** $t = 1.0$, 22981 cells, CPU time = 29.6s. 129

4.33 Shock / square-cylinder interaction, isopycnics and corresponding adaptive meshes (Continued). **e.f.** $t = 2.0$, 38383 cells, CPU time = 82.0s; **g.h.** $t = 3.0$, 48334 cells, CPU time = 147.6s, 1767 steps. 130

4.34 Shock wave interaction with an oblique rectangular. **a.** experimental photo obtained by holographic interferometry; **b.** numerical isopycnics obtained by the VAS2D with the Euler solver. 131

4.35 Validation of the Navier-Stokes solver. The theory of boundary layer states that for a flat wall $\frac{\tau_w}{\sqrt{u^3/xRe}}$ is 0.332. The numerical values along the flat wall are shown by circles. x is a distance mesured from the tip of the flat wall. 132

4.36 Shock / boundary layer interaction, $M_s = 5$, $T_w = 298k$; 6-level refinement, CFL = 0.5, $\alpha_{bl} = 1.4$: **a.b.** 32502 cells, 18832 steps, $t = 25\mu s$; **c.d.** 59205 cells, 25853 steps, $t = 50\mu s$. 134

4.37 Shock wave motion in a nozzle for $M_s = 2.8$. CFL = 0.7, **e** consists of 36591 cells. 135

4.38 Shock wave motion in a nozzle for $M_s = 2.8$: **a.** experimental photo obtained by holographic interferometry; **b.c.** numerical isopycnics and the corresponding grids obtained by the VAS2D. **b** is the result of the Euler solver, and **c** the Navier-Stokes solver. 136

5.1 Experimental setup and optical arrangement 145

5.2 A 150 mm wide circular reflector connected to the 60 mm × 150 mm shock tube 146

5.3 The initial quadrilateral grid for the VAS2D..... 147

5.4 Comparison between an interferogram and the corresponding numerical results of the VAS2D: (a) interferogram for $M_s = 1.502 \pm 0.002$ and at $t = 1114 \pm 8\mu s$; and (b) numerical isopycnics (upper half) and the corresponding adaptive grid (lower half) for $M_s = 1.5$ and at $t =$ 1113 μs. 149

5.5 Sequential numerical isopycnics and adaptive grids of shock wave propagation in the circular reflector for M_s=1.5. 152

5.6 Shock wave focusing for $M_s = 1.50$, $P_0 = 26.7$ kPa: **a.b.** -25 μs; **c.d.** 0 μs; **e.f.** 25 μs : **a.c.e.** are experimental interferomgrams, and **b.d.f.** are isopycnics and corresponding adaptive grids obained by the VAS2D 153

5.7 Sketch of the shock wave system in Fig. 5.6ae, corresponding to Fig. 5.7a and b respectively. 154

5.8 Comparison of density distributions along the centerline: **a.** $t = -25\mu s$, corresponding to Fig. 5.6a; **b.** $t = 0$, corresponding to Fig. 5.6b. 156

5.9 Comparison of density distributions along the centerline (Continued): **c.** $t = 25\mu s$, corresponding to Fig. 5.6c; **d.** numerical density distributions at the three time instants. 157

Chapter 1

Introduction

Flow fields including shock wave interaction with bodies are characterized by disparate length scales. Shock wave has a thickness of a few mean free paths or the order of $10^{-7}m$, the boundary layer in shock tube is around $10^{-3}m$ thick, and the size of the macroscopic body is about $10^{-3}m - 10m$. Numerically a uniformly fine mesh covering every length scale is impractical. To be able to efficiently simulate these phenomena together, a mesh should ensure that the short-length phenomena are not very under-resolved and the long-length phenomena are not over-resolved, or the mesh should be allowed to adapt to these length scales. Unstructured data structure is therefore required to keep the changing topology due to the adaptation. In addition, for complicated geometry, unstructured mesh generation is far more automatable than the tasks associated with structured mesh generation.

An unstructured adaptive solver for numerical simulation consists of three primary parts, pre- and post-processing, flow solver, and grid adaptation procedure. In interaction of shock wave with bodies, the flow patterns, such as shock wave, slipstream, and vortex, are changing with time. An adaptive mesh has to follow the time-dependent flow patterns, thus the flow solver and grid adaptation procedure are closely coupled. This imposes great difficulties in implementing the unstructured adaptive solver with high efficiency on supercomputers.

Numerical simulation is inevitably linked to computing issues. Vector and parallel supercomputers provide much of the computing power that has hindered the computation of complex unsteady flows. The vectorization of a flow solver on un-

structured grids without mesh adaptation is not very challenging. It is accomplished by either sorting or coloring the elements into groups, so that within each group, no element accesses the same node. In general, sorting and coloring procedures themselves can not be efficiently vectorized. For an unstructured grid without mesh adaptation, sorting and coloring are just conducted once before the computation of the flow solver. In this case, the issue of their efficiency is trivial, and one may even design a time-consuming sorting method to optimize any unstructured data structure for an object machine. Unfortunately it is totally different for a grid with local mesh adaptation, because the local adaptation changes the topology of the initial grid. The data structure after adaptation may completely change the sequence which is provided by sorting and coloring procedures. Consequently, the sorting or coloring has to be conducted after every mesh adaptation, not surprisingly a very costly procedure. On the other hand, local mesh adaptation is associated with extremely complicated logic. The logic often contains data dependency which inhibits vectorization, so that the overheads of mesh adaptation may become relatively very large when the flow solver is computed by vector and parallel processing. On the Cray C90, a non-vectorized mesh adaptation procedure may require the computer time 10 times as the flow solver on single processor mode, and around 50 times using eight processors [1]. Low efficiency of mesh adaptation is somewhat tolerable in computation of steady flows, because few adaptations are sufficient. However, mesh adaptation has to be frequently performed in unsteady computation, so that one might even question that whether the efficiency is really gained by such expensive mesh adaptation compared with a uniform grid.

Therefore to be able to efficiently simulate shock wave interacting with bodies, it is required to underlie the flow solver and mesh adaptation by an essentially vectorizable data structure under which both the flow solver and mesh adaptation are able to be performed by vector processing. A novel essentially vectorizable data structure for any unstructured quadrilateral and hexahedral grids is proposed in the present thesis. The data structure is a bi-directional reference between cell and face with a strict ordering method. The ordering is kept during mesh adaptation, therefore no sorting or coloring is necessary. Furthermore, a flow solver using the finite volume method under the data structure can perform local and global operations by vector processing; locally obtain necessary connectivity information without any

logic, and globally scan all control volumes without nested loops. The data structure with these properties considerably decreases the overheads of both handling unstructured grids and adapting grids. Thus it is highly efficient for simulating unsteady and steady compressible flows with shock waves.

To construct an efficient unstructured solver, not only the data structure and grid adaptation should be efficient, but also the scheme should integrate the conservation laws economically. From the viewpoint of space discretization, the central schemes are more efficient and convenient. However all the second-order central schemes of the Lax-Wendroff family [2], [3] often generate oscillations around discontinuities. In order to remove the undesirable oscillations and to satisfy the entropy condition, artificial viscosity is included in these schemes. During the past decades many efforts have been made to attenuate these oscillations [4], [5], [6].

The other alternative lies in upwind discretizations. The upwind schemes utilize concepts of characteristic waves to determine the direction of spatial differencing. On an unstructured grid using the finite volume method, the flow variables are assumed to be constant or linearly distributed in control volumes. It leads to interactions at the interfaces between the control volumes. At each interface, the states on either side can be interpolated constantly or linearly from the variables at two neighboring control volumes. This defines a one-dimensional Riemann problem at each interface, which can be approximately solved by the Roe method [7] or the Osher method [8]. Although many successful codes have been developed utilizing this strategy, the true physics of multi-dimensional system are not modeled correctly, as waves are only allowed to propagate in directions of interfaces. This causes an inconsistency in the decomposition when the dominant characteristic in the flow field lies oblique to the grid. A classic example of this is that a simple slipstream oriented oblique to the interface. Because the Riemann solver is solved in the direction normal to the interface, the oblique slipstream will be decomposed as a shear aligned along the interface, plus non-physical acoustic waves. Over the past decade a substantial effort has been put into the development of a truly multi-dimensional method to eliminate the dependence of interface alignment. The early idea to improve the situation is the rotated Riemann solvers. In this approach, fluxes are still obtained by solving the Riemann problems at the interfaces, but in directions determined by

the physical nature of local flow [9], [10]. The more sophisticated modeling further chooses a multi-dimensional wave-pattern of more than three waves to calculate the fluxes at the interfaces [11], [12].

As far as artificial viscosity is concerned, a fundamental feature of the oscillation-free upwind schemes is that they ingeniously introduce first-order dissipation[1] around discontinuities by nonlinear limiters, and this dissipation is naturally adjusted for a slow or a fast characteristic wave since the upwind schemes involve a matrix dissipation coefficient. Because of the introduction of the first-order dissipation, a hyperbolic equation changes to a parabolic or elliptic one. In both cases, the numerical process is dominated by the artificial viscous effects as well as wave convection. Consequently, numerical results are not very sensitive to any Riemann solver or wave direction. As is well known, an upwind scheme can be written as a central difference scheme plus dissipation terms. This suggests that a central scheme may be free of oscillations by directly adding first-order nonlinear dissipation that should treat waves with different speeds as equals, or nearly as equals. In the present flow solver, the Lax-Wendroff scheme is combined with conservative smoothing, which is the first-order artificial dissipation. The nonlinear coefficient of smoothing is controlled so that the scheme is total variation diminishing (TVD) at least for the linear scalar case. In solving system of the conservation laws, unlike most unstructured solver creating a discontinuity at interface, the present approach assumes the variables are continuous and of finite spatial gradients. The states at interface are then calculated from the variables and their gradients by locally solving the multi-dimensional Euler equations instead of the Riemann solvers. In smooth regions the approach is second order accurate in both space and time. Advantages of the approach are, besides being cheap in arithmetic operations, it resulting in a compact stencil on structured and unstructured grids, and being truly multi-dimensional without recourse to the Riemann solver.

Although numerical simulation may provide all physical quantities some of which are difficult to be measured experimentally, high-resolution experiments are indispensable to study physics of complex flows and to set a standard that numerical

[1]In the thesis the order of dissipation is mentioned according to the coefficient of the second-difference effective viscosity or dissipation term. Any linear monotone scheme approximates solution of a viscous modified equation with the first-order dissipation term [13].

solutions should match up to. Holographic interferometry is just a method to measure quantitatively and non-invasively compressible flows. In two-dimensional flows if the real gas effect is neglected, fringes on interferograms correspond to isopycnics, so that this technique is useful to obtain the whole of density distribution of a flow field. There is no difficulty in resolving fringe spacing in a region where densities vary continuously. However, it is not straightforward to evaluate the density jump at shock waves, because the fringes there are so densely packed and sometimes beyond the limit of photographic resolution. A possible way of evaluating the density jump at the shock wave is to adopt the result of reliable numerical simulations. In this thesis, the density distribution is successfully estimated by simply counting fringes on interferograms and combining with the numerical result that is obtained by an adaptive solver. The technique is applied to quantitatively analyze shock wave focusing in a circular reflector. It is found that the highest density appears after the collision of triple points, not at the point as one usually expected.

The thesis is organized as follows:

- Chapter 1 is a brief introduction;
- Chapter 2 analyzes the Lax-Wendroff scheme with conservative smoothing on structured grids. The approach is first shown to be TVD for the scalar wave equation, then extended to 1-D and 2-D Euler equations. Numerical tests and comparison with some other TVD schemes are also given. The approach is coupled with the finite volume method and extensively tested in Chapter 4;
- Chapter 3 is devoted to the essentially vectorizable data structure for unstructured quadrilateral and hexahedral grids. The ordering method of the cell faces is described and proven here. The practical efficiency of the data structure for the quadrilateral grid is shown in Chapter 4;
- Chapter 4 presents a complete vectorizable adaptive solver on quadrilateral grids. This general-purpose solver is able to simulate unsteady/steady, inviscid/viscid compressible flows. An essentially vectorizable data structure (Chapter 3) underlies the flow solver and mesh adaptation. High efficiency of the solver on the Cray C90 is quantitatively demonstrated. A variety of nu-

merical simulations of shock interacting with different bodies shows that the scheme is reliable and problem-independent;

- Chapter 5 reports a study of shock wave focusing in a circular reflector by experimental and numerical methods. The image data processing partially supported by the numerical simulation is successfully applied to the shock tube experiment, in which very delicate shock wave interactions are well resolved. It is found that apparent shock wave focusing is created by the collision of triple points, however, the peak density appears at the time instant after the triple points merge at the centerline.

References

[1] Domel ND (1996) Research in parallel algorithms and software for computational aerosciences, NAS-96-004, NASA Ames Research Center.

[2] Lax PD, Wendroff B (1960) Systems of conservation laws, *Comm. Pure and Appl. Math.* 13 : 217-237.

[3] MacCormach RW (1969) The effect of viscosity in hypervelocity impact cratering, AIAA Paper 69-354.

[4] von Neumann J, Richtmyer RD (1950) A method for the numerical calculations of hydrodynamic shocks, *J. of Appl. Phys.* 21 : 232-237.

[5] Swanson RC, Turkel E (1987) Artificial dissipation and central diference schemes for the Euler and Navier-Stokes equations, AIAA paper 87-1107.

[6] Ni RH (1982) A multiple-grid schemes for solving the Euler equations, AIAA J. 20 : 1565-1571.

[7] Roe PL (1981) Approximate Riemann solvers, parameter vectors, and difference schemes, *J. of Comp. Phys.* 43 : 357-372.

[8] Osher S (1984) Riemann solvers, the entropy condition and difference approximations, SIAM *J. on Num. Analysis*, 21 : 217-235.

[9] Davis SF (1984) A rotationally biased upwind difference scheme for the Euler equations, *J. of Comp. Phys.* 56 : 65-92.

[10] Levy DW, Powell KG and van Leer B (1989) An implementation of a grid-independent upwind scheme for the Euler equations, AIAA paper 89-1931.

[11] Parpia IH, Michalek DJ (1993) Grid-indenpent upwind scheme for multidimensional flow, AIAA J. 31 : 646-651.

[12] Rumsey C, van Leer B, Roe PL (1993) A multidimensional flux function with applications to the Euler and Navier-Stokes equations, *J. of Comp. Phys.* 105 : 306-323.

[13] Harten A. (1983) High resolution schemes for hyperbolic conservation laws, *J. of Comput. Phys.* 49 : 357-393.

Chapter 2

On the Lax-Wendroff scheme with conservative smoothing

2.1 Introduction

Consider the initial value problem for the scalar wave equation,

$$u_t + f_x = 0 \tag{2.1}$$

or, letting $c = \partial f / \partial u$,

$$u_t + c u_x = 0 \tag{2.2}$$

with discrete initial values $u_{-\infty}, \cdots, u_i, \cdots, u_{+\infty}$ at nodes $x = i\Delta x$. A solution of the scalar equation has the following monotonicity property as a function of t [1]:

1. No new local extrema in x may be created;
2. The value of a local minimum is nondecreasing, the value of a local maximum in nonincreasing. (2.3)

If a scheme maintains the monotonicity property, then it is said to be *monotonicity preserving*. A scheme is said to be *conservative* if it may be written as

$$u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}), \tag{2.4}$$

where $\lambda = \Delta t / \Delta x$ is the ratio of time step Δt to grid size Δx. $\hat{f}_{i+1/2}$ is often referred to as *numerical flux*. The numerical flux function $\hat{f}$ is required to be consistent with the conservation law in the following sense

$$\hat{f}(u_i, \cdots, u_i) = f(u_i). \tag{2.5}$$

Assume that Δt and Δx tend to zero, the solution of a stable and conservative scheme converges to a weak solution of (2.1) with suitable initial conditions.

Based on physical arguments, namely the second principle of thermodynamics, only compression shocks are possible and expansion shocks, which correspond to a negative entropy variation, cannot occur in real flows. Since the inviscid wave model (2.1) does not have any dissipative mechanism, an additional condition is required to reject the non-physical shocks. The condition is known as *entropy condition*. A practical implementation of the entropy condition is adding a dissipation mechanism via *artificial viscosity* or *dissipation*.

In one word, in solving the conservation laws a numerical scheme should be conservative and monotonicity preserving, in addition to stable and consistent. An exact solution of Navier-Stokes equations shows that the thickness of a shock wave changes linearly with the physical viscosity coefficient (see for example [2] p.266). If we try to numerically imitate the physical dissipation by an artificial one and try to get a shock wave in a few cells, the artificial viscosity coefficient should also be a linear function of the cell size, or the coefficient should be first order. If a scheme is with either explicit or built-in second order artificial dissipation, mathematically it cannot sustain a shock wave in a few cells. Appropriate first-order artificial dissipation sufficiently prevents the appearance of spurious oscillations, and the minimum dissipation may be designed to prevent smearing discontinuities.

2.1.1 Total variation diminishing (TVD) scheme

Following the monotonicity property (2.3), Harten [1] was able to show that the total variation, denoted by $\mathrm{TV}(u(t))$, is nonincreasing in t, i.e.,

$$\mathrm{TV}(u(t_2)) \leq \mathrm{TV}(u(t_1)) \qquad \text{for all } t_2 \geq t_1. \tag{2.6}$$

Then, he defined a scheme is TVD, if it follows

$$\mathrm{TV}(u(t^{n+1})) \leq \mathrm{TV}(u(t^n)).$$

Any 3-node consistent schemes can be written in the form

$$u_i^{n+1} = u_i + C_{i+1/2}\Delta_{i+1/2}u - C_{i-1/2}\Delta_{i-1/2}u, \tag{2.7}$$

where

$$\Delta_{i+1/2}u = u_{i+1} - u_i. \tag{2.8}$$

Harten further proved that scheme (2.7) is TVD if

$$\begin{array}{ll} C_{i+1/2} \geq 0, & C_{i-1/2} \geq 0 \\ C_{i+1/2} + C_{i-1/2} \leq 1 & \end{array} \tag{2.9}$$

This criterion is also necessary for arbitrarily initial profiles. It has been proven by Godunov [3] that any linear monotonicity-preserving scheme and therefore any linear TVD scheme is only first order accurate. It is noticed that the conditions (2.9) is not necessary for some initial profiles; thus one may combine a second-order scheme for the regions where the scheme is monotonicity preserving with a first-order monotone scheme for other regions where a second-order scheme violates monotonicity.

2.1.2 The Lax-Wendroff scheme

Let $u(x,t)$ be an exact smooth solution of equation (2.1). It will then satisfy a difference equation approximately; the deviation is called the truncation error. Lax and Wendroff [4] proposed a method to minimize the truncation error, which leads to the well-known Lax-Wendroff scheme.

Expand $u(x, t + \Delta t)$ into a Taylor series up to terms of second order:

$$u(x, t + \Delta t) = u(x,t) + \Delta t u_t + 1/2(\Delta t)^2 u_{tt} + O(\Delta^3). \tag{2.10}$$

With the help of the differential equation (2.1) which u is supposed to satisfy, we can express the time derivatives of u as space derivatives,

$$\begin{array}{l} u_t = -f_x; \\ u_{tt} = -f_{xt} = -f_{tx} = -(cu_t)_x = (c^2 u_x)_x; \end{array} \tag{2.11}$$

Note that all t derivatives are exact x derivatives and therefore can be approximated by exact x differences. Substitute namely (2.11) into (2.10), we get

$$u(x, t + \Delta t) = u(x,t) - (\Delta t f - 1/2(\Delta t)^2 c^2 u_x)_x. \tag{2.12}$$

Comparing (2.12) with (2.4), one may obtain the numerical flux

$$\hat{f}_{i+1/2} = f(u_{i+1/2}) - 1/2\lambda c^2 (u_{i+1} - u_i) \tag{2.13}$$

plus high order terms, where $u_{i+1/2} = (u_{i+1} + u_i)/2$. This is just the Lax-Wendroff scheme. For a linear wave equation, $c =$ constant, the scheme becomes

$$\hat{f}_{i+1/2} = c(u_{i+1/2} - 1/2\sigma\Delta_{i+1/2}u), \tag{2.14}$$

where $\sigma = c\Delta t/\Delta x$. The scheme is stable for $|\sigma| \leq 1$.

The second term on right hand of (2.12) can be rewritten, using (2.11), $\Delta t(f + 1/2\Delta t f_t)_x = \Delta t f(t + 1/2\Delta t)_x$, which leads to a predictor-corrector Lax-Wendroff scheme, known as the Richtmyer scheme,

$$\begin{aligned} u^*_{i+1/2} &= u_{i+1/2} - 1/2\lambda\Delta_{i+1/2}f, \\ \hat{f}_{i+1/2} &= f(u^*_{i+1/2}). \end{aligned} \tag{2.15}$$

For the linear wave equation, the Lax-Wendroff scheme is the unique 3-node finite difference scheme with second order accuracy. However, it is not TVD. To show this, inserting (2.14) to (2.4), one may rewrite the Lax-Wendroff scheme, in the form of (2.7),

$$u_i^{n+1} = u_i + (-\sigma/2 + \sigma^2/2)\Delta_{i+1/2}u - (\sigma/2 + \sigma^2/2)\Delta_{i-1/2}u. \tag{2.16}$$

Then

$$\begin{aligned} C_{i+1/2} &= -\sigma/2 + \sigma^2/2, \\ C_{i-1/2} &= \sigma/2 + \sigma^2/2. \end{aligned}$$

Both $C_{i+1/2}$ and $C_{i-1/2}$ are possibly negative for $-1 \leq \sigma \leq 1$, therefore the Lax-Wendroff scheme is not TVD. It generates oscillations behind a shock discontinuity. As far as artificial viscosity is concerned, the uniformly second-order Lax-Wendroff scheme provides artificial dissipation less than first order, consequently it cannot support a physical shock transition in a few grid cells.

To be able to construct a TVD Lax-Wendroff scheme, additional first-order dissipation is required. In Section 2.2.1, Roe's formulation of adding dissipation is reviewed, and the proposed method will be discussed in Section 2.2.2.

2.2 The scalar case

2.2.1 The TVD Lax-Wendroff scheme via limiter

The first TVD Lax-Wendroff scheme was derived by Davis [7], soon Roe [6] reformulated Davis' scheme and included a new class of TVD schemes. Yee [8] further

extended the idea to implicit methods. In this section, Roe's generalized formulation of the TVD Lax-Wendroff scheme is reviewed. The formulation is basically the Lax-Wendroff plus a conservative dissipation term designed in such a way that the final scheme is TVD. For c =constant, his scheme is written as

$$\begin{aligned} u_i^{n+1} = \; & u_i + (-\sigma/2 + \sigma^2/2)\Delta_{i+1/2}u - (\sigma/2 + \sigma^2/2)\Delta_{i-1/2}u \\ & +1/2|\sigma|(1-|\sigma|)(1-Q_{i+1/2})\Delta_{i+1/2}u \\ & -1/2|\sigma|(1-|\sigma|)(1-Q_{i-1/2})\Delta_{i-1/2}u. \end{aligned} \tag{2.17}$$

The last two terms represent an additional conservative dissipation, and the first terms are just the Lax-Wendroff scheme (2.16). The *limiter* $Q_{i+1/2}$ depends three consecutive difference $\Delta_{i-1/2}u$, $\Delta_{i+1/2}u$ and $\Delta_{i+3/2}u$ and is of the form

$$Q_{i+1/2} = Q(r^-_{i+1/2}, r^+_{i-1/2}),$$

where

$$r^-_{i+1/2} = \Delta_{i-1/2}u/\Delta_{i+1/2}u, \qquad r^+_{i+1/2} = \Delta_{i+3/2}u/\Delta_{i+1/2}u.$$

If one assumes both Q and Q/r are always positive, then a set of sufficient conditions for (2.17) to be TVD is

$$\begin{aligned} & Q < 2/(1-|\sigma|), \\ & Q/r < 2/|\sigma|. \end{aligned} \tag{2.18}$$

Some examples of limiter Q are

$$Q = \text{minmod}(1, r^-) + \text{minmod}(1, r^+) - 1, \tag{2.19}$$

$$Q = \text{minmod}(1, r^-, r^+), \tag{2.20}$$

where the "minmod" function is defined as

$$\text{minmod} = \begin{cases} \min(1, r^\pm), & r^\pm > 0, \\ 0, & r^\pm \le 0. \end{cases}$$

More examples of the limiter Q may be found in [10].

2.2.2 The TVD Lax-Wendroff scheme via nonlinear artificial viscosity

Consider a numerical flux for equation (2.2) with c = constant

$$\hat{f}_{i+1/2} = c[u_{i+1/2} - \sigma(1/2 - \mu)\Delta_{i+1/2}u] - \mu\Delta_{i+1/2}u/\lambda, \tag{2.21}$$

where μ is an artificial viscosity parameter.

Lemma 1. The scheme (2.4) with flux (2.21) is monotone under the CFL condition $|\sigma| \leq 1$ if and only if $|\sigma|/[2(1+|\sigma|)] \leq \mu \leq 1/2$.

Proof: Combining two equations in (2.21), one may rewrite it as

$$u_i^{n+1} = u_i + A\Delta_{i+1/2}u - B\Delta_{i-1/2}u \tag{2.22}$$

where

$$\begin{array}{c} A = -\sigma/2 + \sigma^2/2 + \mu(1-\sigma^2), \\ B = \sigma/2 + \sigma^2/2 + \mu(1-\sigma^2). \end{array} \tag{2.23}$$

A monotone scheme requires (see, (2.9)),

$$A \geq 0; \quad B \geq 0; \quad A + B \leq 1.$$

$A \geq 0$ leads to

$$\begin{array}{c} A = -\sigma/2(1-\sigma) + \mu(1-\sigma^2) \geq 0 \\ (1-\sigma)(\mu(1+\sigma) - \sigma/2) \geq 0 \\ \mu \geq \sigma/[2(1+\sigma)]; \end{array}$$

$B \geq 0$ leads to

$$\begin{array}{c} B = \sigma/2(1+\sigma) + \mu(1-\sigma^2) \geq 0 \\ (1+\sigma)(\mu(1-\sigma) + \sigma/2) \geq 0 \\ \mu \geq -\sigma/[2(1-\sigma)]. \end{array}$$

For $-1 \leq \sigma \leq 1$, both $A \geq 0$ and $B \geq 0$ are thus equivelent to

$$\mu \geq |\sigma|/[2(1+|\sigma|)].$$

$A + B \leq 1$ leads to

$$\begin{array}{c} A + B = \sigma^2 + 2\mu(1-\sigma^2) \leq 1 \\ \mu \leq 1/2. \end{array}$$

This completes the proof of *Lemma 1.*

Remark The truncation error of scheme (2.21) is

$$u_t + cu_x = \mu\frac{\Delta x^2}{\Delta t}(1-\sigma^2)u_{xx} + \cdots$$

Clearly, parameter μ is a sort of artificial viscosity coefficient. A few central and upwind schemes can be written in this form, for instance,

μ	schemes
0	Lax-Wendroff scheme
$\|\sigma\|/[2(1+\|\sigma\|)]$	First-order upwind scheme
1/2	Lax-Friedrichs scheme

These values μ of the Lax-Friedrichs scheme and the first-order upwind scheme are just the upper limit and lower limit of $|\sigma|/[2(1+|\sigma|)] \leq \mu \leq 1/2$. Therefore in monotone schemes written in (2.21), the Lax-Friedrichs scheme is the most diffusive, while the upwind scheme is the least one. It is very interesting that the central scheme (2.21) may trigger artificial dissipation as less as the upwinding scheme do by choosing the lower limit value. Even choosing $\mu = 1/4$ scheme (2.21) is half as diffusive as the Lax-Friedrichs scheme. For convenience, we hereafter use the notation,

$$\mu_{up} \equiv |\sigma|/[2(1+|\sigma|)], \tag{2.24}$$

where the subscript "*up*" denotes the viscosity coefficient of the upwind scheme.

Note that $\mu = 0$ corresponds to the second-order Lax-Wendroff scheme. This property provides a simple way, by switching μ to be 0 in smooth regions, to extend this sort of monotone scheme to be second order, which will be discussed later. ¶

Remark Actually, all 3-node finite difference schemes may be expressed as the central scheme (2.21). A 3-node scheme is

$$u_i^{n+1} = C_{i-1}u_{i-1} + C_i u_i + C_{i+1}u_{i+1}. \tag{2.25}$$

To be able to consistent with equation (2.2) up to the first-order, coefficients C_{i-1},C_i and C_{i+1} should satisfy

$$C_{i-1} + C_i + C_{i+1} = 1, \tag{2.26}$$

$$C_{i+1} - C_{i-1} = -\sigma. \tag{2.27}$$

Substitute (2.26) into (2.25), one gets

$$u_i^{n+1} = u_i - C_{i-1}\Delta_{i-1/2}u + C_{i+1}\Delta_{i+1/2}u. \tag{2.28}$$

Compared with (2.22), because $A - B = C_{i+1} - C_{i-1} = -\sigma$, it is sufficient to let

$$A = C_{i+1},$$
$$-\sigma/2 + \sigma^2/2 + \mu(1-\sigma^2) = C_{i+1},$$
$$\mu = (C_{i+1} + \sigma/2 - \sigma^2/2)/(1-\sigma^2). \tag{2.29}$$

Therefore any 3-node scheme (2.25) is equivalent to the central scheme (2.21) with (2.29). It is seen that Roe's formulation (2.17) corresponds to

$$\mu = \mu_{up}(1-Q). \tag{2.30}$$

¶

Remark For nonlinear equation

$$u_t + f(u)_x = 0,$$

the flux in scheme (2.21) is able to be calculated by a *predictor step*:

$$\begin{aligned} u^*_{i+1/2} &= u_{i+1/2} - \lambda(1/2-\mu)\Delta_{i+1/2} f, \\ \hat{f}_{i+1/2} &= f(u^*_{i+1/2}) - \mu\Delta_{i+1/2} u/\lambda. \end{aligned} \tag{2.31}$$

For $\mu = 0$, (2.31) becomes the Richtmyer scheme (2.15). This method eventually avoids the estimation of the Jacobian matrix that is associated with a system of conservation laws. ¶

Remark A geometrical interpretation of scheme is shown in Fig. 2.1. It is actually an advanced model of the smoothing method. The smoothing method first smoothens a sharp shock to be a continuous profile, as a pre-smoothing procedure, then conducts the Lax-Wendroff scheme on the continuous profile shown in Fig. 2.1b. This method requires neither the Riemann solvers nor the estimation of the Jacobian matrix, and has been extensive applied in solving 1-D and 2-D Euler equations [9]. Scheme (2.21) uses another profile, as shown in Fig. 2.1c, and conducts the two-step Lax-Wendroff on it. This method has its built-in smoothing. The coefficient μ is just the distance from the initial profile to the profile used in calculation. In (2.31), the term $\lambda(1/2-\mu)\Delta_{i+1/2} f$ can be rewritten as

$$1/2\Delta t(1-2\mu)\Delta_{i+1/2} f/\Delta x,$$

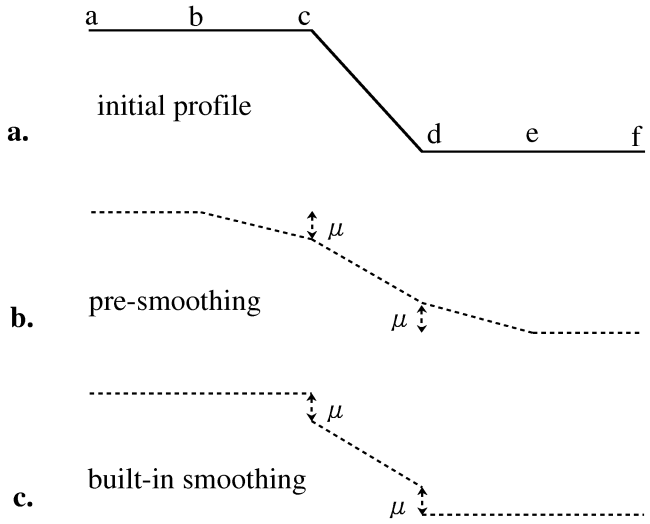


Figure 2.1. Geometrical interpretation of scheme (2.21). The coefficient μ is a distance from the initial profile to the one after smoothing. The distance is normalized by the local jump between two neighboring nodes, say c and d.

where $1/2\Delta t$ propagates the solution by half time step, and additional coefficient $(1-2\mu)$ is due to the change of gradient after smoothing (see Fig. 2.1c). Therefore the scheme (2.21) is referred to as a *smoothing* scheme. If μ is a constant, then it is a linear smoothing scheme; if μ is not constant, a nonlinear smoothing scheme. ¶

Lemma 2. The Lax-Wendroff scheme is monotonicity preserving at nodes u_i for $l < i < k$ under the CFL condition $|\sigma| \leq 1$ if for all nodes i, $l-2 < i < k+2$ hold

$$0 < \frac{\Delta_{i+1/2}u}{\Delta_{i-1/2}u} \leq r_0; \quad 0 < \frac{\Delta_{i-1/2}u}{\Delta_{i+1/2}u} \leq r_0, \tag{2.32}$$

where $r_0 = 2 + 2/|\sigma|$.

Proof: One should prove

$$R = \frac{\Delta^{n+1}_{i+1/2}u}{\Delta^{n}_{i+1/2}u} \geq 0. \tag{2.33}$$

The Lax-Wendroff scheme gives

$$u_i^{n+1} = u_i + (-\sigma/2 + \sigma^2/2)\Delta_{i+1/2}u - (\sigma/2 + \sigma^2/2)\Delta_{i-1/2}u,$$

then

$$R = (1-\sigma^2) + (-\sigma/2 + \sigma^2/2)\frac{\Delta_{i+3/2}u}{\Delta_{i+1/2}} + (\sigma/2 + \sigma^2/2)\frac{\Delta_{i-1/2}u}{\Delta_{i+1/2}}.$$

For $\sigma > 0$, given the ratio $\Delta_{i-1/2}u/\Delta_{i+1/2} > 0$, one has

$$\begin{aligned} R &\geq 1-\sigma^2 + (-\sigma/2 + \sigma^2/2)\Delta_{i+3/2}u/\Delta_{i+1/2} \\ &\geq 1-\sigma^2 + (-\sigma/2 + \sigma^2/2)r_0 \\ &= 0. \end{aligned} \tag{2.34}$$

Similarly for $\sigma < 0$, given the ratio $\Delta_{i+3/2}u/\Delta_{i+1/2} > 0$, one has

$$\begin{aligned} R &\geq 1-\sigma^2 + (-|\sigma|/2 + \sigma^2/2)\Delta_{i-1/2}u/\Delta_{i+1/2} \\ &\geq 1-\sigma^2 + (-|\sigma|/2 + \sigma^2/2)r_0 \\ &= 0. \end{aligned} \tag{2.35}$$

This completes the proof of *Lemma 2*.

Remark Condition (2.33) is weaker than condition

$$0 \leq -\frac{u_i^{n+1} - u_i}{\Delta_{i-1/2}^n u} \leq 1, \qquad (\sigma \geq 0). \tag{2.36}$$

(2.36) leads to $r_0^* = (2 - |\sigma| - \sigma^2)/(|\sigma| - \sigma^2)$, but not $r_0 = 2/|\sigma| + 2$. It is easy to show $r_0^* \leq r_0$ for $|\sigma| \leq 1$. A limiter satisfying condition (2.33) is therefore less diffusive than that satisfying (2.36).

Corollary 1 The Lax-Wendroff scheme is monotonicity preserving at nodes u_i for $l < i < k$ under the CFL condition $|\sigma| \leq 1$ if for all nodes $l-2 < i < k+2$ hold

$$\phi_i \equiv \frac{|u_{i+1} + u_{i-1} - 2u_i|}{\varepsilon_0 + |u_{i+1} - u_{i-1}|} \leq \phi_{\max}, \tag{2.37}$$

where ε_0 is an infinitely small positive value used when $u_{i+1} - u_{i-1} = 0$, and $\phi_{\max} = (r_0 - 1)/(r_0 + 1)$.

Proof: Note that $r_0 \geq 4$ for $|\sigma| \leq 1$, then one has $(r_0 - 1)/(r_0 + 1) < 1$. Using the notation (2.8), one may rewrite relation (2.37) as

$$\phi_i = \frac{|\Delta_{i+1/2}u - \Delta_{i-1/2}u|}{\varepsilon_0 + |\Delta_{i+1/2}u + \Delta_{i-1/2}u|} \leq \frac{r_0 - 1}{r_0 + 1} < 1 \tag{2.38}$$

For $\Delta_{i-1/2}u\Delta_{i+1/2}u \leq 0$, it is easily seen that $\phi_i \geq 1$, thus violates inequality (2.38).

Let $r = \max(\Delta_{i+1/2}u/\Delta_{i-1/2}u, \Delta_{i-1/2}u/\Delta_{i+1/2}u) \geq 1$, *Lemma 2* requests that one should prove $r \leq r_0$, which is equivalent to, for $r \geq 1$,

$$\frac{r-1}{r+1} \leq \frac{r_0-1}{r_0+1},$$

or

$$\frac{|r-1|}{|r+1|} \leq \frac{r_0-1}{r_0+1}. \tag{2.39}$$

Since r is equal to either $\Delta_{i+1/2}u/\Delta_{i-1/2}u$ or $\Delta_{i-1/2}u/\Delta_{i+1/2}u$, in both cases inequality (2.39) leads to

$$\frac{|\Delta_{i+1/2}u - \Delta_{i-1/2}u|}{|\Delta_{i+1/2}u + \Delta_{i-1/2}u|} \leq \frac{r_0-1}{r_0+1}.$$

If $|\Delta_{i+1/2}u + \Delta_{i-1/2}u| = 0$, there are two possibilities. One is $\Delta_{i+1/2}u = -\Delta_{i-1/2}u = 0$, then $\phi_i = 0$ because of the small positive ε_0. Another is $\Delta_{i+1/2}u = -\Delta_{i-1/2}u \neq 0$, *i.e.*, they change sign, then $\phi_i = |\Delta_{i+1/2}u - \Delta_{i-1/2}u|/\varepsilon_0$. Given a sufficiently small ε_0, one has $\phi_i >> 1$, thus violate inequality (2.38). This completes the proof of *Corollary 1*.

Remark The ratio ϕ in (2.37) plays an essential role in limiting procedure. Fig. 2.2 shows five representative stages of ϕ. For any linear function, ϕ is always equal to 0; for a monotone function, ϕ is not larger than 1; the critical case is a monotone function with sharp changes, then $\phi = 1$; if there is an extremum in a function, ε_T at the extremum is greater than 1; the worst case is an oscillation with wave length $2\Delta x$, the ratio ϕ then approaches infinity. Therefore, the ratio ϕ is a quantitative *indicator* of a smooth function or an oscillating function, and the smooth regions or the oscillating regions in one function. The critical value ϕ=1 corresponds to shock discontinuities. As it is generally known that the Lax-Wendroff scheme generates oscillations near a discontinuity, one can conclude that ϕ should be close to 1. This agrees with that the upper limit, $(r_0-1)/(r_0+1)$, has the minimum value 0.6 for $|\sigma| = 1$, and approaches the maximum value 1.0 for very low CFL number $|\sigma|$. It can be easily proven by an algebraic method that a

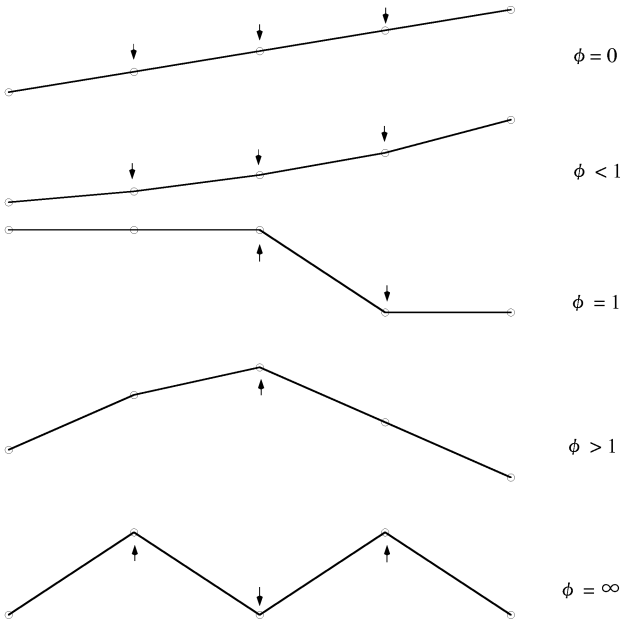


Figure 2.2. Five representative stages of ϕ

sufficient and necessary condition of a monotone sequence is that $\phi < 1$. Note that ϕ=1 or $\phi = \infty$ is independent of the jump of a discontinuity or an oscillation. It is an asset to be able to compute discontinuities with different strengths, and to remove various oscillations using the same indicator.

Compared with the conventional smoothness detector, such as

$$r = \Delta_{i+1/2}u/\Delta_{i-1/2}u,$$

the indicator ϕ has two advantages. One is that ϕ is always positive, so that the possible zero denominator is reliably solved by introducing an infinitely small value ε_0, say, $\varepsilon_0 = \Delta x^2$. Another advantage is that indicator ϕ is linearly increased from 0 to ∞ when a solution changes from smooth regions to oscillations regions. It has no negative values as r do. This property will simplify logic that is built in a limiter. Actually ϕ is often used in adaptive mesh refinement as an error detector. Since a high-order scheme turns on its built-in artificial dissipation using a limiter, the scheme degrades to be first order in some regions. Using ϕ in the limiter for the scheme and using it as an error detector for an adaptation procedure, assures that

fine grids are distributed around regions where the scheme has first-order truncation errors, thus greatly enhances global accuracy and adaptation efficiency. The method is indeed applied in following chapters. ¶

Remark The ratio ϕ_i may be expressed as, in a continuous form,

$$\phi = |\frac{\frac{1}{2}u''\Delta x^2}{u'\Delta x}|.$$

On the other hand, the Taylor series of a continuous function is

$$u(x+\Delta x) = u(x) + u'\Delta x + \frac{1}{2}u''\Delta x^2 + \cdots$$

It is seen ϕ is just the ratio of the second-order term in the Taylor series to the first-order term. *Corollary 1* suggests that the Lax-Wendroff scheme may violate monotonicity for $\phi > 0.6$, that is, the second order term has the same order as the first order term in the Taylor series. A finite difference scheme always assumes that a higher order term should be less than a lower one. From this point of view, the Lax-Wendroff scheme violating monotonicity or generating oscillations is due to unsuitable truncation of the Taylor expansion. ¶

Let's switch the linear smoothing scheme to the Lax-Wendroff scheme, in a nonlinear way, by setting

$$\mu = \begin{cases} 0, & \phi_{i+1/2} \le \phi_0, \\ \mu_0, & \text{otherwise}, \end{cases} \tag{2.40}$$

where $\phi_{i+1/2} = \max(\phi_i, \phi_{i+1})$, and

$$\phi_0 \equiv \frac{r_0(1-2\mu_0)-1}{r_0(1-2\mu_0)+1}.$$

Lemma 3. The smoothing scheme with nonlinear coefficient (2.40) is monotonicity preserving under the CFL condition $|\sigma| \le 1$ if $\mu_{up} \le \mu_0 \le 1/2$.

Proof Without losing generality suppose left nodes of i, and node i belongs to a smooth region, $\phi \le \phi_0$; while right nodes a shock region, $\phi > \phi_0$. The flux through

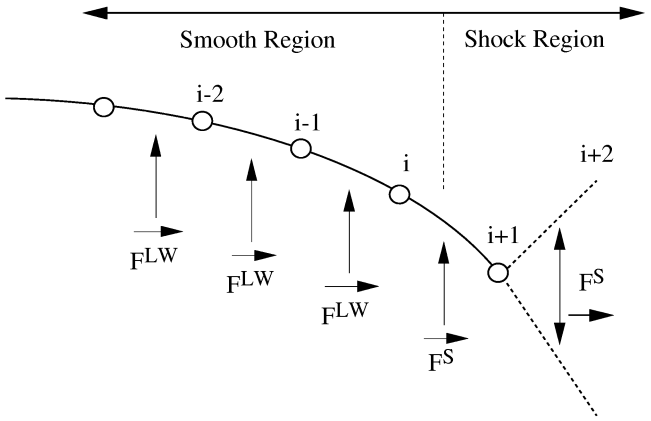


Figure 2.3. Sketch of the transition region between the smoothing scheme and the Lax-Wendroff scheme. F^S and F^{LW} denote the fluxes of the two schemes, respectively.

any interface is either the flux of the Lax-Wendroff scheme or the flux of the linear smoothing scheme, as shown in Fig. 2.3. According to *Lemma 1* and *Corollary 1* the Lax-Wendroff scheme in the smooth region and the linear smoothing scheme in the shock region are monotonicity preserving, therefore it is sufficient to prove that in the transition region of two schemes, the hybrid scheme (2.40) is still monotonicity preserving,*i.e.*, one should prove (a). between nodes i and $i-1$

$$R_1 = \frac{u_i^{n+1} - u_{i-1}^{n+1}}{\Delta_{i-1/2}u} \geq 0; \tag{2.41}$$

(b). between nodes i and $i+1$, if node $i+1$ is an extrema, $\Delta_{i+3/2}u/\Delta_{i+1/2}u < 0$,

$$R_2 = \frac{u_{i+1} - u_{i+1}^{n+1}}{\Delta_{i+1/2}u} \geq 0; \quad R_3 = \frac{u_{i+1} - u_i^{n+1}}{\Delta_{i+1/2}u} \geq 0; \tag{2.42}$$

(c). between nodes i and $i+1$, if node $i+1$ is not an extrema, $\Delta_{i+3/2}u/\Delta_{i+1/2}u \geq 0$

$$R_4 = \frac{u_{i+1}^{n+1} - u_i^{n+1}}{\Delta_{i+1/2}u} \geq 0. \tag{2.43}$$

Using the notation in (2.23), one has

$$\begin{array}{rcl} u_{i+1}^{n+1} & = & u_{i+1} + A\Delta_{i+3/2}u - B\Delta_{i+1/2}u, \\ u_i^{n+1} & = & u_i + \sigma u_{i-1/2} - \sigma^2/2\Delta_{i-1/2}u + Au_{i+1} - Bu_i, \\ u_{i-1}^{n+1} & = & u_{i-1} + (-\sigma/2 + \sigma^2/2)\Delta_{i-1/2} - (\sigma/2 + \sigma^2/2)\Delta_{i-3/2}. \end{array} \tag{2.44}$$

Since $\phi_{i-1}, \phi_i \le \phi_0 < 1$, then one has

$$\begin{aligned} r_0(1-2\mu_0) &\ge \Delta_{i-3/2}u/\Delta_{i-1/2}u > 0, \\ r_0(1-2\mu_0) &\ge \Delta_{i+1/2}u/\Delta_{i-1/2}u > 0, \\ r_0(1-2\mu_0) &\ge \Delta_{i-1/2}u/\Delta_{i+1/2}u > 0. \end{aligned} \tag{2.45}$$

We first prove (a) here:

$$\begin{aligned} R_1 &= [u_i + \sigma u_{i-1/2} - \sigma^2/2\Delta_{i-1/2}u + Au_{i+1} - Bu_i \\ &\quad -(u_{i-1} + (-\sigma/2 + \sigma^2/2)\Delta_{i-1/2}u - (\sigma/2 + \sigma^2/2)\Delta_{i-3/2}u)] \\ &\quad /\Delta_{i-1/2}u \\ &= 1 - \sigma^2 + \sigma/2 + \\ &\quad [(\sigma/2 - B)u_i + \sigma/2u_{i-1} + Au_{i+1} + (\sigma/2 + \sigma^2/2)\Delta_{i-3/2}u]/\Delta_{i-1/2}u \\ &= 1 - \sigma^2 + [A\Delta_{i+1/2}u + (\sigma/2 + \sigma^2/2)\Delta_{i-3/2}u]/\Delta_{i-1/2}u \end{aligned}$$

The condition $A - B = -\sigma$ has been used in the last step. Since $A \ge 0$, then

$$R_1 \ge 1 - \sigma^2 + (\sigma/2 + \sigma^2/2)\Delta_{i-3/2}u/\Delta_{i-1/2}u. \tag{2.46}$$

For $\sigma \ge 0$, then $R_1 \ge 0$;

for $\sigma < 0$,

$$\begin{aligned} R_1 &\ge 1 - \sigma^2 - 1/2|\sigma|(1 - |\sigma|)\Delta_{i-3/2}u/\Delta_{i-1/2}u \\ &\ge 1 - \sigma^2 - 1/2|\sigma|(1 - |\sigma|)r_0(1 - 2\mu_0) \\ &= 2\mu_0(1 - \sigma^2) \\ &\ge 0. \end{aligned}$$

Prove (b) here:

$$\begin{aligned} R_2 &= (-A\Delta_{i+3/2}u + B\Delta_{i+1/2}u)/\Delta_{i+1/2}u \\ &= B - A\Delta_{i+3/2}/\Delta_{i+1/2}u \\ &\ge 0; \end{aligned}$$

$$\begin{aligned} R_3 &= 1 - (\sigma u_{i-1/2} - \sigma^2/2\Delta_{i-1/2}u + Au_{i+1} - Bu_i)/\Delta_{i+1/2}u \\ &= 1 - A - [(A - B + \sigma/2)u_i + \sigma/2u_{i-1} - \sigma^2/2\Delta_{i-1/2}u]/\Delta_{i+1/2}u \\ &= 1 - A + (\sigma/2 + \sigma^2/2)\Delta_{i-1/2}u/\Delta_{i+1/2}u. \end{aligned}$$

For $\sigma \ge 0$, then $R_3 \ge 0$;

for $\sigma < 0$,

$$\begin{aligned} R_3 &= 1 + \sigma/2 - \sigma^2/2 - \mu_0(1 - \sigma^2) + (\sigma/2 + \sigma^2/2)\Delta_{i-1/2}u/\Delta_{i+1/2}u \\ &\ge 1 + \sigma/2 - \sigma^2/2 - \mu_0(1 - \sigma^2) + (\sigma/2 + \sigma^2/2)r_0(1 - 2\mu_0) \\ &= 1 + \sigma/2 - \sigma^2/2 - (1 - \mu_0)(1 - \sigma^2) \\ &\ge 1 + \sigma/2 - \sigma^2/2 - [1 - |\sigma|/(2(1 + |\sigma|))](1 - \sigma^2) \\ &= 1 - |\sigma|/2 - \sigma^2/2 - (1 - 3/2|\sigma| + \sigma^2/2) \\ &= |\sigma| \\ &\ge 0. \end{aligned}$$

Prove (c) here:

$$\begin{aligned} R_4 &= [u_{i+1} + A\Delta_{i+3/2}u - B\Delta_{i+1/2}u \\ &\quad -(u_i + \sigma u_{i-1/2} - \sigma^2/2\Delta_{i-1/2}u + Au_{i+1} - Bu_i)]/\Delta_{i+1/2}u \\ &\geq 1 - B - A+ \\ &\quad [(B - A - \sigma/2)u_i - \sigma/2u_{i-1} + \sigma^2/2\Delta_{i-1/2}u]/\Delta_{i+1/2}u \\ &= (1-\sigma^2)(1-2\mu_0) + (\sigma/2 + \sigma^2/2)\Delta_{i-1/2}u/\Delta_{i+1/2}u. \end{aligned}$$

For $\sigma \geq 0$, then $R_4 \geq 0$;
for $\sigma < 0$,

$$\begin{aligned} R_4 &= (1-\sigma^2)(1-2\mu_0) - 1/2|\sigma|(1-|\sigma|)\Delta_{i-1/2}u/\Delta_{i+1/2}u \\ &\geq (1-\sigma^2)(1-2\mu_0) - 1/2|\sigma|(1-|\sigma|)r_0(1-2\mu_0) \\ &= 0. \end{aligned}$$

This completes the proof of *Lemma 3*.

Remark For a hybrid method which combines the Lax-Wendroff scheme and the linear smoothing scheme through the form of (2.40), *Lemma 3* states that one should switch the Lax-Wendroff scheme to the smoothing scheme when $\phi \geq \phi_0$. Not all monotone smoothing schemes expressed as (2.21) can be combined in this way. Coefficient (2.40) will never switch to $\mu = 0$, when $\phi_0 \leq 0$, *i.e.*,

$$r_0(1-2\mu_0) - 1 \leq 0,$$

$$\mu_0 \geq (2+|\sigma|)/(4+4|\sigma|) \geq 3/8. \tag{2.47}$$

For instance, the Lax-Friedrichs scheme corresponds to $\mu_0 = 1/2$, so that it can not be switched to the Lax-Wendroff scheme while preserving monotonicity.

Since for $\mu = 0$ the nonlinear smoothing scheme is exactly the Lax-Wendroff scheme, it is second order accurate in the regions where $\phi_{i+1/2} \leq \phi_0$. ¶

Remark For any $0 < \mu_0 < 1/2$, ϕ_0 is always less than $\phi_{\max}$. It suggests that in the transition region $\phi_0 \leq \phi < \phi_{\max}$, the Lax-Wendroff scheme is still monotonicity preserving and thus artificial viscosity is not required. The artificial viscosity has to be turn on there just because the jump in coefficient (2.40). Note that for $\mu_0 = \mu_{up}$, ϕ_0 corresponds to $r = 2/|\sigma|$. Compared with (2.30), the nonlinear coefficient can be rewritten as

$$\mu = \mu_{up}(1-Q), \tag{2.48}$$

where,

$$Q = \begin{cases} 1, & \phi_{i+1/2} \le \phi_0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.49}$$

One may design a continuous coefficient μ, by using (2.18) for $\mu_0 = \mu_{up}$. Consider a limiter

$$Q = \max(0, \min(1, \frac{2r}{|\sigma|})), \tag{2.50}$$

which leads to

$$\mu = \mu_{up}[1 - \max(0, \min(1, \frac{2}{|\sigma|} r_{i+1/2}))], \tag{2.51}$$

where

$$r_{i+1/2} = \frac{1 - \phi_{i+1/2}}{1 + \phi_{i+1/2}}. \tag{2.52}$$

¶

Consider a more general coefficient

$$\mu = \begin{cases} 0, & \phi_{i+1/2} \le \phi_c, \\ \mu_t & \text{otherwise} \\ \mu_0, & \phi_{i+1/2} \ge 1 \end{cases} \tag{2.53}$$

or simply,

$$\mu = \min[\mu_0, \max(0, \mu_t)] \tag{2.54}$$

where μ_t denotes artificial viscosity coefficient in transition regions, and ϕ_c represents some critical value.

Lemma 4. The smoothing scheme with nonlinear coefficient (2.54) is monotonicity preserving under the CFL condition $|\sigma| \le 1$ if

$$\mu_0 - (1 - 2\mu_0) r_{i+1/2} \le \mu_t \le \mu_0$$

with $\mu_{up} \le \mu_0 \le 1/2$.

Proof It is sufficient to prove that

$$R = \frac{u_{i+1}^{n+1} - u_i^{n+1}}{\Delta_{i+1/2} u} \ge 0. \tag{2.55}$$

Since

$$u_{i+1}^{n+1} = u_{i+1} - \lambda(\hat{f}_{i+3/2} - \hat{f}_{i+1/2})$$

and

$$u_i^{n+1} = u_i - \lambda(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}),$$

one has

$$R = 1 + \lambda(-\hat{f}_{i+3/2} + 2\hat{f}_{i+1/2} - \hat{f}_{i-1/2})/\Delta_{i+1/2}u.$$

Substituting (2.21) with $\mu_{i+3/2}$, $\mu_{i+1/2}$ and $\mu_{i-1/2}$ for $\hat{f}_{i+3/2}$, $\hat{f}_{i+1/2}$ and $\hat{f}_{i-1/2}$ respectively, gives

$$\begin{aligned} R &= 1 + \{-\sigma u_{i+3/2} + [\sigma^2(1/2 - \mu_{i+3/2}) + \mu_{i+3/2}]\Delta_{i+3/2}u\}/\Delta_{i+1/2}u \\ &\quad + \{2\sigma u_{i+1/2} - 2[\sigma^2(1/2 - \mu_{i+1/2}) + \mu_{i+1/2}]\Delta_{i+1/2}u\}/\Delta_{i+1/2}u \\ &\quad + \{-\sigma u_{i-1/2} + [\sigma^2(1/2 - \mu_{i-1/2}) + \mu_{i-1/2}]\Delta_{i-1/2}u\}/\Delta_{i+1/2}u, \end{aligned}$$

$$\begin{aligned} R &= 1 + [\sigma^2/2 - \sigma/2 + \mu_{i+3/2}(1 - \sigma^2)]r_{i+1/2}^+ - 2[\sigma^2/2 + \mu_{i+1/2}(1 - \sigma^2)] \\ &\quad + [\sigma^2/2 + \sigma/2 + \mu_{i-1/2}(1 - \sigma^2)]r_{i+1/2}^-, \end{aligned}$$

thus

$$\begin{aligned} R = (1 - \sigma^2)(1 - 2\mu_{i+1/2}) &+ [\sigma^2/2 - \sigma/2 + \mu_{i+3/2}(1 - \sigma^2)]r_{i+1/2}^+ \\ &+ [\sigma^2/2 + \sigma/2 + \mu_{i-1/2}(1 - \sigma^2)]r_{i+1/2}^-. \end{aligned} \tag{2.56}$$

1) By setting $\mu_{i+1/2} \leq \mu_0$, one has

$$\begin{aligned} R = (1 - \sigma^2)(1 - 2\mu_0) &+ [\sigma^2/2 - \sigma/2 + \mu_{i+3/2}(1 - \sigma^2)]r_{i+1/2}^+ \\ &+ [\sigma^2/2 + \sigma/2 + \mu_{i-1/2}(1 - \sigma^2)]r_{i+1/2}^-. \end{aligned}$$

For $\sigma > 0$, given the ratio $r_{i+1/2}^- > 0$, it is sufficient that

$$\begin{aligned} &(1 - \sigma^2)(1 - 2\mu_0) + [\sigma^2/2 - \sigma/2 + \mu_{i+3/2}(1 - \sigma^2)]r_{i+1/2}^+ \geq 0 \\ &(1 + \sigma)(1 - 2\mu_0) - [\sigma/2 - (1 + \sigma)\mu_{i+3/2}]r_{i+1/2}^+ \geq 0 \\ &(1 - 2\mu_0) - (\mu_{up} - \mu_{i+3/2})r_{i+1/2}^+ \geq 0 \\ &\mu_{i+3/2} \geq \mu_{up} - (1 - 2\mu_0)/r_{i+1/2}^+ \\ &\mu_{i+3/2} \geq \mu_{up} - (1 - 2\mu_0)r_{i+3/2}^-, \end{aligned}$$

or,

$$\mu_{i+1/2} \geq \mu_{up} - (1 - 2\mu_0)r_{i+1/2}^-; \tag{2.57}$$

similarly for $\sigma < 0$, given the ratio $r_{i+1/2}^+ > 0$, (2.56) leads to

$$\mu_{i+1/2} \geq \mu_{up} - (1 - 2\mu_0)r_{i+1/2}^+. \tag{2.58}$$

It is sufficient for inequalities (2.57) and (2.58) that

$$\mu_{i+1/2} \geq \mu_{up} - (1 - 2\mu_0) r_{i+1/2}. \tag{2.59}$$

2) On the other hand, given $\sigma > 0$ and the ratio $r^-_{i+1/2} > 0$, relation (2.56) leads to

$$\begin{aligned} R &\geq (1-\sigma^2)(1-2\mu_{i+1/2}) + [\sigma^2/2 - \sigma/2 + \mu_{i+3/2}(1-\sigma^2)] r^+_{i+1/2}, \\ &= (1-\sigma^2)[(1-2\mu_{i+1/2}) - (\mu_{up} - \mu_{i+3/2}) r^+_{i+1/2}], \\ &\geq (1-\sigma^2)[(1-2\mu_{i+1/2}) - (1-2\mu_0) r_{i+1/2} r^+_{i+1/2}], \\ &\geq (1-\sigma^2)[(1-2\mu_{i+1/2}) - (1-2\mu_0)], \\ &\geq 2(1-\sigma^2)(\mu_0 - \mu_{i+1/2}), \end{aligned}$$

where (2.59) and $r_{i+3/2} r^+_{i+1/2} \leq 1$ are used; then it is sufficient for $R \geq 0$ if

$$\mu_{i+1/2} \leq \mu_0. \tag{2.60}$$

The case of $\sigma < 0$ gives the same results. Let $\mu_{up} \leq \mu_0 \leq 1/2$, a sufficient condition for (2.59) and (2.60) is

$$\mu_0 - (1 - 2\mu_0) r_{i+1/2} \leq \mu_t \leq \mu_0.$$

This completes the proof of *Lemma 4*.

By choosing the lower limit for μ_t, the viscosity coefficient (2.54) becomes

$$\mu = \min[\mu_0, \max(0, \mu_0 - (1 - 2\mu_0) r_{i+1/2})]. \tag{2.61}$$

For convenience let

$$\mu_0 = \alpha \mu_{up}. \tag{2.62}$$

It can be shown that the coefficient (2.61) is identical to (2.51) for $\alpha = 1$ or $\mu_0 = \mu_{up}$.

To be able to guarantee the scheme with the coefficient (2.61) is second-order accurate, the scheme should switch to the Lax-Wendroff scheme in smoothing regions. Because for $r_{i+1/2} = 1$ coefficient μ must be 0, one has $\mu_0 \leq 1/3$. Using notation α, a sufficient condition is

$$1 \leq \alpha \leq 4/3. \tag{2.63}$$

2.2.3 Numerical experiments

We briefly summarize the results in previous sections for solving equation (2.1). The Lax-Wendroff scheme can be made to be TVD by adding artificial dissipation. The scheme is a conservative one,

$$u_i^{n+1} = u_i^n - \lambda(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}), \tag{2.64}$$

with the numerical flux

$$\begin{aligned} u^*_{i+1/2} &= u_{i+1/2} - \lambda(1/2 - \mu)\Delta_{i+1/2} f, \\ \hat{f}_{i+1/2} &= f(u^*_{i+1/2}) - \mu\Delta_{i+1/2} u/\lambda, \end{aligned} \tag{2.65}$$

where μ is a smoothing coefficient. Its geometrical meaning is shown in Fig. 2.1c. A variety of schemes may be designed by defining coefficient μ. One may transform Roe's symmetric TVD Lax-Wendroff by following

$$\mu = \mu_{up}(1 - Q), \tag{2.66}$$

where Q is a limiter, for example limiter (2.19), (2.20) or (2.50). A somewhat generalized coefficient is

$$\mu = \min[\mu_0, \max(0, \mu_0 - (1 - 2\mu_0)r_{i+1/2})], \tag{2.67}$$

where $\mu_0 = \alpha\mu_{up}$ and $r_{i+1/2}$ is defined as (2.52). The scheme (2.64) (2.65) with the coefficient (2.67) is monotonicity preserving and second order accurate for $1 \leq \alpha \leq 4/3$. We remark that $\alpha = 1$ is equivalent to Roe's symmetric approach using limiter (2.50) in solving linear scalar equations. Although $\alpha = 1$ corresponds to the least dissipation, a slightly larger α is able to suppress the nonlinear instability in solving nonlinear conservation laws.

In this section we present some numerical experiments to verify the performance of the symmetric TVD scheme via smoothing. All schemes tested here are using the form of (2.64) (2.65) with different coefficient μ;

1. $\mu = 0$, the Lax-Wendroff scheme, denoted by "Lax-Wendroff",
2. $\mu = \mu_{up}$, the first-order upwind scheme, denoted by "First-order upwind",
3. $\mu = \mu_{up}(1 - Q)$, with limiter (2.19), denoted by "Roe's TVD",

4. using (2.48) and (2.49), denoted by "Smoothing TVD A",

5. using (2.67) with $\alpha = 1$, denoted by "Smoothing TVD B".

A square wave with 19 nodes on the top is used as an initial condition of the linear wave equation. Fig. 2.4ab show the results of above mentioned schemes. It is seen that the Lax-Wendroff scheme produces post-shock oscillations, while the first-order upwind scheme smears the solution. All nonlinear TVD schemes perform much better. Note that the smoothing TVD B is less diffusive that the smoothing TVD A.

Fig. 2.5ab compares the performance of two smoothing schemes under different CFL numbers. The smoothing TVD B is more diffusive at low CFL numbers. In Fig. 2.5b, although $\sigma = 0.5$ is five times as large as $\sigma = 0.1$, the difference between their results is small. It suggests that the profile of $\sigma = 0.1$ is close to the asymptotic solution of the scheme. The smoothing TVD A performs rather irregularly, due to its discontinuous coefficient. Therefore only the coefficient (2.67) will be applied hereafter.

2.3 Extension to system of one-dimensional conservation laws

We are interested in the solution to the initial value problem of one-dimensional hyperbolic conservation laws

$$\mathbf{U}_t + \mathbf{F}_x = 0, \tag{2.68}$$

where $\mathbf{U}$ and $\mathbf{F}$ are vectors with equal length. For the 1-D Euler equations, they reads, using the conservative variables $\mathbf{U} = (\rho, m, \varepsilon)$,

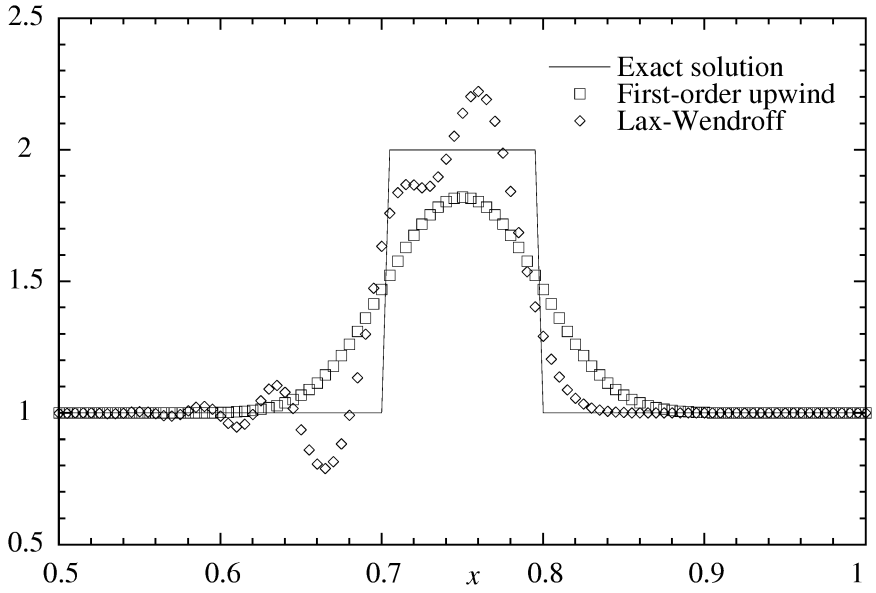
$$\mathbf{U} = \begin{pmatrix} \rho \\ m \\ \varepsilon \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} m \\ m^2/\rho + p \\ m/\rho(\varepsilon + p) \end{pmatrix}, \tag{2.69}$$

or, using the primitive variables $\mathbf{M} = (\rho, u, p)$,

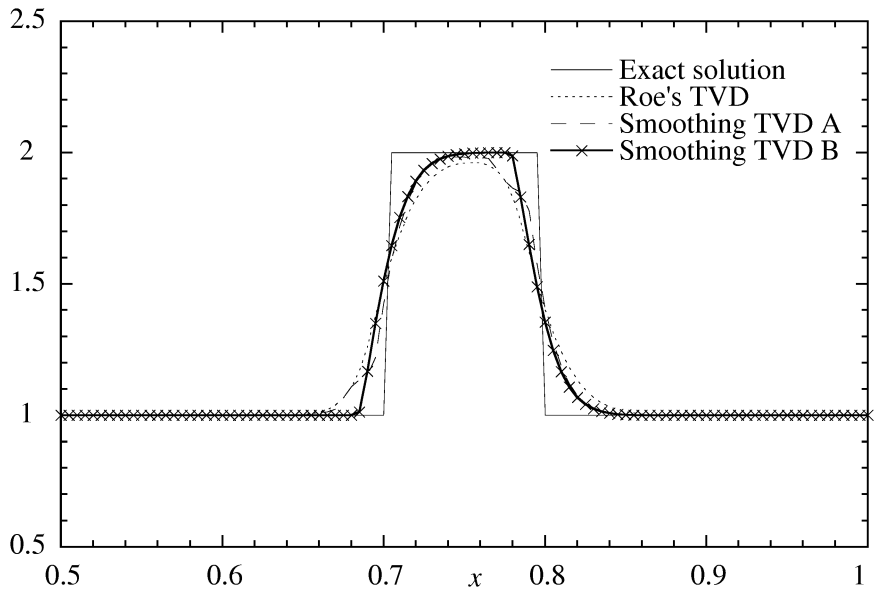
$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho e \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u u + p \\ \rho e u + p u \end{pmatrix}, \tag{2.70}$$

and

$$p = (\gamma - 1)[\varepsilon - m^2/(2\rho)]; \qquad e = \varepsilon/\rho.$$

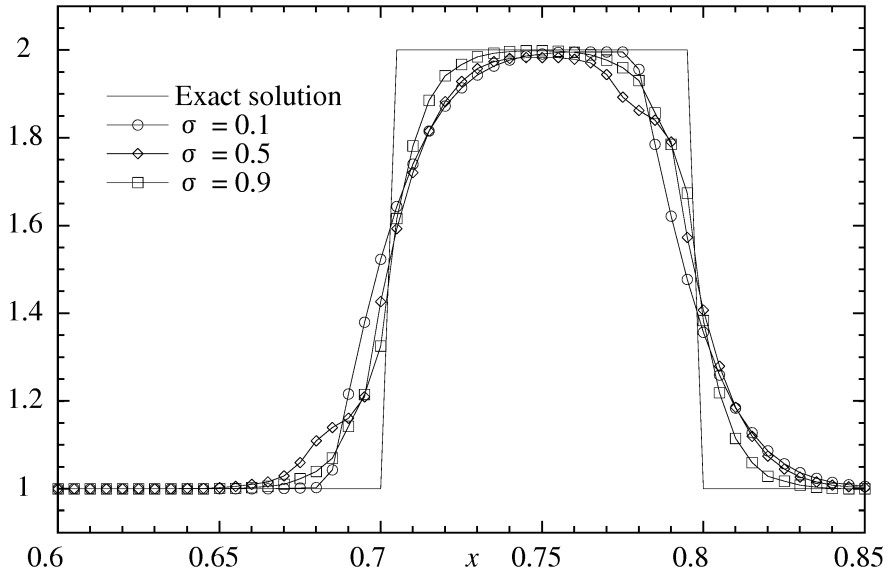


a.

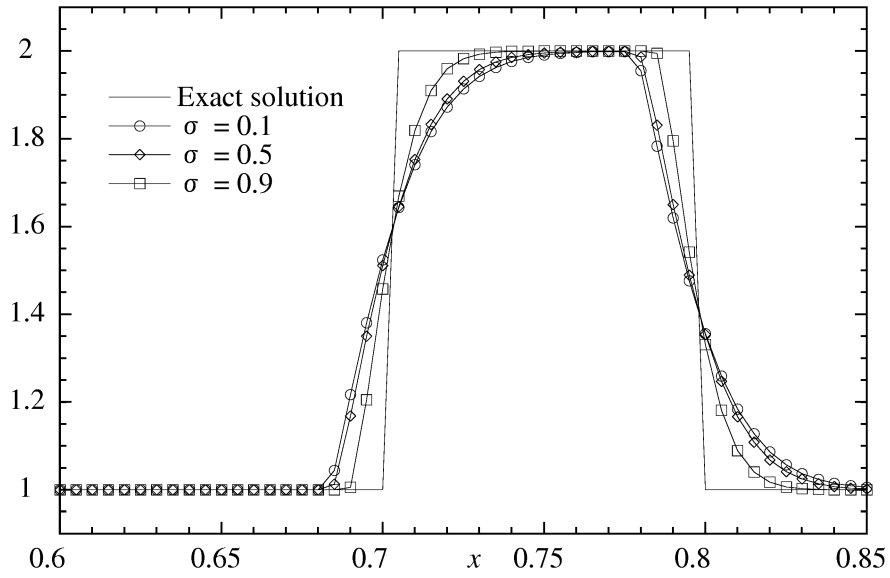


b.

Figure 2.4. Convection of a square wave, $\sigma = 0.5$, 200 steps. **a.** results of the first-order upwind scheme and the Lax-Wendroff scheme; **b.** results of smoothing TVD scheme and Roe's symmetric TVD scheme using limiter (2.19).



a.



b.

Figure 2.5. Convection of a square wave under different CFL numbers by the smoothing TVD scheme **a.** smoothing TVD A; **b.** smoothing TVD B. 1000, 200 and 112 steps for $\sigma = 0.1, 0.5, 0.9$ respectively.

System of equations (2.68) can be written as

$$\mathbf{U}_t + \mathbf{A}\mathbf{U}_x = 0, \tag{2.71}$$

where matrix $\mathbf{A} = \partial\mathbf{F}/\partial\mathbf{U}$. If the eigenvalues of $\mathbf{A}$ are real and the eigenvectors exist, the system of equations is hyperbolic. The numerical solution of (2.68) is one of the most challenging problems, due to the possible appearance of discontinuous solutions, even from smooth initial data.

Remark The Euler equations in the form of primitive variables (2.70) is able to be simplified to

$$\begin{aligned} &\rho_t + \rho u_x + u\rho_x = 0, \\ &u_t + uu_x + 1/\rho p_x = 0, \\ &p_t + \gamma p u_x + up_x = 0, \end{aligned} \tag{2.72}$$

which will be used to in the predictor step to enhance efficiency. ¶

2.3.1 Extension via the local characteristic approach

The extension technique is the so-called local characteristic approach. The approach was originally suggested by Roe, and then generalized by Harten (see [1]). The basic idea is to extend a scalar scheme to the system case by applying it scalarly to each of the characteristic variables. If the system is nonlinear, it has to be locally linearized.

Denote $\mathbf{R}$ as the matrix whose columns are right eigenvectors of $\mathbf{A}$. Then one has

$$\mathbf{A} = \mathbf{R}\mathbf{\Lambda}\mathbf{R}^{-1},$$

where $\mathbf{\Lambda}$ is the matrix of eigenvalues. Letting $\mathbf{W} = \mathbf{R}^{-1}\mathbf{U}$ and multiplying (2.71) by $\mathbf{R}^{-1}$ gives

$$\begin{aligned} \mathbf{R}^{-1}\mathbf{U}_t + \mathbf{R}^{-1}\mathbf{A}\mathbf{U}_x &= 0, \\ \mathbf{W}_t + (\mathbf{R}^{-1}\mathbf{A}\mathbf{R})\mathbf{W}_x &= 0, \\ \mathbf{W}_t + \mathbf{\Lambda}\mathbf{W}_x &= 0, \end{aligned} \tag{2.73}$$

which is a system of uncoupled scalar equations. For the Euler equations, $\mathbf{R}$, $\mathbf{R}^{-1}$ are

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 1 \\ u-c & u & u+c \\ H-uc & u^2/2 & H+uc \end{pmatrix}, \ \mathbf{R}^{-1} = \begin{pmatrix} (b_1+u/c)/2 & -(1/c-b_2u)/2 & b_2/2 \\ 1-b_1 & b_2u & -b_2 \\ (b_1-u/c)/2 & (1/c-b_2u)/2 & b_2/2 \end{pmatrix},$$

where $H = (\varepsilon + p)/\rho = c^2/(\gamma - 1) + u^2/2$, and

$$b_2 = (\gamma - 1)/c^2, \qquad b_1 = b_2 u^2/2.$$

The eigenvalues $\mathbf{\Lambda}$ are

$$\mathbf{\Lambda} = \text{diagonal } (u - c,\ u,\ u + c). \tag{2.74}$$

Rewrite (2.73) in the scalar form, one has

$$W_t^k + \Lambda^k W_x^k = 0, \tag{2.75}$$

where $k = 1, 2, 3$ denote individual equations. Each equation can now be solved by the nonlinear smoothing scheme, following (2.21),

$$\begin{aligned} F_{i+1/2}^k &= \Lambda_{i+1/2}^k [W_{i+1/2}^k - \lambda \Lambda_{i+1/2}^k (1/2 - \mu^k)] \Delta_{i+1/2} W^k / \lambda, \\ W_i^{k,n+1} &= W_i^k - \lambda (F_{i+1/2}^k - F_{i-1/2}^k); \end{aligned} \tag{2.76}$$

or, following (2.65) using the predictor step,

$$\begin{aligned} W_{i+1/2}^{*,k} &= W_{i+1/2}^k - \lambda \Lambda_{i+1/2}^k (1/2 - \mu^k) \Delta_{i+1/2} W^k, \\ F_{i+1/2}^k &= \Lambda_{i+1/2}^k W_{i+1/2}^{*,k} - \mu^k \Delta_{i+1/2} W^k / \lambda, \\ W_i^{k,n+1} &= W_i^k - \lambda (F_{i+1/2}^k - F_{i-1/2}^k). \end{aligned} \tag{2.77}$$

Multiplied by $\mathbf{R} = (R^{lk})$ $(l, k = 1, 2, 3)$, (2.76) leads to

$$\begin{aligned} \hat{F}_{i+1/2}^l &= F^l(\mathbf{U}_{i+1/2}) - \sum_k \left\{ R^{lk} [\lambda (\Lambda_{i+1/2}^k)^2 (1/2 - \mu^k) + \mu^k/\lambda] \Delta_{i+1/2} W^k \right\}, \\ U_i^{l,n+1} &= U_i^l - \lambda (\hat{F}_{i+1/2}^l - \hat{F}_{i-1/2}^l); \end{aligned} \tag{2.78}$$

while (2.77) leads to

$$\begin{aligned} U_{i+1/2}^{*,l} &= U_{i+1/2}^l - \lambda \sum_k \left\{ R^{lk} [\Lambda_{i+1/2}^k (1/2 - \mu_{i+1/2}^k) \Delta_{i+1/2} W^k] \right\}, \\ \hat{F}_{i+1/2}^l &= F^l(\mathbf{U}^*{}_{i+1/2}) - \sum_k \left\{ R^{lk} \mu_{i+1/2}^k \Delta_{i+1/2} W^k / \lambda \right\}, \\ U_i^{l,n+1} &= U_i^l - \lambda (\hat{F}_{i+1/2}^l - \hat{F}_{i-1/2}^l), \end{aligned} \tag{2.79}$$

where U^l and F^l denote the elements of vectors $\mathbf{U}$ and $\mathbf{F}$. In calculating $\mu_{i+1/2}^k$, the local CFL number, defined as

$$|\sigma_{i+1/2}^k| = \lambda \max(|\Lambda_i^k|, |\Lambda_{i+1}^k|),$$

is used for each eigenequation.

Remark Note that $U^l_{i+1/2}$ is simply set to be arithmetic average $(U^l_i + U^l_{i+1})/2$. The flux function $F^l(\mathbf{U^*}_{i+1/2})$ is exactly evaluated by its definition in (2.69) or (2.70), instead of $((F^l(\mathbf{U^*}_i) + (F^l(\mathbf{U^*}_{i+1}))/2$. The former has an advantage that it is evaluated at any location where $\mathbf{U}^*$ is known, therefore it can be directly performed at the center of an interface when using the finite volume method on an unstructured grid.

2.3.2 Extension via scalar artificial viscosity

Suppose a scalar artificial viscosity μ exists in solving the system of the Euler equations, by (2.79), one has

$$\begin{aligned} U^{*,l}_{i+1/2} &= U^l_{i+1/2} - \lambda(1/2 - \mu_{i+1/2})\Delta_{i+1/2}F^l, \\ \hat{F}^l_{i+1/2} &= F^l(\mathbf{U}^*_{i+1/2}) - \mu_{i+1/2}\Delta_{i+1/2}U^l/\lambda, \\ U^{l,n+1}_i &= U^l_i - \lambda(\hat{F}^l_{i+1/2} - \hat{F}^l_{i-1/2}). \end{aligned} \tag{2.80}$$

Let gradient $U^l_x = \Delta U^l/\Delta x$ and $F^l_x = \Delta F^l/\Delta x$, (2.80) becomes,

$$\begin{aligned} U^{*,l} &= U^l - (1/2 - \mu_{i+1/2})\Delta t F^l_x, \\ \hat{F}^l_{i+1/2} &= F^l(\mathbf{U}^*) - \mu(\Delta x)^2/\Delta t U^l_x, \\ U^{l,n+1}_i &= U^l_i - \lambda(\hat{F}^l_{i+1/2} - \hat{F}^l_{i-1/2}), \end{aligned} \tag{2.81}$$

or, in the vector form,

$$\begin{aligned} \mathbf{U}^* &= \mathbf{U} - (1/2 - \mu_{i+1/2})\Delta t \mathbf{F}_x, \\ \hat{\mathbf{F}}_{i+1/2} &= \mathbf{F}(\mathbf{U}^*) - \mu_{i+1/2}(\Delta x)^2/\Delta t \mathbf{U}_x, \\ \mathbf{U}^{n+1}_i &= \mathbf{U}_i - \lambda(\hat{\mathbf{F}}_{i+1/2} - \hat{\mathbf{F}}_{i-1/2}), \end{aligned} \tag{2.82}$$

where subscript $i + 1/2$ of some variables is removed for clearance. The first two equations in (2.82) evaluate the flux through an interface $i + 1/2$ completely based on *local* values and gradients. The flux evaluations depends on only two nodes i and $i+1$, therefore it is compact. Although as far as arithmetic operations are concerned scheme (2.82) might not be the most efficient on a structured grid, the scheme can be easily extended to any unstructured grids after gradient reconstruction. We will discuss how to realize the compactness on an unstructured grid in solving multi-dimensional flows.

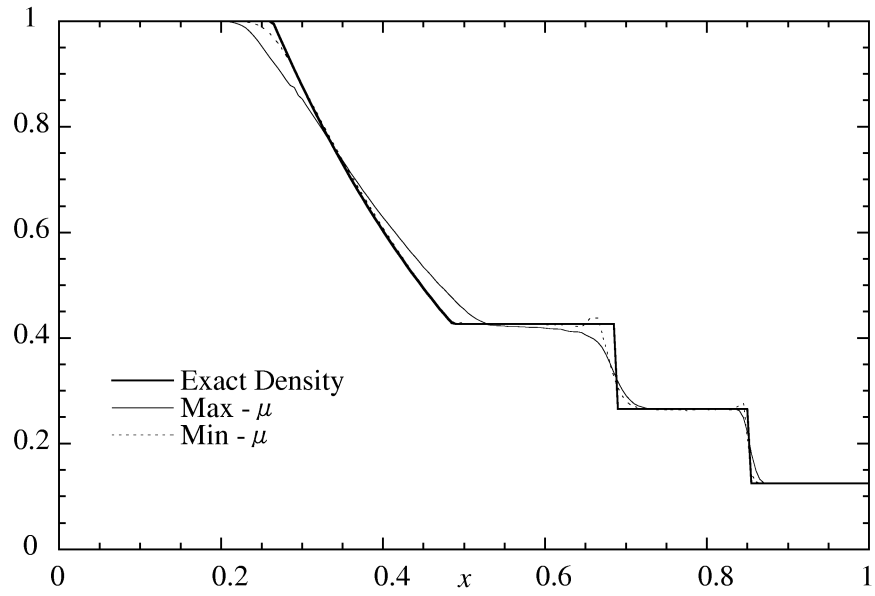


Figure 2.6. Numerical results obtained by using two scalar coefficients: **a.** scalar coefficient (2.83), denoted by $\mathrm{Max}-\mu$; **b.** scalar coefficient (2.84), denoted by $\mathrm{Min}-\mu$.

Consider the largest value in three coefficients of the eigenequations as a scalar coefficient, i.e.,

$$\mu = \max(\mu^1,\ \mu^2,\ \mu^3), \tag{2.83}$$

which is a sufficient condition to achieve TVD for all waves. Another possible scalar coefficient is a necessary condition

$$\mu = \min(\mu^1,\ \mu^2,\ \mu^3). \tag{2.84}$$

It is clearly not sufficient for some waves which require a larger μ. One may expect that it might generate oscillations. Fig. 2.6 shows the result of a shock tube problem by using the two scalar coefficients. It is seen that coefficient (2.83) does produce a monotone profile, although it is somewhat too diffusive. Coefficient (2.84) gives a result much less diffusive and more accurate in smooth regions, but generates some visible oscillations.

Both the scalar coefficients require time-consuming calculation of the characteristic variables W^k. Two possible ways of simplification are one suggested by Davis,

and one suggested by Roe (see the review by Yee [10]). Davis proposed a scalar limiter via the original variables instead of characteristic variables and the viscosity coefficient is determined by the fastest wave. Roe suggested the use of the strongest wave. Because all primitive or conservative variables are functions of characteristic variables, it is intuitively believed that a scalar limiting procedure via the primitive or conservative variables should lie between (2.84) and (2.83). We propose a scalar indicator:

$$\phi_i = \frac{\sum\limits_k \left\{ |U^k_{i+1} + U^k_{i-1} - 2U^k_i| / \bar{U}^k_i \right\}}{\varepsilon_0 + \sum\limits_k \left\{ |U^k_{i+1} - U^k_{i-1}| / \bar{U}^k_i \right\}}, \tag{2.85}$$

and the scalar coefficient μ follows (2.67) with

$$\mu_{up} = \frac{\sigma^{fast}}{2(1+\sigma^{fast})},$$

$$\sigma^{fast} = \lambda \max(|u_i| + c_i, |u_{i+1}| + c_{i+1}).$$

All differences between variables U^k in (2.85) are normalized by characteristic quantities $\bar{U}^k_i$. We may choose $\bar{U}^k_i$ to be the local conservative values,

$$\begin{aligned} \bar{\rho}_i &= \rho_i \\ \bar{m}_i &= \max(|m_i|, \rho_i) \\ \bar{\varepsilon}_i &= \varepsilon_i \end{aligned} \tag{2.86}$$

where the momentum $\bar{m}_i$ is slightly modified because it might be negative and zero. In calculation we set $\varepsilon_0 = (\Delta x)^2/10$ to avoid division by zero. It is easily seen that in the case of single scalar equation (2.85) becomes (2.37).

Remark One may construct this kind of scalar coefficient by following Roe's formulation, in the form of (2.7),

$$\begin{aligned} C_{i+1/2} &= -\sigma/2 + \sigma^2/2 + \mu, \\ C_{i-1/2} &= \sigma/2 + \sigma^2/2 + \mu. \end{aligned} \tag{2.87}$$

The TVD region of this μ is shown in Fig. 2.7b, compared with the TVD region of the present approach shown in Fig. 2.7a. In figure **a**, the minimum coefficient is a monotone function of the CFL number $|\sigma|$; therefore a scalar coefficient which is defined by the fastest wave is sufficient for other waves. However, there is an extreme value $|\sigma| = 1/2$ in Fig. 2.7b. A trouble of the existence of the extreme value is that

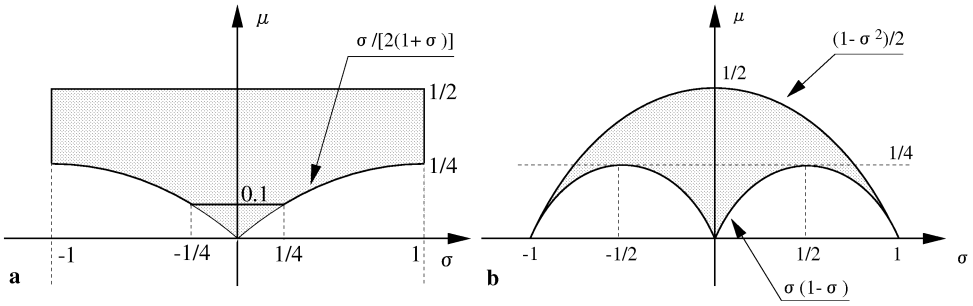


Figure 2.7. TVD region of artificial viscosity coefficient μ as a function of σ. **a.** The proposed artificial viscosity coefficient μ defined in (2.21), or more clearly in (2.23); **b.** The artificial viscosity coefficient μ defined by (2.87), a similar coefficient was used by Davis. In figure **a**, the minimum coefficient is a monotone function of the CFL number $|\sigma|$; while there is an extreme value for $|\sigma| = 1/2$ in figure **b**. A trouble of the existence of the extreme value is that defining μ according to a fast wave or larger $|\sigma|$ is not sufficient for slow waves in solving the system of the conservation laws.

defining μ according to a fast wave or larger $|\sigma|$ is not sufficient for slow waves in solving the system of conservation laws. One way to solve it is, suggested by Davis, choosing $\mu = |\sigma|/(1 - |\sigma|)$ for $|\sigma| \leq 1/2$ and $\mu = 1/4$ for $|\sigma| \geq 1/2$. Unfortunately it can be seen in the figure this method is not TVD for whole region of $|\sigma| \leq 1$. ¶

The predictor step can be calculated by, denoting $U_t^* = (U^* - U)/[(1/2 - \mu_{i+1/2})\Delta t]$ and using the gradients of conservative variables

$$\begin{aligned} \rho_t^* &= -m_x \\ m_t^* &= -(2m/\rho m_x - m^2/\rho^2 \rho_x + p_x) \\ \varepsilon_t^* &= -[((\varepsilon + p)m_x + m(\varepsilon_x + p_x))/\rho - m(\varepsilon + p)/\rho^2 \rho_x], \end{aligned} \tag{2.88}$$

where $p_x = (\gamma - 1)(\varepsilon_x - m/\rho m_x + m^2/(2\rho^2)\rho_x$. (2.88) is just a reformulation of (2.68) with (2.69) in terms of derivatives of the conservative variables.

Remark The predictor step can also be computed by (2.72), using the gradients of the primitive variables

$$\begin{aligned} \rho_t^* &= -(\rho u_x + u\rho_x), \\ u_t^* &= -(uu_x + 1/\rho p_x), \\ p_t^* &= -(\gamma p u_x + u p_x), \end{aligned} \tag{2.89}$$

and then m_x and ε_x are determined by

$$\begin{aligned} m_x &= \rho^* u_x + u^* \rho_x, \\ \varepsilon_x &= p_x/(\gamma - 1) + \rho^* u^* u_x + 1/2(u^*)^2 \rho_x. \end{aligned} \tag{2.90}$$

In general, it is not efficient to constructed the gradients of both the primitive and conservative variables. To be consistent with the predictor step using the primitive variables, the $U^k,\ k = (1, 2, 3)$ in the scalar indicator (2.85) is replaced by ρ, u and p, and corresponding characteristic quantities are

$$\begin{aligned} \bar{\rho}_i &= \rho_i \\ \bar{u}_i &= \max(|u_i|, \rho_i) \\ \bar{p}_i &= p_i \end{aligned} \tag{2.91}$$

It is seen that two indicators using the conservative variables or the primitive variables are totally different, however numerical tests show that two indicators work similarly in solving the Euler equations. ¶

2.3.3 Numerical experiments

In this section we present numerical results in solving one-dimensional Euler equations. The schemes tested are

1. extension via the local characteristic approach, denoted by "Roe",
2. extension via the scalar viscosity using the conservative variables, denoted by "STVD A",
3. extension via the scalar viscosity using the primitive variables, denoted by "STVD B",

and the artificial viscosity coefficient follows (2.61). The local CFL number $|\sigma^k|$ or $|\sigma^{fast}|$ is taken as

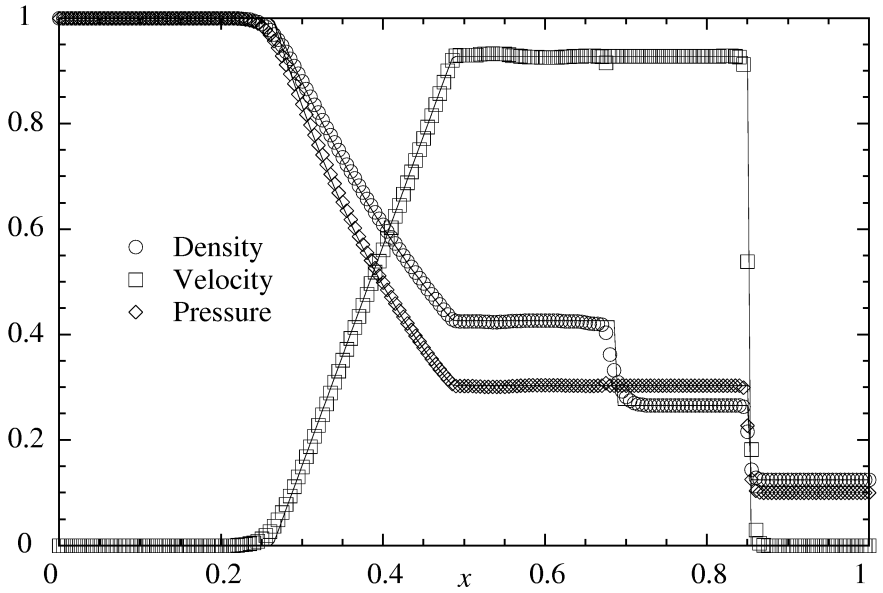
$$|\sigma^k| = \max(1/4, |\sigma^k|),$$
$$|\sigma^{fast}| = \max(1/4, |\sigma^{fast}|)$$

to guarantee the entropy condition. The matrix $\mathbf{R}$ and $\mathbf{R}^{-1}$ are calculated by the arithmetic averaged values of two neighboring nodes in scheme "Roe".

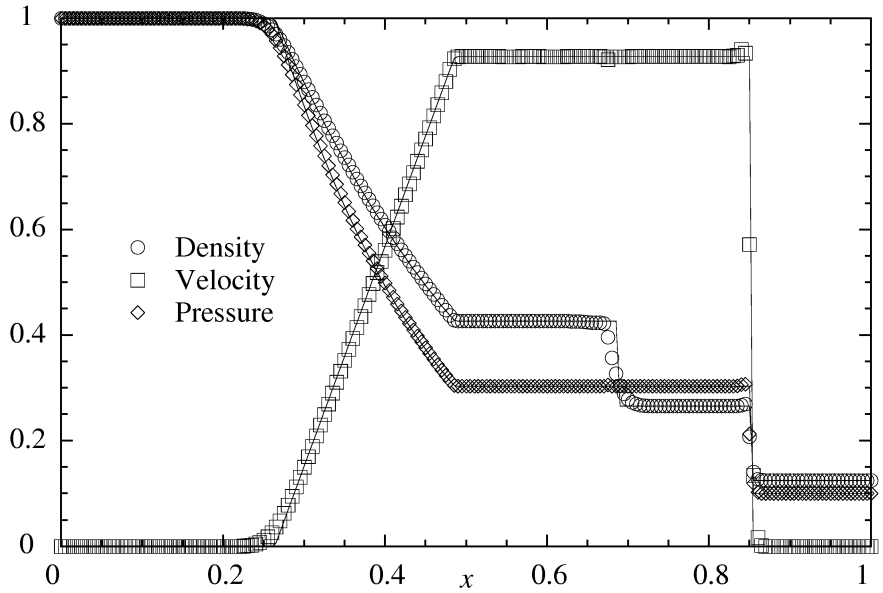
For comparison Harten's original TVD scheme [1], denoted by "Harten", is chosen, and Roe's approximate Riemann solver [11] is used.

Sod's shock tube problem with left-side and right-side values

$$(\rho, u, p)_L = (1, 0, 1),$$



a.



b.

Figure 2.8. Sod's problem solved by scheme "Roe", $\alpha = 1.0$: **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$.

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

is tested in this section. Fig. 2.8ab is the results of scheme "Roe". The scheme performs similarly at two different CFL numbers 0.2 and 0.8, except a low peak for $\sigma = 0.8$ behind the shock wave. The results of STVD A in Fig. 2.9ab is almost identical to that of "Roe". STVD B is a little more diffusive than STVD A, but it could suppress nonlinear peaks behind the shock front better. All these symmetric scheme performs not worse than the classic Harten's upwind TVD scheme as shown in Fig. 2.11ab.

The post-shock peak appeared in Fig. 2.8b and Fig. 2.9b can be easily dampened by choosing a larger α. A result of $\alpha = 1.2$ is shown in Fig. 2.12 . It produces the monotone profile across the shock without smearing much in other regions.

2.4 Extension to system of multi-dimensional conservation laws

The Euler equations in two dimensions are given by

$$\mathbf{U}_t + \mathbf{F}_x + \mathbf{G}_y = 0, \tag{2.92}$$

where the conservative state vector $\mathbf{U}$ and the flux functions $\mathbf{F}$ and $\mathbf{G}$ are defined as

$$\mathbf{U} = \begin{Bmatrix} \rho \\ m \\ n \\ \varepsilon \end{Bmatrix}, \mathbf{F} = \begin{Bmatrix} m \\ m^2/\rho + p \\ mn/\rho \\ m/\rho(\varepsilon + p) \end{Bmatrix}, \mathbf{G} = \begin{Bmatrix} n \\ mn/\rho \\ n^2/\rho + p \\ n/\rho(\varepsilon + p) \end{Bmatrix}. \tag{2.93}$$

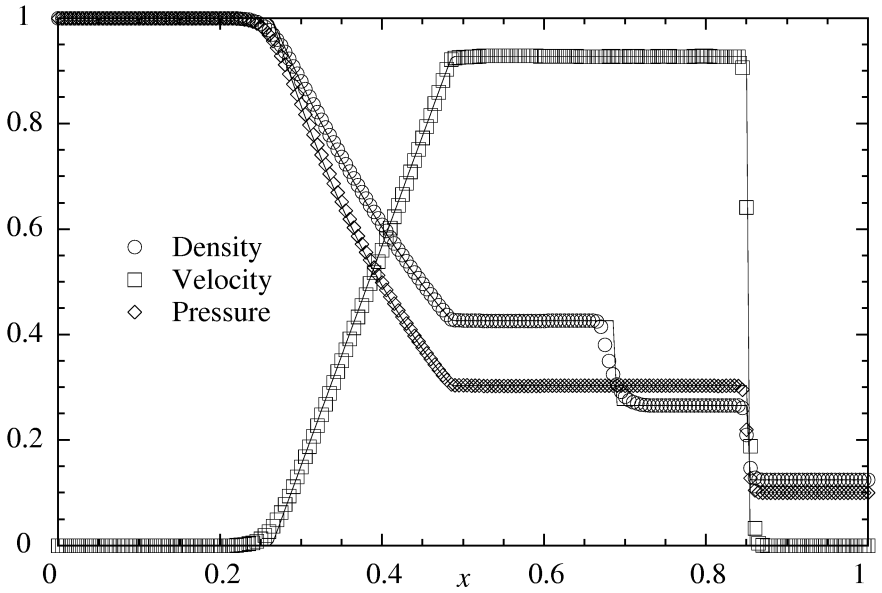
In this system, m and n are the momentum components in x and y directions, respectively. The static pressure p is given by

$$p = (\gamma - 1)[\varepsilon - (m^2 + n^2)/(2\rho)].$$

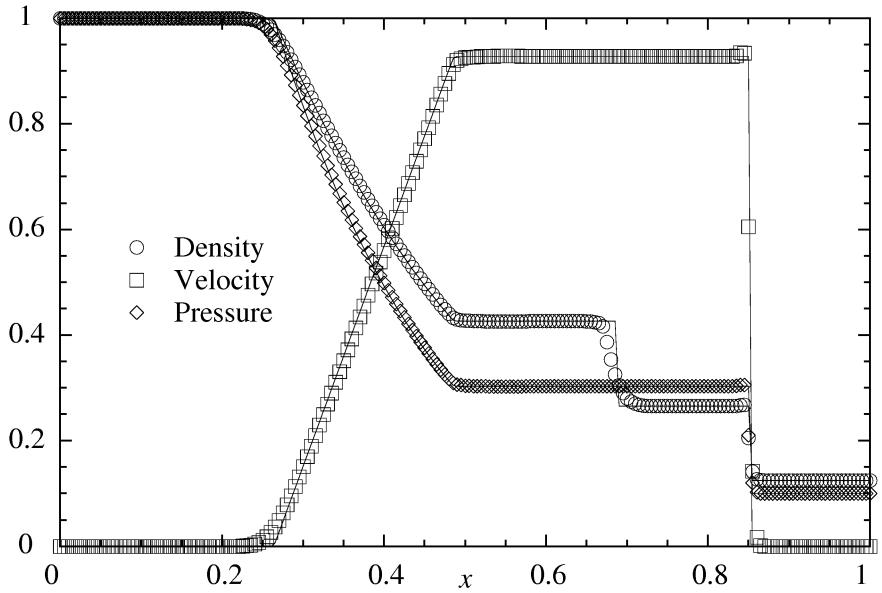
Following the finite volume method, the change of the conservative values of a control volume is equal to the sum of fluxes through its interfaces, i.e.

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i - \frac{\Delta t}{\Delta \Omega} \sum_{faces} \hat{\mathbf{F}}, \tag{2.94}$$

where $\Delta\Omega$ is the area of the control volume, and $faces$ denotes all interfaces which surround the control volume. $\hat{\mathbf{F}}$ is just the flux through one of the interfaces.

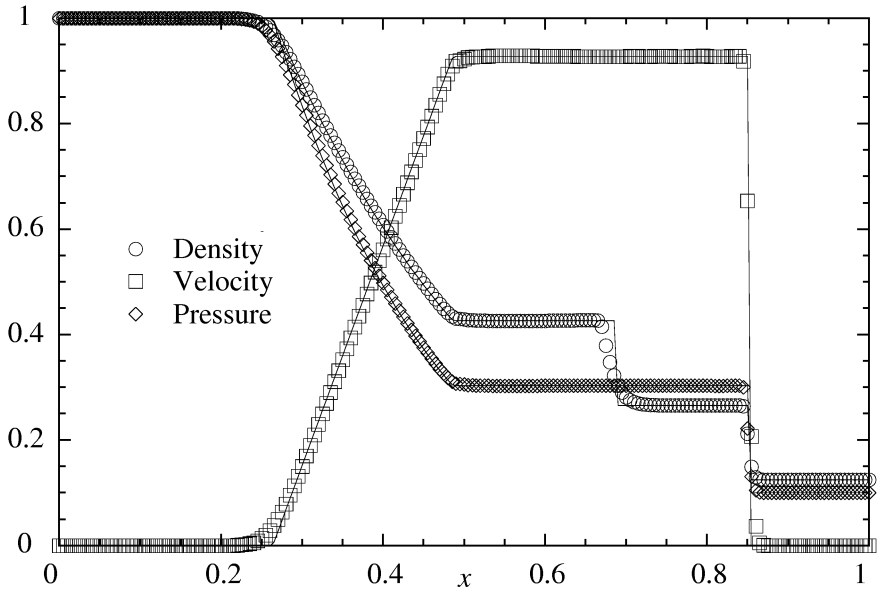


a.

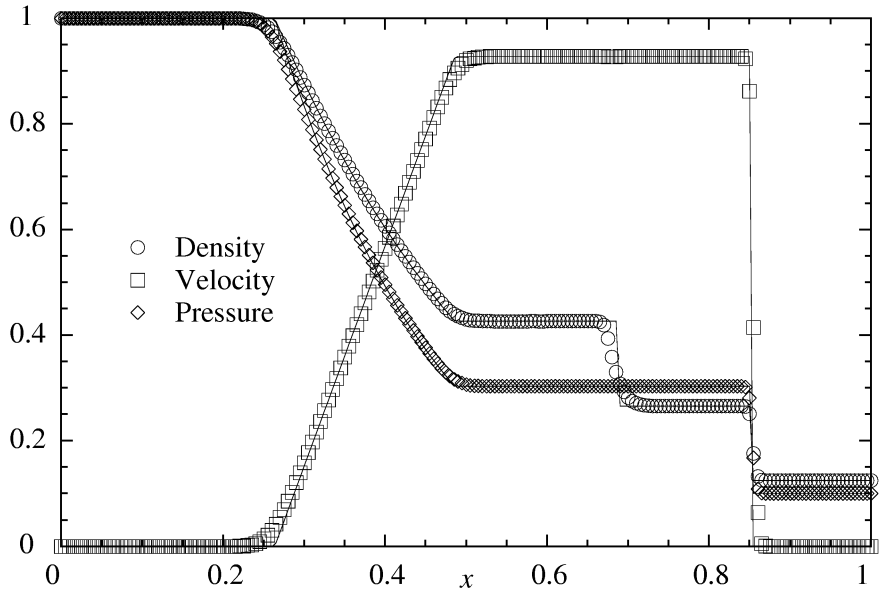


b.

Figure 2.9. Sod's problem solved by scheme STVD A, $\alpha = 1.0$: **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$.

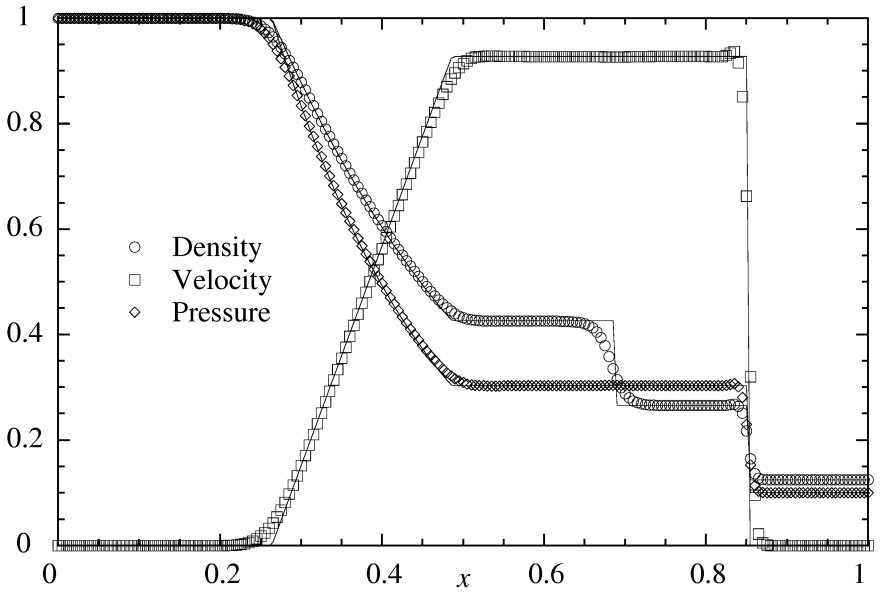


a.

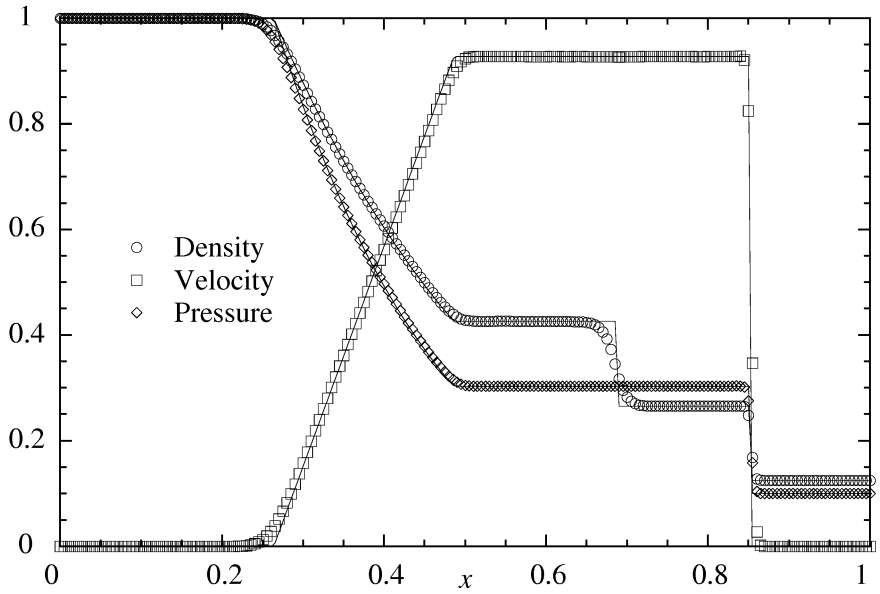


b.

Figure 2.10. Sod's problem solved by scheme STVD B, $\alpha = 1.0$: **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$.



a.



b.

Figure 2.11. Sod's problem solved by Harten's TVD scheme. **a.** $\sigma = 0.2$; **b.** $\sigma = 0.8$.

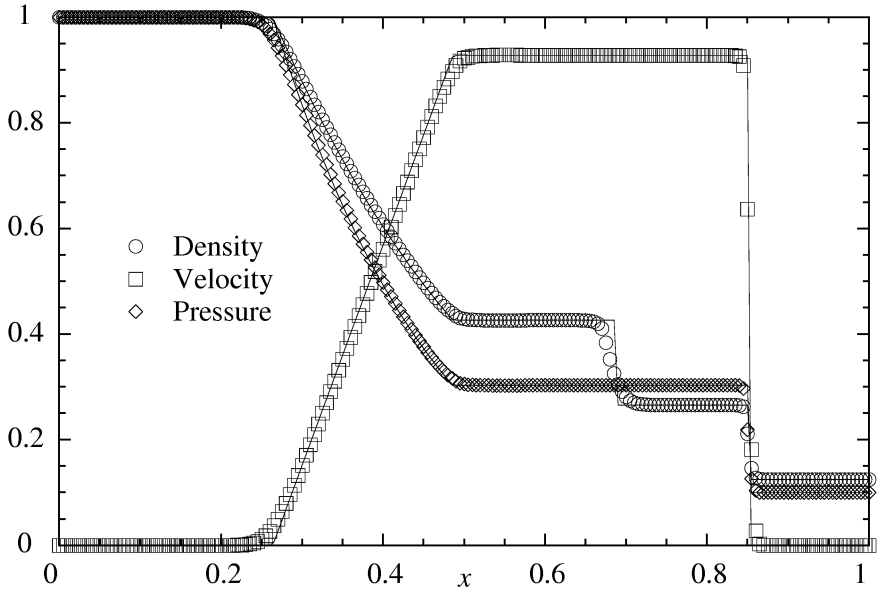


Figure 2.12. Sod's problem solved by scheme STVD A, $\alpha = 1.2$, $\sigma = 0.8$.

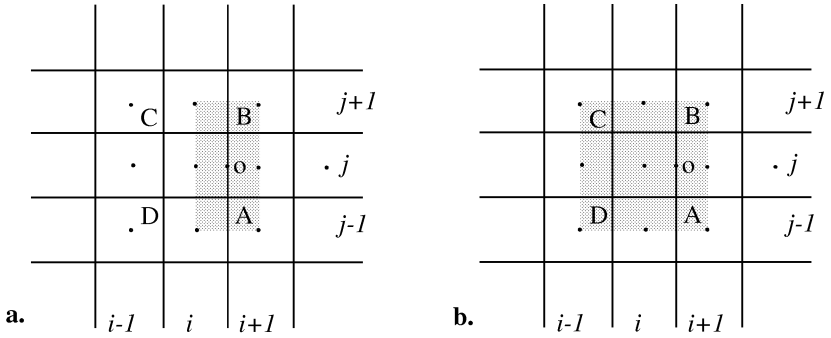


Figure 2.13. Computational stencil for the proposed scheme on the Cartesian grid: **a.** stencil for evaluating the flux through interface AB; **b.** stencil for the control volume (i, j), bounded by ABCD.

Fig. 2.13 shows a control volume on the Cartesian grid. The conservative variables are restored at the centroid of the control volumes, which is labeled with i, j in x and y direction respectively. The volume (i, j) is bounded by interfaces AB, BC, CD, and DA. The fluxes through the interfaces are required to solve (2.94). Since all interfaces can be similarly treated, we just describe the calculation of the flux through interface AB. Let $\mathbf{r}_{AB} = (dx, dy)$ is the spatial vector from A to B, and then the outward normal of AB is $(dy, -dx)$. Denote the spatial vector from i to j be $(\Delta x, \Delta y)$, the flux vector through AB, $\hat{\mathbf{F}}$ is given by

$$\mathbf{U}_o^* = \mathbf{U}_o - (1/2 - \mu_{i+1/2})\Delta t(\mathbf{F}_x + \mathbf{G}_y), \tag{2.95}$$

$$\hat{\mathbf{F}} = \mathbf{F}(\mathbf{U}_\mathbf{o}^*)dy - \mathbf{G}(\mathbf{U}_\mathbf{o}^*)dx - \mu_1\frac{(\Delta x)^2}{\Delta t}\mathbf{U}_x dy + \mu_2\frac{(\Delta y)^2}{\Delta t}\mathbf{U}_y dx, \tag{2.96}$$

where all variables and gradients are located at node o, the center of interface AB. The conservative variables at node o is the arithmetic average of two neighbors,

$$\mathbf{U} = (\mathbf{U}_{i,j} + \mathbf{U}_{i+1,j})/2.$$

The gradients of flux function $\mathbf{F}_x$ and $\mathbf{G}_y$ are given by, in terms of the gradients of the conservative variables,

$$\mathbf{F}_x = \left\{ \begin{array}{l} m_x \\ p_x + m/\rho(2m_x - m/\rho\rho_x) \\ (nm_x + mn_x - mn/\rho\rho_x)/\rho \\ (\varepsilon + p)(m_x - m/\rho\rho_x)/\rho + m(\varepsilon_x + p_x)/\rho \end{array} \right\} \tag{2.97}$$

and

$$\mathbf{G}_y = \left\{ \begin{array}{l} n_y \\ (nm_y + mn_y - mn/\rho\rho_y)/\rho \\ p_y + n/\rho(2n_y - n/\rho\rho_y) \\ (\varepsilon + p)(n_y - n/\rho\rho_y)/\rho + n(\varepsilon_y + p_y)/\rho \end{array} \right\}, \tag{2.98}$$

where

$$p_x = (\gamma - 1)[\varepsilon_x - (mm_x + nn_x)/\rho + (m^2 + n^2)/(2\rho^2)\rho_x],$$

$$p_y = (\gamma - 1)[\varepsilon_y - (mm_y + nn_y)/\rho + (m^2 + n^2)/(2\rho^2)\rho_y].$$

In (2.96), (2.97) and (2.98) the gradients of conservative variables are given by,

$$\begin{array}{l} \mathbf{U}_x = (\mathbf{U}_{i+1,j} - \mathbf{U}_{i,j})/\Delta x \\ \mathbf{U}_y = (\mathbf{U}_{i+1,j+1} + \mathbf{U}_{i,j+1} - \mathbf{U}_{i,j-1} - \mathbf{U}_{i+1,j-1})/(4\Delta y) \end{array} \tag{2.99}$$

The artificial viscosity coefficient μ is determined by (2.67) similarly as that in one-dimensional case, but the indicator ϕ_o is replaced by

$$\phi_o = \max(\phi_i, \phi_{i+1}),$$

where ϕ_i is, neglecting subsript j,

$$\phi_i = \frac{\sum\limits_k \left\{ |U^k_{i+1} + U^k_{i-1} - 2U^k_i| / \bar{U}^k_i \right\}}{\varepsilon_0 + \sum\limits_k \left\{ |U^k_{i+1} - U^k_{i-1}| / \bar{U}^k_i \right\}}, \tag{2.100}$$

with the characteristic quantities $\bar{U}^k_i$ chosen as the conservative variables at node o,

$$\begin{aligned} \bar{\rho}_i &= \rho_o, \\ \bar{m}_i &= \max(|m|_o, \rho_o), \\ \bar{n}_i &= \max(|n|_o, \rho_o), \\ \bar{\varepsilon}_i &= \varepsilon_o. \end{aligned} \tag{2.101}$$

A conceptual difference compared with the one-dimensional case is that the smoothing flux is decomposed according to the local wave direction $\mathbf{K} = (k_1, k_2)$,

$$\begin{aligned} \mu_1 &= |k_1|\mu, \\ \mu_2 &= |k_2|\mu, \end{aligned} \tag{2.102}$$

where the wave direction is approximately deduced from velocity difference between i and j,

$$\begin{aligned} k_1 &= \frac{du}{(du^2+dv^2)^{1/2}+\varepsilon_0}, \\ k_2 &= \frac{dv}{(du^2+dv^2)^{1/2}+\varepsilon_0}. \end{aligned} \tag{2.103}$$

Note that the calculation of gradients in the direction normal to the interface requires *two* cells, while in the direction along the interface requires *three* cells; the stencil is shown in Fig. 2.13a. It is also the stencil for evaluating the flux through interface AB. After summing the four fluxes, the stencil for volume (i, j) consists nine nodes, as shown in Fig. 2.13b.

Remark The extension from one-dimensional case to two dimensions follows the idea that a two-dimensional scheme should calculate a one-dimensional flow field with any orientation in the same way as the 1-D smoothing scheme do. Following this idea one should consider the possible modification of the indicator, the predictor step, and the smoothing term in flux evaluation.

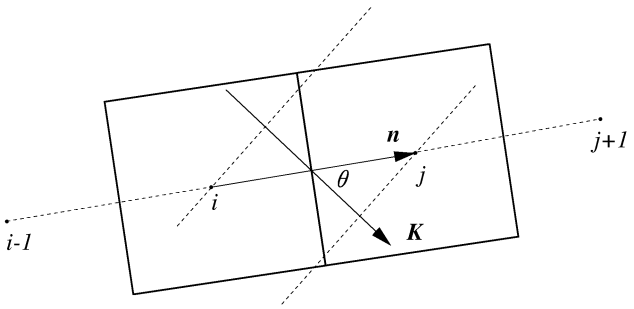


Figure 2.14. A locally one-dimensional wave lying on a two-dimensional control volume.

Fig. 2.14 shows the sketch of a wave front with direction **K** and two control volumes i and j. We are interested in the flux through their shared face, and should determine the indicator, the predictor step, and the smoothing term locally. First consider the indicator. It has been shown in previous section that the indicator is equivalent to the ratio of the second-order derivative term to the first-order one; thus a 1-D indicator along direction **K** gives

$$\phi_k = |\frac{\frac{1}{2}\xi''_{kk}\Delta x_k^2}{\xi'_k \Delta x_k}|,$$

where ξ represents a flow variable, and Δx_k denotes grid size along **K**. On the 2-D grid the simplest ratio is, along **N** which connects volume i and j as shown in Fig. 2.14,

$$\phi_n = |\frac{\frac{1}{2}\xi''_{nn}\Delta x_n^2}{\xi'_n \Delta x_n}|,$$

where Δx_n denotes grid size along **N**. Since

$$\begin{aligned} |\xi'_k| &= |\mathbf{K} \cdot \nabla\xi| = |\nabla\xi|, \\ |\xi'_n| &= |\mathbf{N} \cdot \nabla\xi| = \cos(\theta)|\nabla\xi|, \end{aligned}$$

then one has

$$|\xi'_n| = \cos(\theta)|\xi'_k|;$$

similarly

$$|\xi''_{nn}| = \cos^2(\theta)|\xi''_{kk}|.$$

Then

$$\frac{\phi_k}{\phi_n} = \frac{1}{\cos(\theta)}\frac{\Delta x_k}{\Delta x_n}.$$

It is reasonable to assume that ratio $\Delta x_k/\Delta x_n = \cos(\theta)$, by Fig. 2.14[1], then we have

$$\frac{\phi_k}{\phi_n} = 1. \tag{2.104}$$

(2.104) states that an indicator determined along grid direction **N** is the same as that along the normal of the wave front **K**. Therefore it is not essential to trace the wave direction in determining the indicator at an interface.

In the predictor step, according to the smoothing scheme, the slope of variables should be multiplied by $1 - 2\mu$, which finally leads to the modification in time step of the predictor step (2.95).

In multi-dimensional flow, the artificial viscosity coefficient is actually a tensor. Since it should be as less diffusive as possible, it is set to be zero in directions other than the normal direction of the wave front. Then the tensor is written, assuming the normal of the wave front be locally x-axis, as

$$\begin{vmatrix} \mu & 0 \\ 0 & 0 \end{vmatrix}.$$

It can be easily rotated to the Cartesian coordinates, and become

$$\begin{vmatrix} \mu_1 & 0 \\ 0 & \mu_2 \end{vmatrix}.$$

where μ_1 and μ_2 are given by (2.102). This decomposition is shown in Fig. 2.15. Having obtained the artificial coefficient in x and y directions, it is naturally to design a smoothing flux as that in (2.96). ¶

Fig. 2.16ab show results of shock diffraction obtained by the present scheme using the conservative and the primitive variables. The incident shock Mach number M_s is taken as 2. The geometry combines three squares; each of them consists of 32×32 cells. It can be seen that the shock thickness is about two cells. Although the head and tail of the shock may extend further, it shows that the flow monotonely goes through the shock discontinuity. A vortex and a secondary shock, which are close

[1]The assumption seems reasonable only for small angle θ. However for θ close to 90°, the assumption implies Δx_k approaches to zero, which is obviously incorrect. This singularity is due to difficulties in detecting a wave based on the information only at two nodes which lies on the same side of a discontinuity. Another more reasonable choice of (2.104) is to replace 1 by $\min(\sqrt{2}, 1/\cos(\theta))$, which is however less efficient.

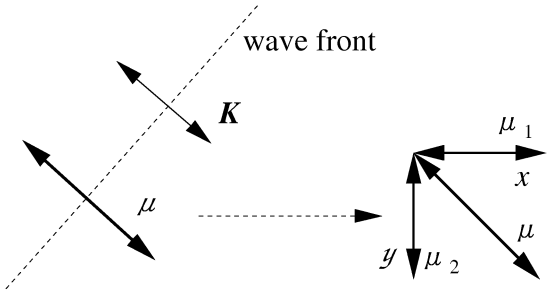


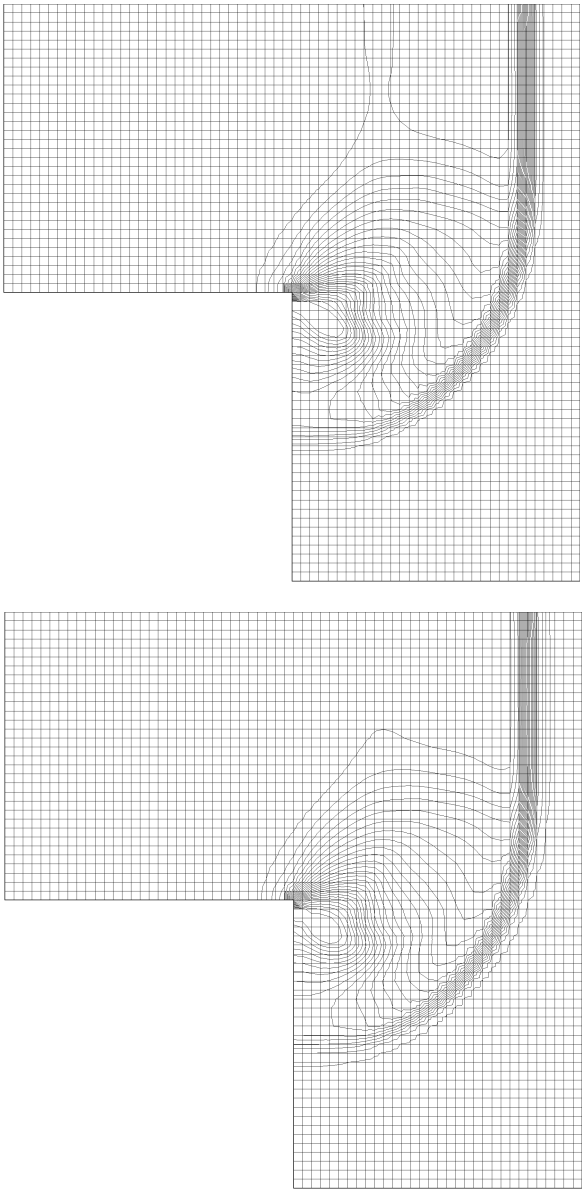
Figure 2.15. Decomposition of the artificial viscosity coefficient according to the wave direction.

to the corner, have been resolved on such a coarse grid. In the uniform region far behind the incident shock wave, a weak disturbance is observable. This is due to an initially discontinuous shock approaching to a continuous shock. This phenomenon can also exist in the results using some other schemes. It should be noted that although the Euler equations in the form of the conservative or primitive variables have no difference, the smoothness indicator (2.101) is different because the scalar variables used are either the conservative or the primitive variables. By comparing two results shown in Fig. 2.16ab, it is seen that the results are not sensitive to the variables used in the indicator. This is also ture for the 1-D flow by comparing Fig. 2.9 and Fig. 2.10.

2.5 Summary

In this chapter the TVD Lax-Wendroff scheme is reformulated via the smoothing flux. The conventional limiters then correspond to the nonlinear coefficient of smoothing. In the linear scalar case, two approaches are equivalent. In solving system of the conservation laws by using scalar limiters, the present approach is able to be split to two steps. In the first predictor step, local variables at interfaces are calculated by the Euler equations. Then the fluxes through the interfaces are determined by the local variables, plus the smoothing flux.

The extension from one-dimensional case to two dimensions follows the idea that a two-dimensional scheme should calculate a one-dimensional flow field with any orientation in the same way as the one-dimensional scheme do. It turns out that the artificial coefficient should be decomposed according to the direction of wave front. Unlike most schemes that employ the Riemann solver at interfaces, the



a.

b.

Figure 2.16. A shock wave with $M_s = 2$ diffracting over a 90° corner: $32 \times 32 \times 3$ cells, CFL=0.7, $\alpha = 1.2$. **a.** the result obtained by using the conservative variables; **b.** the result obtained by using the primitive variables.

present approach locally solves the exact multi-dimensional Euler equations there, consequently the Riemann solver is not required. The computational stencil for the scheme consists of nine nodes with three in each dimension, therefore it is compact. The scheme is an essentially central difference scheme, and can be as less diffusive as the first-order upwind scheme around a discontinuty in solving the scalar equation. The scheme will be formulated for any unstructured grid and applied to a variety of gas-dynamic problems in Chapter 4.

References

[1] Harten A. (1983) High resolution schemes for hyperbolic conservation laws, *J. of Comput. Phys.* 49 : 357-393.

[2] Shivamoggi BK (1985) Theoretical fluid dynamics, Martinus Nijhoff Publishers, Dordrecht.

[3] Godunov SK (1959) A difference scheme for numerical computation of discontinuous solutions of hydrodynamics equations, *Math. Sbornik*, 47 : 271-306.

[4] Lax PD, Wendroff B (1960) Systems of conservation laws, *Comm. Pure and Appl. Math.* 13 : 217-237.

[5] Davis SF (1984) A rotationally biased upwind difference scheme for the Euler equations, *J. of Comp. Phys.* 56 : 65-92.

[6] Roe PL (1984) Generalized fomulation of TVD Lax-Wendroff schemes, ICASE Report No. 84-53.

[7] Davis SF (1984) TVD finite difference schemes and artificial viscosity, ICASE Report No. 84-20.

[8] Yee HC, (1987) Construction of explicit and implicit symmetric TVD schemes and their applications, *J. of Comput. Phys.* 68 : 151-179.

[9] Sun M, Takayama K (1997) On the essentially central difference scheme. 8th National Shock Wave Symposium in Japan, 1997.

[10] Yee HC, (1987) Upwind and symmetric shock-capturing schemes, NASA TM 89464.

[11] Roe PL (1981) Approximate Riemann solvers, parameter vectors, and difference schemes, *J. of Comp. Phys.* 43 : 357-372.

Chapter 3

An essentially vectorizable data structure for some unstructured grids

3.1 Introduction

3.1.1 Structured and unstructured grids and data structures

A *structured* grid is one which can be mapped from the physical space to a computational space by one or more subdomains in which it appears as a rectangle. A structured grid usually allows a two-dimensional (i, j) or three dimensional (i, j, k) array to be employed for denoting the grid points with suitable definitions of the computational domain and interfaces between subdomains. Fig. 3.1 shows two examples of 2-D structured grids made of rectangles and triangulars. A most important characteristic of a structured grid is that an interior node point on it can uniquely determine its neighboring node points from its own index. For instance, four closest neighbors of node (i, j) on the rectangular grid are $(i, j + 1)$, $(i, j - 1)$, $(i + 1, j)$ and $(i - 1, j)$ as shown in Fig. 3.1a, while six closest neighbors of node (i, j) on the triangular grid are $(i - 1, j - 1)$, $(i, j - 1)$, $(i + 1, j)$, $(i, j + 1)$, $(i - 1, j + 1)$ and $(i - 1, j)$ as shown in Fig. 3.1b.

The generation of an initial body-fitted structured grid about arbitrary geometries is a difficult task. The *multiblock* strategy has been used for complex geometries. This concept is to divide the complex physical domain into simple subdomains, or

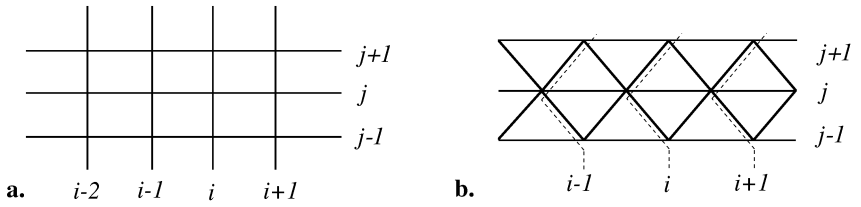


Figure 3.1. Illustration of structured grids: **a.** rectangular structured grid; **b.** triangular structured grid.

blocks, which can be easily covered by a structured grid. However, grid generation around complex configurations entails the need for a large number of blocks, and the problems of defining the blocks and their interfaces become increasing difficult as the number of blocks increases. To alleviate the difficulties in structured grid generation, the *unstructured* grid is generally adopted. A unstructured grid does not determine the connectivity information from indices, but locally stores it. Actually a multiblock grid is unstructured between blocks, but provides structured portions within blocks, it represents a compromise between a globally structured mesh and a locally defined unstructured mesh.

It should be noted that whether a grid is structured or not sometimes depends on the way of restoring the adjacency information of the grid, i.e., what *data structure* has been used. A structured node point is usually labeled by an index (i, j), and neighboring nodes are deduced from the index. Although the triangular mesh shown in Fig. 3.1b is able to be defined as a structured one, most researchers prefer to order the mesh in an unstructured way.

Two popular methods of restoring unstructured grids are the *cell-centered* and the *cell-vertex* approaches. In the cell-centered approach, the unstructured grid is stored according to its mesh cells, so the variables are stored at the centroids of the cells, whereas in the cell-vertex approach the grid is stored following grid nodes or vertices and the variables are stored at the vertices. Both approaches should include adjacency information, in some way by array

$$listnbr(i, inbr); \quad inbr = 1, 2, \cdots, maxnbr(i). \tag{3.1}$$

In the cell-vertex approach i represents the index of a node point or a vertex, $listnbr(i, inbr)$ stores the indices of its neighboring cells or vertices. $maxnbr(i)$

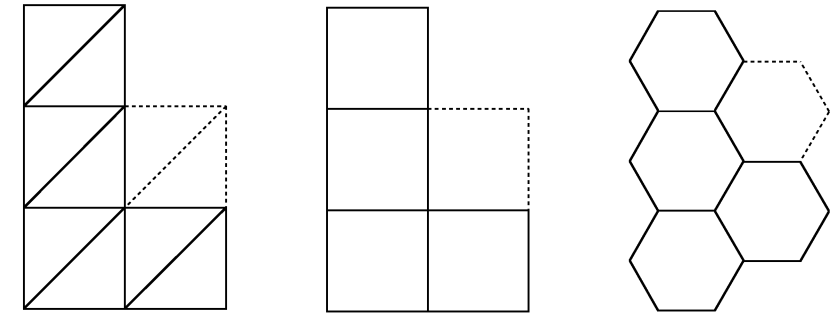


Figure 3.2. The ratio of the number of cells and vertices for different cells: **a.** triangular grid, **b.** quadrilateral grid, **c.** hexagonal grid. One new vertex introduces two triangular cells, or one quadrilateral, but one new hexagonal cell requires the addition of two vertices, as shown by dashed lines.

saves the number of i's neighbors. In the cell-centered approach i will represent the index of a cell, $listnbr(i, inbr)$ stores the indices of it neighbors, and $maxnbr(i)$ is a constant, three or four for a uniform triangular or quadrilateral grid respectively. The issue of cell-vertex vs. cell-centered data structure is still an open one. In 2-D case, the ratio of the number of cells to the vertices is approximately two for a triangular grid, nearly one for a quadrilateral grid, and half for a hexagonal grid, as shown in Fig. 3.2. As far as memory requirement is concerned, the cell-vertex structure is cheaper for the triangular grid, whereas the cell-centered one performs better for the hexagonal grid. In general, we are inclined to that both data structures require similar computational effort and memory requirements.

Another promising data structure is edge-based, or face-based in 3-D, which gathers information from all of neighbors of each edge, operates it on the edge, and then scatters it back to the neighbors of the edge. The neighbors of each edge are either two end nodes of the edge [1], or two cells of the edge on either side [2]. The edge-based structure has several major advantages. First the flux evaluation can be done by directly using the edge-based structure without any searching and logic. Another benefit is that the neighboring cells or nodes are of a fixed number, say 2 in two dimensions, which further removes the uncertainty in the data structure. In principle, an edge-based data structure is able to treat various control volumes

without difficulty.

3.1.2 Mesh adaptation

Most phenomena in nature are nonlinear systems which often contains many disparate length scales. In simulating them by computer, a numerical method should be able to adapt resolution to the regions with different length scales, otherwise computer resources will be largely wasted on regions where solutions do not require the maximum resolution. Mesh adaptation is just a way to distribute mesh cells in the interested regions so as to resolve a physical phenomenon economically.

Under the finite volume method, a grid can be adapted to the solution by two distinct ways h-refinement and r-refinement. R-refinement moves the existing nodes on a mesh to the regions of importance. In h-refinement or mesh enrichment or mesh embedding, the cells in the regions of interest are locally subdivided, and some other cells may be coarsened in the regions of less importance. Both adaptation methods are effective in providing higher accuracy for a given expenditure of computer time and memory than is possible with non-adapted methods. However, the accuracy improvement possible with r-refinement is limited by the number of nodes in the initial grid [3]. Several investigators have combined the r-refinement with h-refinement to compute complex flows. R-refinement is burdened with the difficulty in dealing with mesh quality issues and low efficiency of the global interpolation from the old to the new grid[1], so that a refinement method containing remeshing is more expensive than a simple h-refinement. Therefore only h-refinement is considered in this chapter. For a survey on r-refinement, the reader is referred to the review article by Hawken et al [4].

Any grid with h-refinement is actually unstructured because of the changing topology. The h-refinement methods appear in the literature are roughly divided to two groups, structured and unstructured h-refinement. The structured one keeps some structure in grid or in data structure, while the unstructured one completely uses unstructured grid and data structure.

One structured h-refinement is the so-called adaptive mesh refinement (AMR). In the approach, the refined regions are not an individual cell, but rather large rect-

[1]The interpolation may be avoided by taking into account the motion of grid, which is still not cheap.

angular regions with hundreds to tens of thousands of structured cells in each block [5]. As a result of this block-structured approach, it is possible to amortize the overheads of managing a complex unstructured data structure over a large regular region, and the calculation on the region can be easily vectorized [6]. A difficulty behind the AMR method is the use of a regular background mesh composed entirely of structured cells to allow a description of arbitrarily complex boundaries. On the other hand, much of the complexity of this method is concerned with the management of irregular data structures describing the placement of the finer grids within the coarser grids and the determination of the optimum placement of the finer grids. A serial implementation of the AMR algorithm for gas dynamics may comprise lengthy lines, sometimes over 27,000 exclusive of support libraries [6].

Another structured h-refinement is the adaptive Cartesian method [1], [7], [8], [9], [10]. In this method, the refined regions can be an individual cell, and the grid is represented on the Cartesian grid. The mesh generation is simply to cut the bodies in the flow out of a Cartesian grid, although boundary conditions for cut cells have to be carefully treated. Advantages of using Cartesian meshes for the Euler solvers, besides increased ease of mesh generation, include simpler flux formulations and fortuitous cancellation of truncation errors not occurring on less regular meshes. In addition, Cartesian mesh generation can be component-based [11]. However, the adaptive Cartesian method has two drawbacks as pointed out by Coirier and Powell [12]. First, viscous flux functions rely heavily upon grid smoothness and orthogonality to obtain accuracy and positivity. The non-smoothness of the cut cells has negative implications for the smoothness of aerodynamics parameters that rely on derivative quantities at walls, such as skin friction and heat transfer, and can also have a detrimental effect on convergence. Second, the isotropic Cartesian mesh fails to adapt mesh cells to anisotropic viscous phenomena at high Reynolds numbers, such as the boundary layer. Stretched cells are needed for the economic resolution of these flow phenomena.

3.1.3 Vector and parallel processing

A *vector* is a series of values, such as an array, on which instructions operate. When arithmetic, logical, or memory operations are applied to vectors, it is referred to as *vector processing*. Vector processing is made possible by the hardware feature,

among others, *pipelining*. Pipelining within the functional unit for an operation allows each step of the operation to pass its result to the next step after only one clock period on the Cray C90.

Parallel processing on the Cray C90 is achieved through splitting of loops containing no data dependencies among the processors of the architecture. Each processor receives vectors of smaller length which are then concurrently processed in standard vectorized mode. This distribution of workload is done dynamically at runtime. The Cray C90 is a shared memory multiple instruction multiple data (MIMD) architecture consisting of a few high performance vector processors sharing a common, global memory. All data is held within the shared memory effectively eliminating any kind of communication among the processors.

The vectorization of the steady flow solver on an unstructured grid has reached a mature state. It is accomplished by either sorting or coloring the elements into groups, so that within each group, no element accesses the same node. However the efficiency of vector processing of both sorting and coloring procedures is not clear. If an unstructured grid works without mesh adaptation, sorting and coloring are just conducted once before the computation of the flow solver. In this case, the issue of their efficiency is trivial. One may design a time-consuming sorting method to optimize any unstructured data structure for an object machine. However it is totally different for a grid with local mesh adaptation. The local adaptation changes the topology of the unstructured grid. The data structure after adaptation may completely change the order which is provided by sorting and coloring procedures. Consequently sorting and coloring have to be conducted with every mesh adaptation.

On the other hand, local mesh adaptation is associated with extremely complicated logic. The logic often contains data dependency which inhibits vectorization, so that the overheads of mesh adaptation becomes very large when the flow solver is well performed by vector and parallel processing. On the Cray C90, a non-vectorized mesh adaptation procedure may require the computer time 10 times as the flow solver in single processor mode, and around 50 times using eight processors [13]. In computation of steady flows, the low efficiency of mesh adaptation is somewhat tolerable because few adaptations are sufficient. But mesh adaptation has to be frequently performed in unsteady computations, so that one might even question

that whether we can really gain efficiency from such expensive mesh adaptation compared with a uniform grid.

Therefore, for general-purposed applications, one should propose a data structure for unstructured grids, which must make both the flow solver and mesh adaptation vectorizable.

3.1.4 The essentially vectorizable data structure

An *essentially vectorizable data structure* is a data structure under which

- a flow solver should be able to perform local and global operations by vector processing;
- mesh adaptation should be able to be performed by vector processing.

If a data structure satisfies the first condition, it is vectorizable; if it further satisfies the other condition, it is essentially vectorizable. The flow solver coupled with mesh adaptation under the essentially vectorizable data structure is able to compute unsteady and steady flows with very low overheads.

In this chapter, a class of the essentially vectorizable data structure is proposed. A flow solver under this novel data structure is able to locally obtain necessary connectivity information without any logic, and globally scan all elements without nested loops. Its high efficiency is shown by applying it to calculate 2-D compressible flows with shock waves in Chapter 4.

3.2 The cell-face data structure

An efficient data structure often depends on the numerical method which is chosen in the flow solver. For instance, a data structure is easily made cheaper in memory by just storing what is required by the solver. In this thesis, discussions are mostly focused on the finite volume method. The method computes the change of the conservative values of all control volumes by integrating their interface fluxes, which consists of two steps:

1. calculating fluxes at every interface,
2. gathering interface fluxes for every control volume.

One may write the first step in a pseudo-code format,

$$\begin{array}{l} \text{DO } all\ faces \\ \cdots\ flux\ evaluation \\ \text{END DO,} \end{array} \tag{3.2}$$

and the second step

$$\begin{array}{l} \text{DO } all\ control\ volumes \\ \cdots\ suming\ fluxes\ of\ faces \\ \text{END DO.} \end{array} \tag{3.3}$$

The operation on the DO loops scans all faces and control volumes of a grid, it is a *global* operation; while the operation inside the loops *locally* handles flux evaluation and summing fluxes. This section is devoted to a novel data structure which efficiently optimizes the global and local operations, thus greatly decreases the overheads of handling some unstructured grids.

3.2.1 Optimization of global operations

An intuitive data structure to underlie the two steps is using two lists of control volumes and faces with a bi-directional reference,

$$\text{control volume} \Longleftrightarrow \text{face.}$$

There are no reconcurrence appearing in the bi-directional reference, so that it naturally avoids sorting or coloring.

In vector processing, the number of loop iterations is preferably integer times as much as the machine vector length, say 128 for Cray C90. Let's consider the second step which sums the fluxes of a control volume,

$$\begin{array}{l} \text{DO } all\ control\ volumes \\ \quad \text{DO } all\ faces \\ \quad \cdots\ suming\ fluxes\ of\ faces \\ \quad \text{END DO} \\ \text{END DO.} \end{array} \tag{3.4}$$

Some data structures may completely adopt the nested loop (3.4), by inserting the first step (3.2) into the innermost loop [14]. The innermost loop is as long as the number of faces of a control volume. The numbers of faces of practical control volumes are much less than the machine length, say 128, so that (3.4) is poorly vectorizable. The easiest way to improve vectorization in this case is to remove the innermost loop by writing the iterations explicitly, or *inline expansion*. However

not all unstructured data structure can be expanded. Fig. 3.4a shows an arbitrarily quadrilateral grid. The nodes have changing numbers of their neighboring nodes, say from 2 to 5 in the figure. It is actually true for any arbitrarily unstructured grid that nodes have changing numbers of neighboring nodes. A cell-vertex data structure defines control volumes according to nodes, then control volumes will consists of changing numbers of faces. In this case the innermost loop cannot be expanded due to the unfixed number of faces. On the other hand, in a cell-centered data structure control volumes are often taken as mesh cells. If an unstructured grid is made of the same shaped cell, the face numbers of all cells are constant, 3 for triangular, 4 for quadrilateral and tetrahedron and 6 for hexahedron. The unstructured grid Fig. 3.4a is equivalent to a description of mesh cells with four faces as shown in Fig. 3.4b. For any constant number of faces, the innermost loop is replaced by directly summing the fluxes, for quadrilateral cells,

$$\begin{aligned} &\text{DO } \textit{all control volumes} \\ &\quad \cdots f = f_1 + f_2 + f_3 + f_4 \\ &\text{END DO.} \end{aligned} \tag{3.5}$$

Then the length of the loop to be vectorized becomes the total number of control volumes, which is usually much larger than any machine vector length. As far as the efficiency of vector processing is concerned, a control volume is preferably of constant number of faces. This is a topological advantage of the cell-centered structure to avoid the short innermost loop. The data structure using the bi-directional reference between control volumes and faces is then called a *cell-face* one.

In addition, the logic associated with local mesh adaptation is simper under the cell-face structure than the cell-vertex one. Fig. 3.3 illustrates adapting a quadrilateral cell. The mesh adaptation is realized by dividing or merging cells, then the operations handling the adaptation is able to be conducted on a single cell without influencing other cells, but always interacting with a few vertices. Under the cell-face data structure, the communication between cells are through their common face. The logic associated with adaptation on a cell face, such as whether split the face or not, is rather simple as one face has only two neighbors. It is not so for the cell-vertex structure. The logic of adaptation based on cell vertices are so complicated that it is extremely difficult to be vectorized.

The cell-face data structure not only takes the advantage of vector processing, but

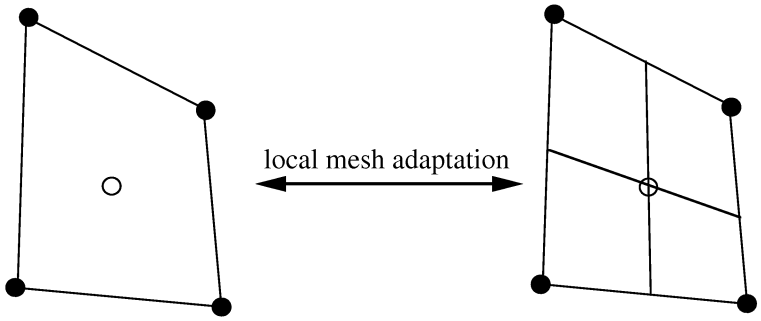


Figure 3.3. Illustration of local mesh adaptation of a cell. The adaptation is always dividing a cell or merging a few subcells, then a cell-centered data structure is preferred.

also is able to be implemented without user intervention on a shared-memory parallel machine. Both two steps (3.2) and (3.5) are long loops, a compiler may equally distribute the iterations to available processors without difficulty. This alleviate us from considering the machine dependent issues such as partitioning of the grid, or generating local data structures for each processor [13], [15].

3.2.2 Optimization of local operations

In the previous subsection, it has been shown that the cell-face data structure is efficient in handling global operations. However there are some uncertainties in local operation under the data structure. First, the direction of the flux through a face is uncertain since the outward normal are opposite for two neighboring cells of the face. Practically, the residual of fluxes in (3.5),

$$f = f_1 + f_2 + f_3 + f_4, \tag{3.6}$$

is not applicable for all control volumes due to this uncertainty. Second, if a face simply stores two neighboring cells, either cell cannot directly refer to each another. These two uncertainties of course can be removed in some way by introducing additional logic to it, however the additional logic contributes no improvement to a solution but increases the overheads of the cell-face data structure. This subsection shows a novel ordering method to remove these uncertainties without any additional overheads for unstructured quadrilaterals, tetrahedra and hexahedra. For simplicity only an unstructured mesh of the same shaped cells is analyzed.

Let's first revisit the famous *Königsberg bridge* problem on graph: when can

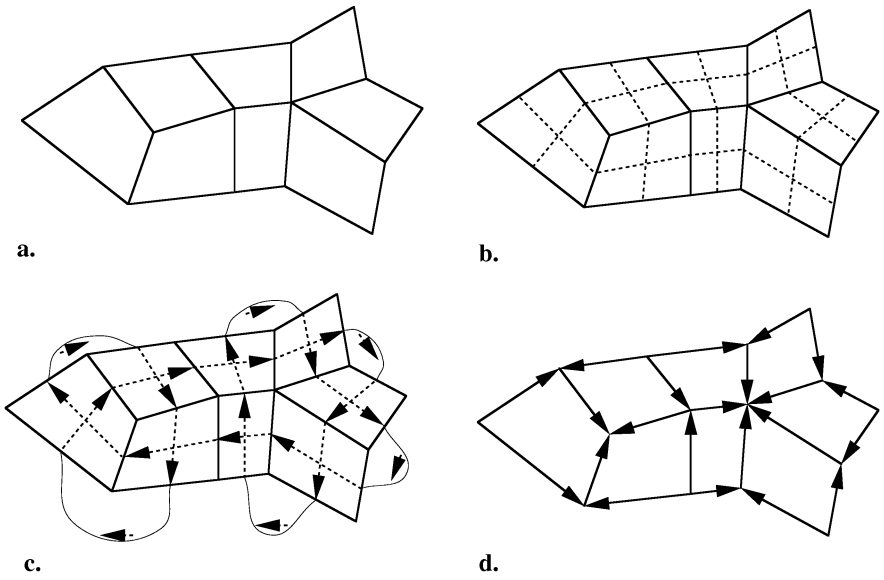


Figure 3.4. Illustration of generating a cell-face data structure for a quadrilateral grid: **a.** an arbitrarily unstructured quadrilateral grid; **b.** roads inside all quadrilateral cells; **c.** roads outside of the grid and a path to trace all cells shown by arrows; **d.** a definition of face directions according to the road direction.

one trace out a graph without travelling over any road twice (see [16])? The answer depends the number of intersection points such that the numbers of roads emanating from the intersection points are odd. If this number is greater than 2, the graph cannot be traced. If the number is 2, it can be traced, but only by starting at one of these, and ending at the other. If the number is 0, one can start anywhere, and will end at where one starts. To see that one can do it under these conditions, one way is to make any trip, starting at an odd vertex if there are two such, continuing until you get stuck. Then make another trip, but adding a side trip along untraveled roads, until you get back to the old route at the same point. Each trip becomes longer, until the whole is traced.

On an unstructured grid one may construct *roads* from any interior point of a cell to the cell faces, the point is then an intersection of roads. The number of roads that emanate from the point is equal to the face numbers. The dashed lines in Fig. 3.4b shows roads on a quadrilateral grid. If an unstructured mesh consists of the same shaped cell with even faces $2F$, the number of boundary faces is even[2].

[2]We show here that if an unstructured mesh consists of the same shaped cell with even faces $2F$, the number of boundary faces is even. Suppose a mesh has N total cells and I interior faces,

One may further join every two boundary faces and construct roads outside. All inside and outside roads constitute a graph, and the number of interaction points is the same as the total cell number. Clearly the number of roads emanating from the intersection points equals to the number of cell faces, then *if every cell is of even numbers of faces, the graph is able to be traced without travelling over any road twice.* Fig. 3.4c shows one path to trace the quadrilateral grid.

Theorem 1. If an unstructured mesh consists of the same shaped cells with even faces $2F$, the residual of outward fluxes of all cells can follow a definite way, such as

$$f = (f_1 + f_2 + \cdots + f_F) - (f_{F+1} + f_{F+2} + \cdots + f_{2F}).$$

Proof: One is just required to find a definite way for all cells to calculate the residual of outward fluxes. Since every cell is of even faces, there exists a path to trace all roads without any repeat. Without losing generality let the direction of the roads be along the path. When a road passes through a face, as shown in Fig. 3.5a, it must travel two cells. Then one may label the cell which it travels first *right cell*, the another *left cell*, and define the outward direction of any face by the left cell for flux evaluation. Because all roads are traced once, any cell must have outgoing roads and incoming roads of equal numbers F. One may label the faces which outgoing roads travel from 1 to F, others from $F+1$ to $2F$, then total outward flux becomes

$$f = (f_1 + f_2 + \cdots + f_F) - (f_{F+1} + f_{F+2} + \cdots + f_{2F}).$$

Fig. 3.5b shows an example of face labels for a hexagonal cell. This completes the proof of *Theorem 1.*

It should be noticed that once the sum of fluxes of cells follows the definite way, a cell can directly refer to its neighboring cells through their shared faces in the definite way. For example following the definition as shown in Fig. 3.5, the outside

---

and every cell has $2F$ faces. Since an interior face has two neighboring cells, it is countered twice. Then the boundary faces equals to

$$2F \times N - 2 \times I = 2(FN - I),$$

which is always even.

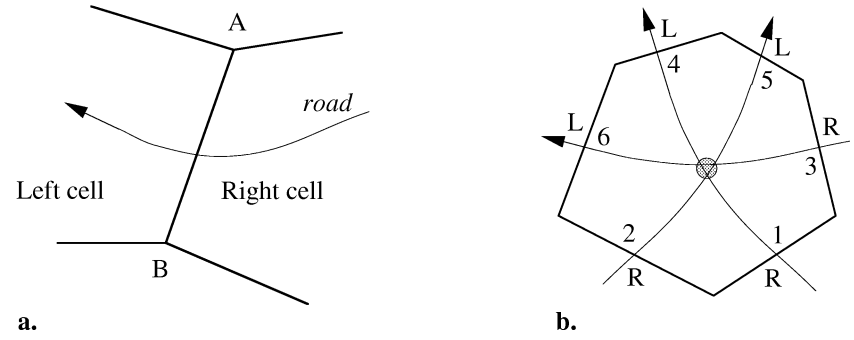


Figure 3.5. Ordering adjacent connectivity by the direction of the road: **a.** labeling two neighboring cells of a face *left cell* and *right cell*; **b.** numbering all faces of a cell.

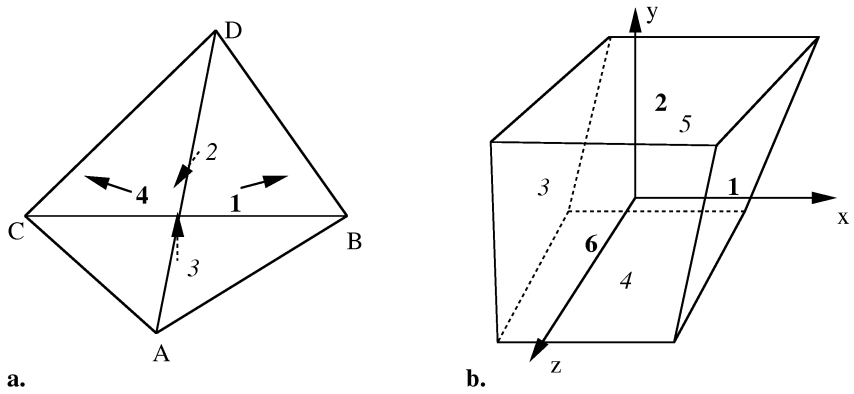


Figure 3.6. Possible order of three dimensional cells: **a.** for a tetrahedron; **b.** for a hexahedron.

neighboring cells of the cell in Fig. 3.5b are the right of faces 1, 2 and 3, and the left of faces 4,5 and 6.

In brief, by storing neighboring information with the order following a path which traces the whole graph of an unstructured grid, the uncertainties in local operation under the data structure are removed for any cells with an even number of faces. The conclusion is also valid for three dimensional cells. Examples of numbering tetrahedral and hexahedral cells are shown in Fig. 3.6.

In some numerical schemes which solve the conservation laws, a geometrical order of neighboring cells is required. For instance, applying Green's theorem in two dimensions to evaluate an averaged gradient of a control volume, one should then integrate along the boundary contour of the control volume. Fig. 3.7a shows a contour by the dashed line. Clearly, the integration procedure will greatly benefit from the geometrical order of boundary points, say anticlockwise ABCD$\cdots$. This requires a cell should preferably store its faces in the same order. On the other hand, in three dimensions the integration would be performed over the boundary surface. One portion of the surface is shown in Fig. 3.7b for a hexahedral cell. The hexahedral cell ought to store its faces in such a way that there is no ambiguity of which three of the six outside neighboring cells that form a valid portion of the boundary surface. However these geometrical orders are usually not consistent with the order which is defined following a path tracing all roads. The following part of this section will show that it is possible to construct a cell-face data structure which not only stores faces according the direction of roads but also preserves the necessary geometrical order for both quadrilateral and hexahedral grids.

First consider the quadrilateral grid. For convenience the direction of the roads is represented by the direction of cell faces, following that the face is pointing from road's left to road's right. The complicated road graph on Fig. 3.4c thus becomes Fig. 3.4d. Let's further impose that the roads inside a cell connects opposite faces only, then the fluxes of opposite faces must have opposite signs. Now the four faces of a quadrilateral can be numbered as that follows. One may first label a face with the positive sign 1, other faces are consecutively labeled anticlockwise, so that the residual of fluxes is

$$f = +f_1 \circ f_2 - f_3 \circ f_4,$$

where the signs of f_2 and f_4 are unknown. Since the signs of f_2 and f_4 are opposite, there are two possible cases, either

$$f = +f_1 + f_2 - f_3 - f_4$$

or

$$f = +f_1 - f_2 - f_3 + f_4.$$

In the first case one may change the label of face 1 to be 4, and number again other three faces anticlockwise, then

$$f = +f_4 + f_1 - f_2 - f_3 = +f_1 - f_2 - f_3 + f_4,$$

which is the same as the second case. Fig. 3.8 describes the consecutive labels of four faces which is also geometrically anticlockwise. Therefore, *there exists an anticlockwise sequence of four faces of any cell on an unstructured quadrilateral grid such that the residual of fluxes follows*

$$f = f_1 - f_2 - f_3 + f_4. \tag{3.7}$$

For the hexahedral grid, one may impose that the roads inside a cell should connect opposite faces only, then three outgoing roads always constitute a local system of "coordinates". Define three coordinates x, y and z according to the right-hand rule, which is not necessary to be the Cartesian coordinates. Label the faces which coordinates x, y and z go through 1, 2 and 6 respectively, then number their opposite surfaces 3, 4 and 5 in the same order. It is seen that once the system of "coordinates" is found, the labeling method is unique. The geometrical relation of neighboring cells is easily deduced from the unique labeling method, and the residual of fluxes follows

$$f = -f_1 - f_2 + f_3 + f_4 + f_5 - f_6.$$

3.3 Mesh adaptation under the cell-face data structure

In the previous section, it has been derived that a class of cell-face structure with strict definitions in summing fluxes and numbering faces. Under the data structure for quadrilateral and hexahedral grids, a flow solver may globally scan all control

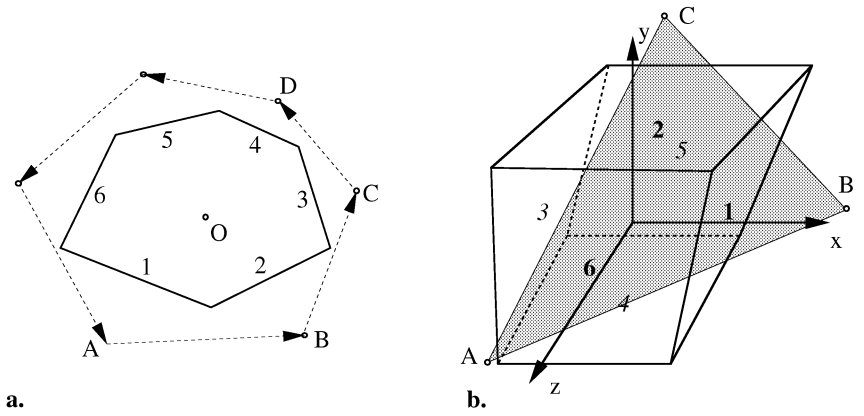


Figure 3.7. Illustration of orders of geometry required by some numerical methods: **a.** contour integral along outside neighboring cells, the order of these cells is useful; **b.** surface integral over a surface consisting of six neighboring cells, the information of which three cells that form a valid triangular surface is useful.

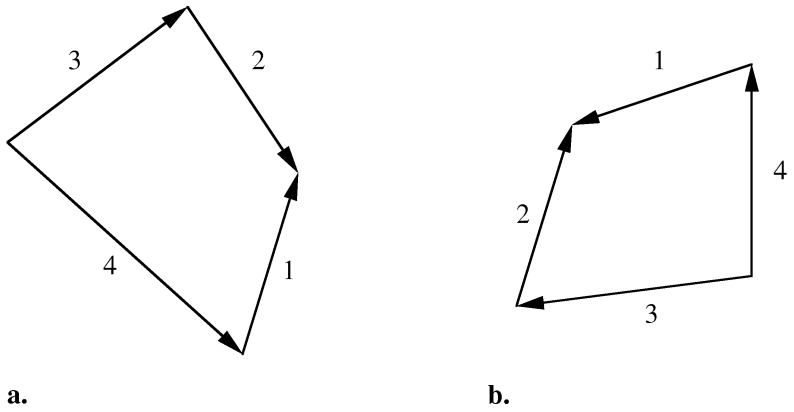


Figure 3.8. The order of face for quadrilaterals

volumes without nested loops, and locally obtain adjacent connectivity without any uncertainty. In this section we are going to show that the cell-face structure also permit vectorizable local mesh adaptation.

In local mesh adaptation, one or a few *hanging nodes* will appear on the face between two cells with different levels of refinement. Fig. 3.9 illustrates two handing nodes on quadrilateral and triangular grids. Treating the hanging nodes has to take into account the states of all neighboring cells of cell j, and furthermore treating the hanging node would possibly introduce some new hanging nodes. The procedure is so logically complicated that it is poorly vectorizable if not impossible. To be able to easily vectorize the adaptation procedure, it is decided in the early stage of the development of the present data structure that *any treatment for the hanging nodes is not considered*. On the cell-vertex data structure a hanging node represents a control volume, then carefully treating the node becomes essential for the flow solver. However a hanging node under the cell-centered data structure just implies that a few shorter faces replace their combined one. It does not change the control volume itself, so that whether treating the hanging nodes becomes trivial under the cell-centered data structure for the flow solver.

To preserve conservation the flux through a split face is equal to the sum of its subfaces' fluxes with the same sign. In Fig. 3.9a, the flux through AB equals to

$$f_{AB} = f_{AH} + f_{HB}.$$

Obviously it had better make the subfaces have the same direction as the original face so as to not have to consider the sign of fluxes. Fig. 3.10b demonstrates a direction definition of possible adaptation for the cell shown in Fig. 3.10a. It is straightforward to define the direction of internal faces, if two opposite faces are split equally. Fig. 3.12 illustrates a few possible adaptation strategies with their direction definitions. The adaptation for a hexahedral cell is similarly defined, an example is shown Fig. 3.11. The proposed adaptation strategies preserve the property that the residual of fluxes may follow a definite way no matter whether the faces are split or not.

It should be noticed that the mesh adaptation consists completely of local operations, such as defining directions of subfaces and locations of subcells. The op-

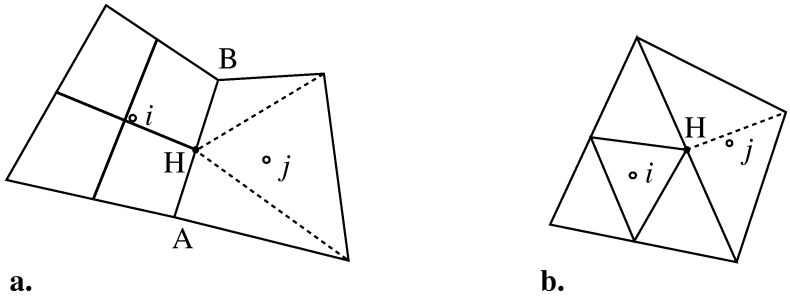


Figure 3.9. Illustration of hanging nodes: **a.** hanging node H on an adaptive quadrilateral grid; **b.** hanging node H on an adaptive triangular grid. Dashed lines indicate possible connections to remove the hanging nodes. The present cell-face data structure will leave the hanging nodes intact.

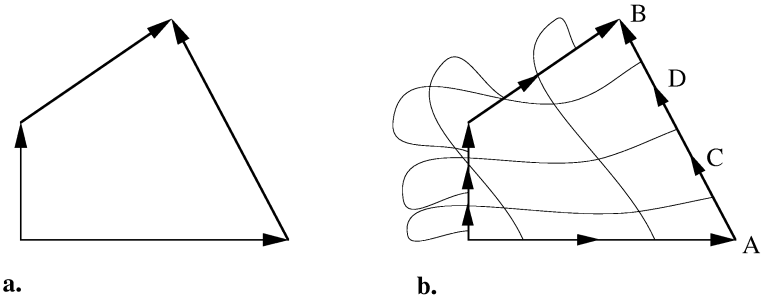


Figure 3.10. Illustration of the direction of the outside subfaces: **a.** face directions of a cell; **b.** directions of the outside subfaces. The directions of subfaces are the same as these of their unsplit faces.

erations are able to be determined from the original cell and its four faces. This procedure can be conducted by vector processing without difficulty. The data structure is thus essentially vectorizable.

3.4 Summary

A class of essentially vectorizable data structure for any unstructured quadrilateral and hexahedral grids is proposed in this chapter. The data structure is a bi-directional reference between cell and face with a strict ordering method. A flow solver using the finite volume method under the data structure is able to

1. perform local and global operations by vector processing;

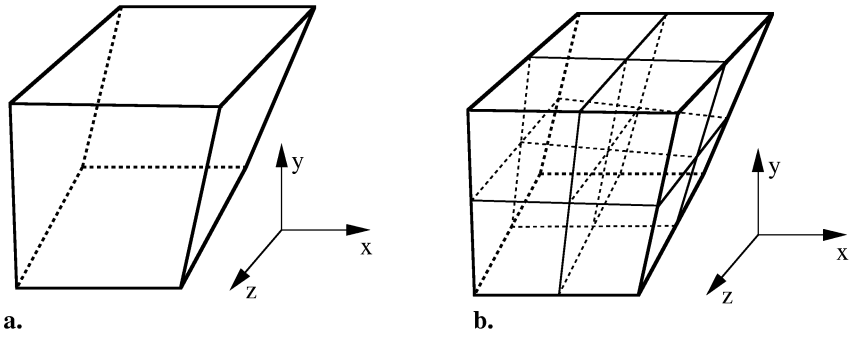


Figure 3.11. A vectorizable adaptation strategy for a hexahedral cell

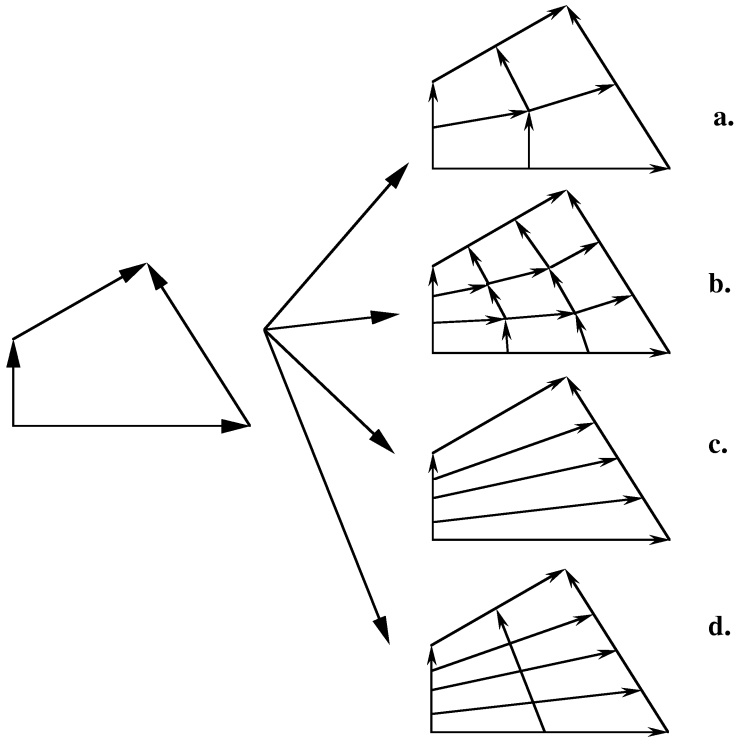


Figure 3.12. Illustration of four possible adaptation strategies under the cell-face data structure.

2. locally obtain necessary connectivity information without any logic;

3. globally scan all cells or control volumes without nested loops.

It is clear that the data structure with these properties is efficient not only on supercomputers but also on other platforms, such as personal computers. In addition the data structure may perform local mesh adaptation by vector processing. The data structure considerably decreases the overheads of handling unstructured grids and adapting grids, and thus it is high efficient for most applications besides unsteady and steady compressible flows with shock waves.

References

[1] Luo H, Baum JD, Löhner R (1994) Edge-based finite element scheme for the Euler equations, AIAA J. 32 : 1183-1190.

[2] van der Vegt JJW, van der Ven H (1998) Discontinuous Galerkin finite element method with anisotropic local grid refinement for inviscid compressible flows, *J. of Comp. Phys.* 141 : 46-77.

[3] Dannenhoffer JF (1991) A comparison of adaptive-grid redistribution and embedding for steady transonic flows, *Intl. J. for Numerical Methods in Engineering*, 32 : 653-663.

[4] Hawken DF, Gottlieb JJ, Hansen JS (1991) Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations, *J. of Comp. Phys.* 95 : 254-302.

[5] Berger MJ, Colella P (1989) Local adaptive mesh refinement for shock hydrodynamics, *J. of Comp. Phys.* 82 : 67-84.

[6] Colella P, Crutchfield *http://www.nersc.gov/research/CCSE.*

[7] Pember RB, Bell JB, Colella P, Crutchfield WY, Welcome ML (1995) An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, *J. of Comp. Phys.* 120 : 278-304.

[8] Young DP, Melvin RG, Bieterman MB (1991) A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics, *J. of Comput. Phys.* 92 : 1-66.

[9] Zeeuw DD, Powell KG (1993) An adaptively refined Cartesian mesh solver for the Euler equations, *J. of Comput. Phys.* 104 : 56-68.

[10] Akutsu K, Watanabe H, Honma H (1998) Numerical simulation for three dimensional shock waves discharged from open ends, 9th National Shock Wave Symposiums (in Japanese), 139-142.

[11] Aftosmis MJ, Berger MJ, Melton JE (1997) Robust and efficient Cartesian Mesh Generation for component-based geometry, AIAA 97-0196.

[12] Coirier WJ, Powell KG (1996) Solution-adaptive Cartesian cell approach for viscous and inviscid flows, AIAA J. 34 : 938-945.

[13] Domel ND (1996) Research in parallel algorithms and software for computational aerosciences, NAS-96-004, NASA Ames Research Center.

[14] Voinovich PA (1993) Two-dimensional locally adaptive unstructured unsteady Euler code, Advanced Technology Center, St. Petersburg, Russia (unpublished).

[15] Venkatakrishnan V, Simon H, Barth TJ (1991) A MIMD implementation of a parallel Euler solver for unstructured grids, RNR-91-024, NASA Ames Research Center.

[16] Fulton W (1995) Algebraic Topology, a first course, Springer-Verlag.

Chapter 4

Two-dimensional vectorizable adaptive solver (VAS2D)

4.1 Introduction

The unstructured meshes are usually made of triangles and quadrilaterals. Extensive studies have been conducted on unstructured meshes of triangles for either the finite volume method or finite element analysis (see Venkatakrishnan's review [1]). Since a triangular mesh has the number of edges roughly 50% over the quadrilateral counterpart, it is clear that the triangular mesh consumes more storage and more flux evaluations than the quadrilateral one. However it is not yet established whether the additional edges of the triangular mesh yields higher accuracy. Aftosmis et al [2] compared a variety of reconstruction schemes on both meshes, and concluded that on regular and stretched meshes the additional edges do not lead to any apparent accuracy advantage. The conclusion supports the method that removes unnecessary edges from boundary layer regions. The method makes use of a structured quadrilateral mesh in the near-wall regions and other regular portions of an unstructured triangular mesh (for example, [3]). If these structured quadrilaterals are unavoidable in the boundary layer, and quadrilaterals are more cheap and efficient without losing accuracy, a natural idea is that one may completely make use of unstructured quadrilateral meshes but generate a layer of quadrilateral mesh which is orthogonal to the wall surface in mesh generation procedure. Repeated adaptation on the layer of quadrilateral mesh automatically generates a structure-like mesh around the wall

surface. This idea finally leads to the work in this chapter. It should also be noted that the effort involved in generating a quadrilateral mesh of a certain fineness is much less than the effort required in generating triangles of similar fineness [4]. In addition, Chapter 3 has shown that the essentially vectorizable data structure exists for quadrilaterals but not for triangles. The data structure will greatly simplify the complicated logic especially in the adaptation procedure.

In this chapter, a two-dimensional unstructured vectorizable adaptive solver (VAS2D) is developed on quadrilateral grids. This general-purpose solver is able to simulate unsteady/steady, inviscid/viscid compressible flows. The VAS2D consists of three parts, pre- and post-processing, flow solver, and adaptation procedure. All parts are equally important, and demand considerable skills and resources.

The organization of these parts is shown in Fig. 4.1. In the pre-processing, a quadrilateral mesh generator (QuadMesh) is proposed for automatically generating quadrilateral grids for arbitrary geometry. In the flow solver the finite volume method is used to treat complicated shapes of control volumes. The novel central difference scheme with smoothing or artificial viscosity (see Chapter 2) is adopted. The fluxes through interface are determined by locally solving the two-dimensional Lax-Friedrichs scheme; neither the Riemann solver nor one-dimensional decomposition is required. The artificial coefficient of smoothing is controlled so that the scheme holds monotonicity property or satisfies the total variation diminishing (TVD) condition at least for the scalar case. The adaptation procedure refines flow regions with large truncation errors, and coarsens regions with small truncation errors. The truncation errors are measured by an error detector or sensor, which employs the ratio of the second derivative term to the first one of the Taylor series expansion of density. It turns out that the flow solver with the error sensor is nearly independent of flow conditions for a wide range of the Mach number. The cell-edge data structure discussed in Chapter 3 is applied to the flow solver and the adaptation procedure. Post-processing transforms data on the cell-edge data structure to the conventional vertex-centered or cell-centered structure, then flow visualization and data processing can be realized by using commercial softwares. The VAS2D provides a few subroutines to draw iso-contours of flow variables in PostScript format.

The chapter is organized as follows:

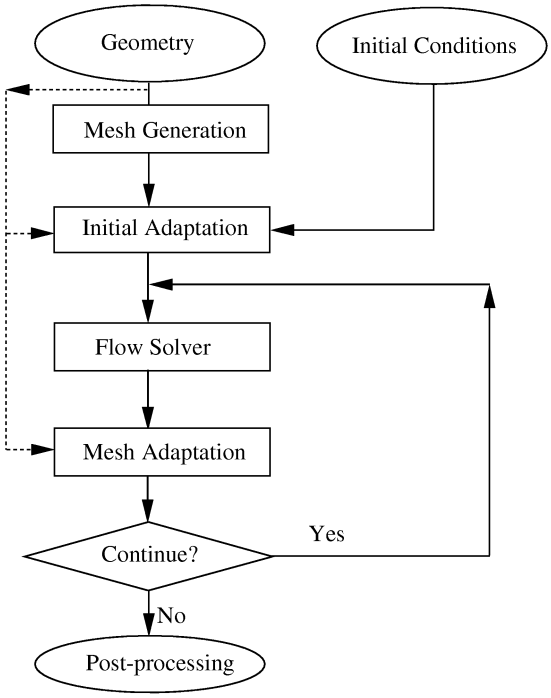


Figure 4.1. The overview of the algorithm VAS2D

- Section 4.2 gives a brief description of the automated mesh generation algorithm. A few mesh examples are shown. Some numerical results calculated on these meshes are demonstrated in Section 4.7;

- Section 4.3 introduces the cell-edge structure which is employed in the VAS2D. The data structure is used by both the adaptation procedure and the flow solver. Detailed implementation of the structure on the adaptation procedure is discussed in Section 4.4, and on the flow solver in Section 4.5.6;

- Section 4.4 is devoted to the adaptation procedure, including criteria for adaptation, strategy for adaptation, conservative variables at new cells, and finally efficiency of vector processing of the adaptation procedure;

- Section 4.5 presents the flow solver on quadrilateral grids. Although the methods in this section are described for cell-centered quadrilaterals, they can be applied to arbitrarily shaped control volumes without difficulty;

- Section 4.6 measures practical efficiency of vector and parallel processing of the

VAS2D. A quantitative comparison of the adaptation method and the uniform Cartesian grid is also shown. The comparison shows the VAS2D performs better than the uniform Cartesian grid by only two levels of refinement in both computer time and memory requirement;

- Section 4.7 contains a variety of numerical results obtained by the VAS2D. The results include shock diffraction, steady supersonic tunnel flows, unsteady shock / circular-cylinder interaction, shock / square-cylinder interaction, shock / boundary layer interaction, and shock moving in a nozzle.

4.2 Automated quadrilateral mesh generation

A locally adaptive mesh algorithm requires an initial background grid on which the adaptation can be conducted. Although there are some available algorithms to generate the grid, data structures in these algorithms are quite different from that used in the VAS2D. Furthermore, it is decided early in the development of the present method that a mesh should be suitable not only for inviscid flows but also for viscous ones, that is, high quality meshes may be automatically created in the boundary layer when the mesh is adapted by the VAS2D. This imposes more restrictions on the mesh generation procedure. In this section an algorithm is proposed to generate quadrilateral meshes for arbitrary geometry to complement the VAS2D solver. The technique used is similar to the *advancing front method* which is widely applied in triangular and quadrilateral mesh generation [5], [6], [7], [8]. Special attention is paid to generate high quality boundary cells.

4.2.1 Outline of the mesh generation algorithm

The automated quadrilateral mesh generation algorithm (QuadMesh) is programmed interactively for personal computers, since extensive computation is not necessary in 2-D mesh generation and personal computers often have a more flexible graphical interface than supercomputers.

An outline of the QuadMesh is summarized in Fig. 4.2. Any 2-D geometry can be described by any combination of line segments and/or arcs to some accuracy. Users are just required to input the data of these segments and arcs in a data file. The algorithm reads the data from the file, then it generates meshes automatically.

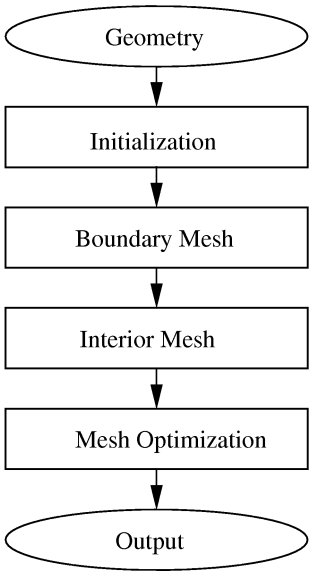
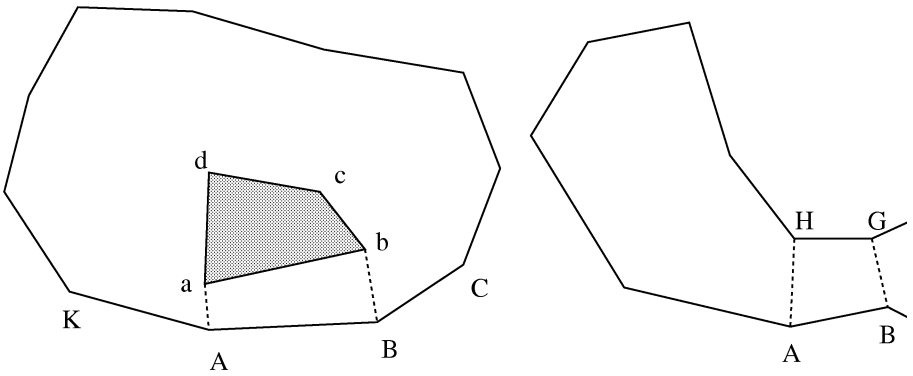


Figure 4.2. The overview of the quadrilateral mesh generation algorithm (QuadMesh)

In initialization, the QuadMesh creates one initial boundary front for a simply connected domain, and a few boundary fronts for a multiply-connected domain. In the code, every node on front(s) is linked by storing indices of its left and right neighboring nodes; a front thus becomes a chain of nodes. The QuadMesh then generates new mesh cells for the extremely sharp concave corners ($\leq 30^\circ$) if they exist. These sharp corners may cause some logic difficulties in generating interior nodes, such as new nodes are generated outside the domains.

After start advancing the boundary front(s), the QuadMesh first generates mesh cells along the boundary of the geometry. This layer of boundary mesh is well controlled so that it can be orthogonal or nearly orthogonal to the given geometry after optimization. The accuracy of a flow solver is enhanced by the adaptively refined meshes based on the orthogonal mesh. Most applications may benefit from this treatment because the accuracy at the boundary is of great value especially in viscous flows. After the boundary mesh cells are generated, the QuadMesh generates interior mesh cells by repeatedly advancing the updated front.

During advancing the front(s), new mesh cells are created on the front at sharp corners ($\leq 120^\circ$). If there are no sharp corners, a new cell is generated in a smooth region on the front. The new cell introduces new sharp corners. Then the procedure



a. **b.**

Figure 4.3. Illustration of joining-front **a.** Joining a multiply connected front; **b.** Joining a simply connected front. A new cell is created between front(s); the new cell changes the connectivity of front.

above is repeated until the mesh cells cover the whole domain of the given geometry. If some nodes on the front(s) are very close together in space, but not close in the chain of front nodes, they are joined together by creating a new mesh cell there. This joining-front procedure is illustrated in Fig. 4.3. Joining-front is unavoidable for a multiply connected domain, and it also possibly appears in the narrow region of a simply connected domain. Note that after joining a front, a simply connected domain becomes a multiply connected one, while a multiply connected domain may become simply connected.

In the advancing front procedure the quality of mesh is not taken into account, so the generated mesh should be optimized. Two optimization methods are used, one is to optimize the number of neighbors of the interior nodes, and another optimizes the nodes' locations by some elliptic iterations. By noticing that an interior node on any structured grid has four neighbors, the number of neighbors is optimized close to four wherever possible. For example, a node which has only two neighbors are removed; Kinney [9] summarized many other cases which are able to be improved. In the iterations, locations of the nodes that are close to a boundary are adjusted to be orthogonal to the boundary, and the nodes inside are averaged by their neighboring nodes. A few iterations (less than ten) are practically sufficient to obtain an acceptable mesh.

The generated mesh is further transformed to the data under the cell-edge data

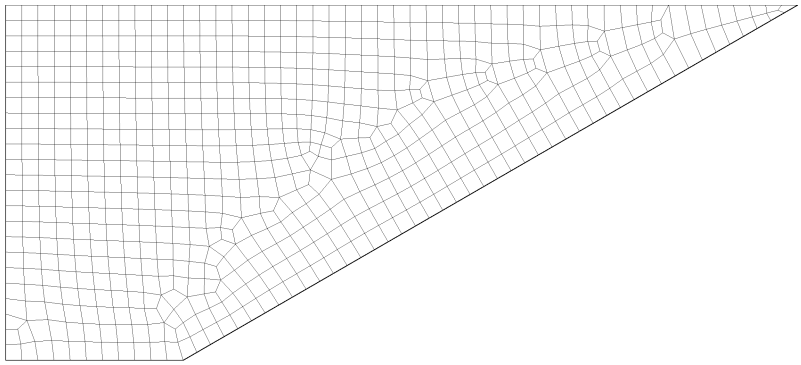


Figure 4.4. Unstructured quadrilateral mesh generated by the QuadMesh: a 30° wedge, 640 cells

structure (see Section 4.3), by following the method describe in Chapter 3. The output of the QuadMesh is a direct input for the VAS2D flow solver.

4.2.2 Grid examples

Some representative meshes are demonstrated in this subsection. All meshes are automatically generated by the QuadMesh. Fig. 4.4 shows the mesh of a 30° wedge. The cells are truly unstructured although a structured mesh can be obtained in this simple geometry. The interior cells near the 30° corner are specially generated in the initialization procedure. Fig. 4.5 is a mesh for simulating flow over a circular-cylinder. Note that the mesh cells close to the half circle are nearly perpendicular to the boundary surface. The computational domain for simulating flow in a nozzle is shown in Fig. 4.6. Fig. 4.7 shows the mesh generated for a multi-body geometry, in which a square-cylinder lies in a long tube. In four meshes, the QuadMesh generates high-quality regular mesh cells in large portions of the computational domains. More complicated geometry will be a combination of these simple geometry, and will not be shown here.

4.3 Data structure

A fundamental nature of the data structure is that every cell points to its four edges, and every edge points to its two neighboring cells. The basic information saved for a cell is its location and the indices of four edges. The four edges are uniquely ordered. In Fig. 4.8a, edge BC, CD, DA, AB are neighboring edges NE1, NE2, NE3, NE4 respectively. Every edge is defined as a directional segment as shown in Fig. 4.8b.

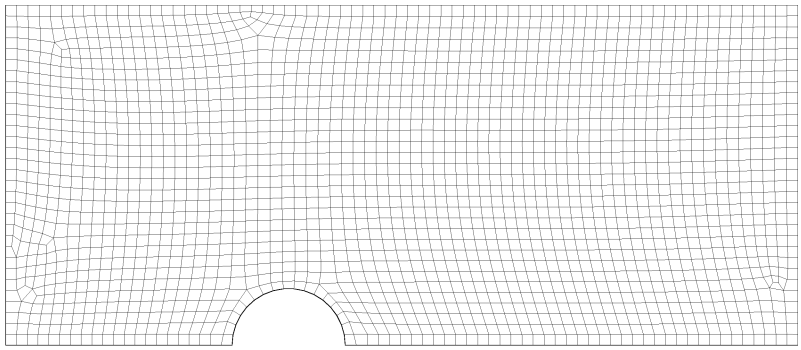


Figure 4.5. Unstructured quadrilateral mesh generated by the QuadMesh: a half-circle, 2112 cells

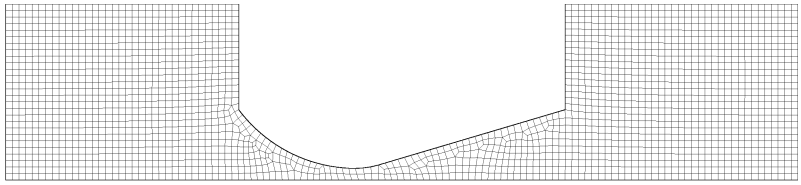


Figure 4.6. Unstructured quadrilateral mesh generated by the QuadMesh: a nozzle, 2169 cells

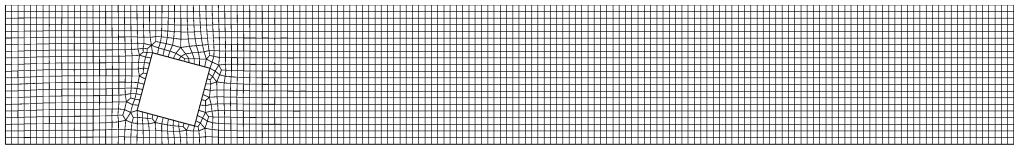


Figure 4.7. Unstructured quadrilateral mesh generated by the QuadMesh: an oblique square in a tube, 3316 cells

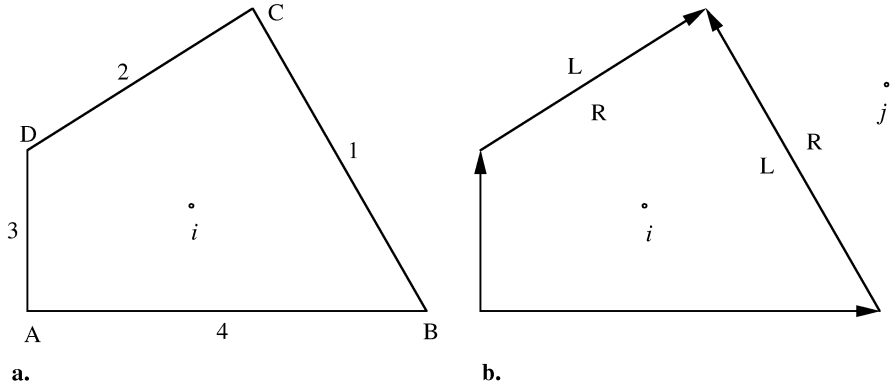


Figure 4.8. Cell-edge data structure: **a.** Cell information, a cell points to four edges; **b.** Edge information, an edge points to two neighboring cells.

The direction assures that the left of NE1, NE4 and the right of NE2, NE3 are the cell itself, while the right of NE1, NE4 and the left of NE2, NE3 corresponds to the right, below, above and left neighboring cells, respectively. In the cell, the conservative and primitive variables are located at the *centroid* of the quadrilateral. The centroid is often referred to as the *center of mass*. The locations of the centroid $\mathbf{r}_G$ of a quadrilateral are exactly obtained by dividing it into two triangles, say ΔABC and ΔACD in Fig. 4.8a. Denoting the centroids of two triangles $\mathbf{r}_{G1}$, $\mathbf{r}_{G2}$, and volumes S_1, S_2 respectively, one has

$$\mathbf{r}_G = \frac{\mathbf{r}_{G1}S_1 + \mathbf{r}_{G2}S_2}{S_1 + S_2}.$$

The centroid of a triangle is the algebraic average of the locations of three vertices, say triangle ΔABC,

$$\mathbf{r}_{G1} = \frac{1}{3}(\mathbf{r}_A + \mathbf{r}_B + \mathbf{r}_C).$$

The volume or area of a triangle is given by, say triangle ΔABC,

$$S_1 = \frac{1}{2}\mathbf{r}_{AB} \times \mathbf{r}_{AC}.$$

The basic information saved for an edge is the locations of its two ends and the indices of its left and right cells. A cell never directly points another cell but through their common edge. For instance, if cell i need the information of right cell j, then it should get the right index of its NE1 as shown in Fig. 4.8b. It is clear that one adjacent connectivity only needs two memory reads during computation, because of the unique definition of geometry.

The strict neighboring definition in the data structure reduces the complexity of logic in the unstructured adaptive solver without losing generality. It has been proven (see Chapter 3) that the definition exists for any mesh which consists of arbitrary quadrilaterals. We remarks that the data structure does not require much more memory compared with an unstructured data structure, it just stores the necessary topological information in the proper order.

Another advantage of the data structure which is better than other locally adaptive unstructured ones is that it can be naturally vectorized. It is generally known that data dependency prevents vectorization. Because a cell never directly points

to another cell but through their common edge, then *an operation on a cell can be organized to be independent of other cells.* This property simplifies the vectorization not only in solving conservation laws but also in refining and coarsening grids. Under the architecture of Cray C90, it turns out that the data structure even allows *parallel* execution on multiple CPUs by autotasking [1] without any user intervention. The efficiency of vector and parallel processing will be discussed in Section 4.6.

4.4 Adaptation procedure

4.4.1 Criteria for adaptation

There are two main reasons for using an adaptive mesh instead of uniformly refined one. First, physical phenomena have disparate length scales. The grid should be allowed to adapt to the solution to ensure that the regions with short length scales are not under-resolved and that the regions with long length scales are not over-resolved. Second, there are also abundant phenomena in the domain only some regions of which we are interested. Then the regions need to be studied at high resolution. For instance, in simulating flows past a blunt body, the vortex shedding in the wake behind the body may extend very far. Although main vortices in the long wake have the similar scales and should be resolved by similarly refined grids, it is much more economical just to refine the vortex shedding close to the body where we are interested. Another important example is geometry-based mesh refinement [11], in which meshes are adapted according to curvature of boundaries of computation domain. The geometry-based refinement mainly objects to improve resolution of complicated geometries. Note that the second reason for mesh adaptation is either problem-dependent or can be realized in mesh generation procedure, therefore only the adaptation criteria based on physical phenomena are discussed in this section.

Theoretically, an attractive criterion is that of equidistributing the numerical error[2]. In other words, fine cells should be distributed in any region where the error is large. In practice, such a concept only seems to work well in idealized, academic examples. There is substantial difficulty in defining appropriate measures

[1] Autotasking can be generally described as the automatic distribution of loop iterations to multiple processors, so the iterations can run on multiple processors concurrently, and it makes program run faster (wall-clock time) than it does without autotasking. (see [10])

[2] Another methodology emphasizes that all adaptive processes aim at a specified accuracy solution, then the adaptation criterion would be different (See [12]).

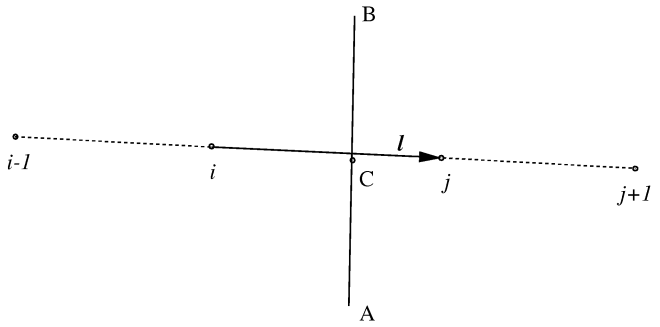


Figure 4.9. Sketch of an interface: interface AB and its two neighboring cells i and j; $i-1$ and $j+1$ are ghost cells.

of error, and in evaluating them numerically. To deal with this problem, heuristic measures are needed. Roe reviewed these measures in [13], and Zeeuw [14] compared characteristics of a few measures in steady flow calculations.

The current criterion for adaptation is based on the truncation error of the Taylor series expansion of density. The criterion is, for cell i

$$Refine = \text{Maximum } \varepsilon_{\mathrm{T}} \text{ in four neighboring edges} > \varepsilon_r,$$

$$Coarse = \text{Maximum } \varepsilon_{\mathrm{T}} \text{ in four neighboring edges} < \varepsilon_c,$$

where $Refine$ and $Coarse$ are logical flags which indicate whether the cell needs to be refined or coarsened. ε_r and ε_c are threshold values for refinement and coarsening. The truncation error indicator ε_T is given by the ratio of the second-order derivative term to the first-order one in Taylor expansion, so that

$$\varepsilon_T = \mathrm{Max}\left(\frac{|(\nabla_l\rho)_C - (\nabla_l\rho)_i|}{\alpha_f\rho_c/dl + |(\nabla_l\rho)_i|}, \frac{|(\nabla_l\rho)_C - (\nabla_l\rho)_j|}{\alpha_f\rho_c/dl + |(\nabla_l\rho)_j|}\right) \tag{4.1}$$

where ∇_l denotes the gradient along direction l as shown in Fig. 4.9. The locations of density appeared in (4.1) are shown in Fig. 4.9. The logic in (4.1) chooses the larger one in two ratios of the second-order term to the first-order one at nodes i and j along direction l. There are three parameters in the criterion. α_f is initially designed to prevent a zero denominator, and it can also filter extremely small flow variations. For example, if the amplitude of the variations in some regions is much less than $\alpha_f\rho_c$, then the truncation error indicators are also small. Consequently cells will not be refined there. The subscript f therefore denotes "filter". ρ_c/dl which follows α_f is used to make the indicator dimensionless. Three parameters are

slightly problem-dependent to achieve the most economical adaptation. If without description, $\varepsilon_r = 0.08$, $\varepsilon_c = 0.05$ and $\alpha_f = 0.03$ are chosen throughout the thesis.

Remark Consider the ratio of the second order term to the first order one in the Taylor series of function ξ, at node i,

$$|\frac{1/2\xi''\Delta x^2}{\xi'\Delta x}| = \frac{|\xi_{i+1} + \xi_{i-1} - 2\xi_i|}{|\xi_{i+1} - \xi_{i-1}|}.$$

Denote $\nabla\xi_i = \frac{\xi_{i+1}-\xi_{i-1}}{2\Delta x}$, $\nabla\xi_c = \frac{\xi_{i+1}-\xi_i}{\Delta x}$, the relation above becomes

$$\frac{|\nabla\xi_c - \nabla\xi_i|}{|\nabla\xi_i|}.$$

To avoid division by zero, adding a small value on the denominator, one has

$$\frac{|\nabla\xi_c - \nabla\xi_i|}{\alpha_f\xi_c/\Delta x + |\nabla\xi_i|}. \tag{4.2}$$

The ratio in the form of (4.2) is just that appears in (4.1). The flow solver that will be discussed in the following sections also uses this unstructured form to calculate the smoothness indicator. ¶

Remark The error detector in the present criterion compares the ratios of the second derivative term to the first one in four directions around one cell, it is powerful in detecting all important physical phenomena in compressible flows, such as shock wave, contact surface and vortex. Numerically it seems somewhat expensive. However it will be shown that the criterion is actually calculated together with solving the conservation laws, in which the similar smoothness indicator should be calculated. ¶

4.4.2 Strategy for adaptation

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

1. handling the inside of cells which are flagged to be refined or coarsened,

2. handling the edges of the flagged cells,

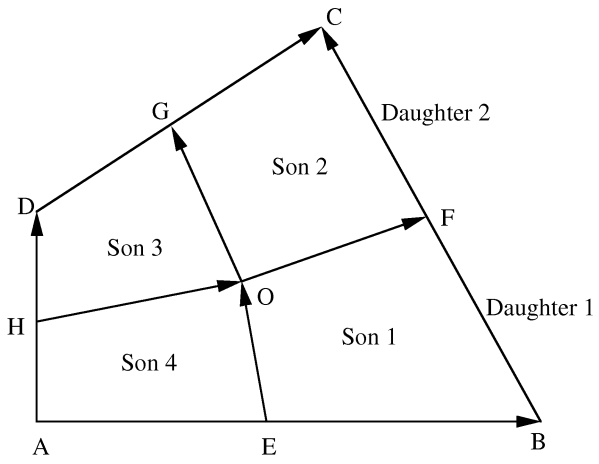


Figure 4.10. Refinement strategy: father cell ABCD is divided into four sons EBFO, OFCG, HOGD and AEOH.

3. arranging memory.

Step 1 is conducted based on cells, and updates all inside information, such as edge deletion and adding of finer cells, which is independent of other cells. Step 2, based on edges, renews every edge of refined or coarsened cells and its two neighboring cells, which may be done without influencing other edges. In this way the adaptation procedure has no data dependence and can be naturally vectorized.

In step 1, a cell to be refined is divided into four subcells or *sons* as shown in Fig. 4.10. If an edge is split into two subedges, we call the edge *a mother*, and the subedges *daughters*. The *refinement level* of sons is equal to their father cell's level plus one; the level of the coarsest cells is set to zero. To avoid unlimitedly adding fine cells around shock waves, either the maximum level of refinement or the minimum cell volume or both should be prescribed. When the level or the volume of a cell exceeds the given limit, further refinement is prohibited. For convenience, if the maximum level is three in one calculation, it is called 3-level refinement, and so on. Fig. 4.11 illustrates a 3-level refinement.

In the refinement procedure, it is required that no two neighboring cells differ by more than one refinement level. This rule prevents pathologically large volume ratios under certain circumstances. In Fig. 4.11 refining cell 5 is then not allowed, otherwise the levels of two sons of cell 5 and cell 2 differ by two.

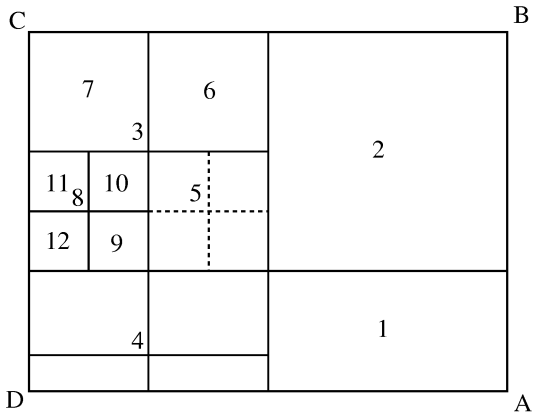


Figure 4.11. The levels of refinement. Suppose cell ABCD is one of the coarsest cells; the level of it is then 0. The refinement level of a son equals to that of his father plus 1. In the figure the level of cell 1, 2, 3 and 4 is 1; the level of cell 5, 6, 7 and 8 is 2, and so on. It is not allowed that the difference between levels of two neighboring cells is more than 1. Refining cell 5 is then not allowed.

In Fig. 4.10, the locations of new nodes E, F, G and H on mother edges are defined by the ratio of the lengths of two daughters,

$$\alpha_{bl} = \frac{\text{Length of daughter 2}}{\text{Length of daughter 1}}.$$

In most situations the ratio is set to 1, i.e. new nodes are the centers of mother edges. However when simulating viscous flows along a solid surface, stretched cells are more suitable in the boundary layer. The subscript bl denotes the ratio in the boundary layer. Only cells which are close to a solid boundary in integrating the Navier-Stokes equations are refined by following the parameter α_{bl}, and in the other cases the ratio is set to 1. The locations of the node inside can be optimized to get good shapes of the sons. In the present adaptation the locations of node O are given by a weighted average of the new nodes at outside edges,

$$\mathbf{r}_O = [(w_E\mathbf{r}_E + \mathbf{r}_G)/(1 + w_E) + (w_H\mathbf{r}_H + \mathbf{r}_F)/(1 + w_H)]/2.$$

The weights w_E and w_H are

$$w_E = (\alpha_{AD} + \alpha_{BC})/2, \quad w_H = (\alpha_{AB} + \alpha_{DC})/2,$$

where α_{AD} the ratio of the lengths of two daughters of edge AD, and so on.

Fig. 4.12 illustrates refining a cell along a viscous boundary. Suppose edge AB is the viscous boundary, nodes F and H are moved close to the boundary as shown

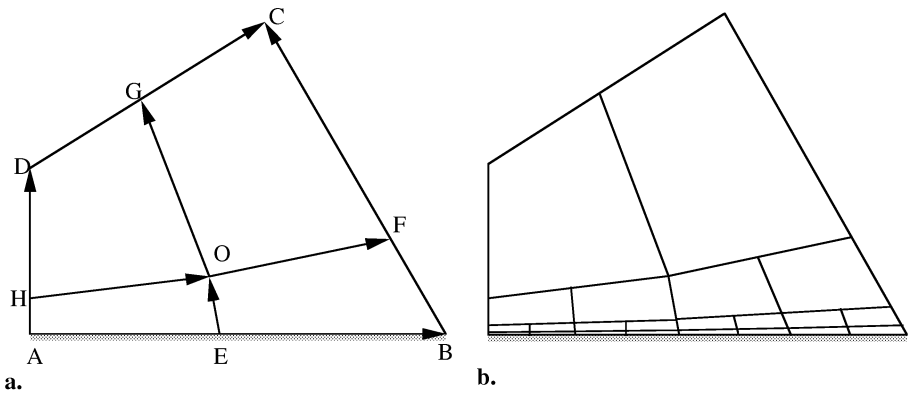


Figure 4.12. Directional refinement inside the boundary layer. A highly stretched mesh can be generated by repeatedly refining cells along the viscous boundary.

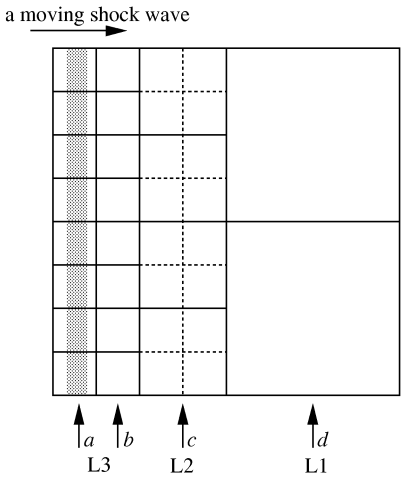


Figure 4.13. Illustration of pre-refinement. Cells in column c cannot be refined to capture the coming shock wave, because the level of refinement in column d is lower. The cells in column d has to be pre-refined.

in Fig. 4.12a. After conducting a few levels of refinement, stretched cells are formed along the boundary as shown in Fig. 4.12b.

In unsteady calculations, the rule that two neighboring cells differ by no more than one level may mismatch moving refined regions, say shock wave regions. An illustration is shown in Fig. 4.13. The refinement level of cells in column a and b are 3; these in column c and d are 2 and 1 respectively. A shock wave moving from left to right lies in the finest cells. The shock wave will arrive at column c in two time steps by using a high CFL number; then the cells in column c should be refined to be able to capture the shock wave. Since a shock wave can be resolved within two cells

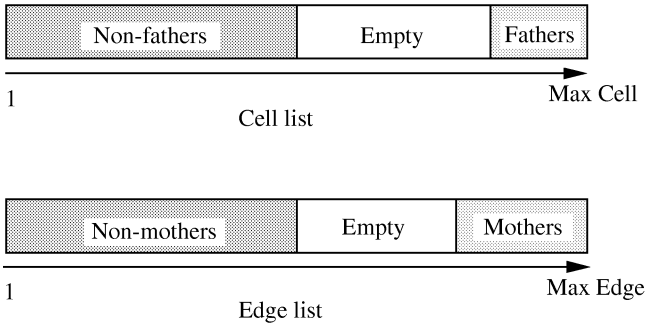


Figure 4.14. Memory organization

by a high-resolution scheme, cells in column d are not able to detect that the shock is coming from its closest neighbouring cells. The adaptation criterion then labels the cells in column d, $refine = false$, and thus column c cannot be refined because of the rule. Consequently the shock wave moves into the region with coarser cells and is smeared dramatically. This indeed happened in our early unsteady calculations. To combat this problem, *pre-refinement* is introduced. Once a cell cannot be refined due to the level difference between itself and its neighboring cells, the neighboring cells should be pre-refined. In the case shown in Fig. 4.13, the cells in column d are pre-refined, no matter what they are labeled.

In coarsening procedure only when all four sons are flagged $coarse = true$, are they deleted. In Fig. 4.10 all sons and inside edges are removed, and recovered to their father cell.

Step 1 and step 2 change the status of some cells and edges, for example from sons to fathers. Step 3 just cleans and arranges the memory, and divides cells and edges into two groups, father/non-father and mother/non-mother, according to their updated status. The non-mother edges are further divided into boundary/non-boundary edges. These classifications generate an efficient data structure for a flow solver. The memory arrangement is also vectorized. Actually, the adaptation subroutine without vectorization of memory arrangement would require about 200% additional computer time in Cray C90.

Fig. 4.14 illustrates memory strides of the edge list and the cell list after the adaptation procedure. All cell indices are saved in the cell list, while edge indices in the edge list. Both of them are constant strides, which are required for vectorization.

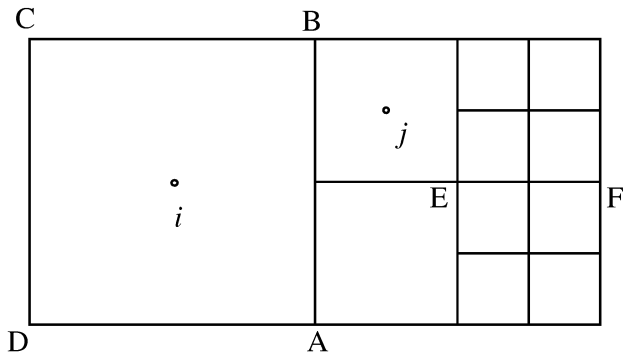


Figure 4.15. Illustration of the active mother edge. Active mother edge AB is referred by cell i, then information of it should be updated.

The lists are very efficient especially for a flow solver using a finite volume method. Non-fathers are just non-overlapping physical control volumes. These control volumes are numbered consecutively in the cell list. On the other hand, all non-mothers are all interfaces of the control volumes. Flux evaluations may be computed along the list of non-mother edges with a vector process.

Some mother edges need to be specially treated under the cell-edge data structure. These edges are split, but are still used by their neighboring cells; they are called *active* mothers as shown in Fig. 4.15. If only one side of an edge is refined, the edge will be referred by the cell on the another side, then the edge is an active mother. In Fig. 4.15, edge AB is an active mother. The error indicator ε_T of an active mother is set to the larger one of these of its two daughters. A numerical flux through an active mother edge is equal to the sum of the daughters. Note that if both neighboring cells are refined, no information of their shared mother edge, say EF in Fig. 4.15, is required in the flow solver.

4.4.3 The conservative variables at new cells

In both refinement and coarsening procedure, some new cells appear. The conservative or primitive variables at these cells have to be imposed by the information on the old mesh. In the globally remeshing adaptation method [15], [16], [17], [18], all information of new mesh are interpolated in some way from the old ones, which is very expensive. In the local adaptation method, only the new cells are required to be calculated. In practice the number of the new cells is a very small portion of

total cells, usually less than 5% for unsteady flows and much less for steady flows.

In refinement procedure, the conservative variables of new sons are interpolated from these of their father. Suppose a father i and one of its sons j, as shown in Fig. 4.16a, the interpolation reads

$$\xi_j = \xi_i + \nabla_i^{n-1}\xi \Delta \mathbf{r}_{ji}, \tag{4.3}$$

where ξ represents a conservative variable. Because the gradient at last time step has been evaluated in solving the conservation laws[3], it is directly used for the sake of efficiency, which introduces only a second-order error compared with using the gradient of the new step. In coarsening procedure, flow variables of a coarsened father cell are the volume-weighted average of these of its "dead" sons,

$$\xi_i = \frac{1}{\Delta\Omega_i}\sum \Delta\Omega_j \xi_j, \tag{4.4}$$

where $\sum$ sums four sons.

Remark The interpolation and the weighted average for new cells in the adaptation procedure are conservative. The conservative property of the volume-weighted average (4.4) is very straightforward,

$$\Delta\Omega_i \xi_i = \sum \Delta\Omega_j \xi_j.$$

However it is not so obvious for the interpolation (4.3). The sum of new conservative quantity inside four sons is

$$\sum \xi_j \Delta\Omega_j = \sum (\xi_i + \nabla_i^{n-1}\xi \Delta \mathbf{r}_{ji})\Delta\Omega_j = \xi_i \sum \Delta\Omega_j + \nabla_i^{n-1}\xi \sum \Delta \mathbf{r}_{ji}\Delta\Omega_j.$$

Note that node i is the centroid of the father cell and j are those of the sons, so

$$\sum \Delta \mathbf{r}_{ji}\Omega_j = \sum \int_{\Delta\Omega_j} (\mathbf{r} - \mathbf{r}_i) d\Omega = \int_{\Delta\Omega_i} (\mathbf{r} - \mathbf{r}_i) d\Omega = 0,$$

then

$$\sum \xi_j \Delta\Omega_j = \xi_i \Delta\Omega_i.$$

Clearly locating the solution at the centroid of a control volume is indispensable to preserve conservation in the interpolation. ¶

[3]In the VAS2D, because only the gradients of primitive variables are evaluated in solving the conservation laws, the gradients of conservative variables in the cells to be refined are calculated from these of primitive variables.

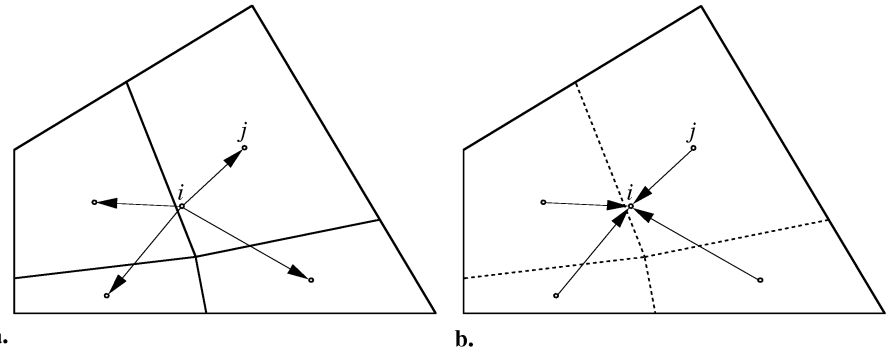


Figure 4.16. Flow variables at new cells. **a**. in refinement procedure, conservative variables of new sons are interpolated from these of their father; **b**. in coarsening procedure, conservative variables of a father cell are volume-weighted average of these of its "dead" sons. Both the interpolation and the average preserve conservation.

4.4.4 Efficiency of vector processing of the adaptation

The adaptation subroutine includes 32 loops and about 1,100 Fortran lines. All loops are vectorized. Its practical vectorization efficiency is tested by refining one cell from level 0 to level 6 (4096 cells), then coarsening the refined cells from level 6 to level 0. Fig. 4.17 shows the average CPU time normalized by 10,000 cells. No flow solver and data interpolation are computed in this test. The CPU time is decreased with increasing total cell numbers. When the total cells are over 1,000, the speedup is about 30. This indicates that the adaptation procedure is well vectorized.

4.5 Unstructured flow solver

4.5.1 The two-dimensional conservation laws

Consider the conservative vector quantity per unit volume $\mathbf{U}$, in an arbitrary control volume $\delta\Omega$, fixed in space and bounded by a closed surface $\delta\mathbf{S}$ with the outward vector $d\mathbf{S}$ (see Fig. 4.18). The conservative vector quantity are namely mass, momentum, and energy. The local intensity of $\mathbf{U}$ varies through the effect of *fluxes* $\mathbf{F}$, which express the contributions from the surrounding points to the local value and through sources. These sources can be divided into volume and surface sources, however only surface sources $\mathbf{Q}_s$ are considered in this thesis. The conservation laws are expressed in stating that the variation per unit time of the quantity $\mathbf{U}$ within the control volume $\delta\Omega$ should be equal to the net contribution from the incoming

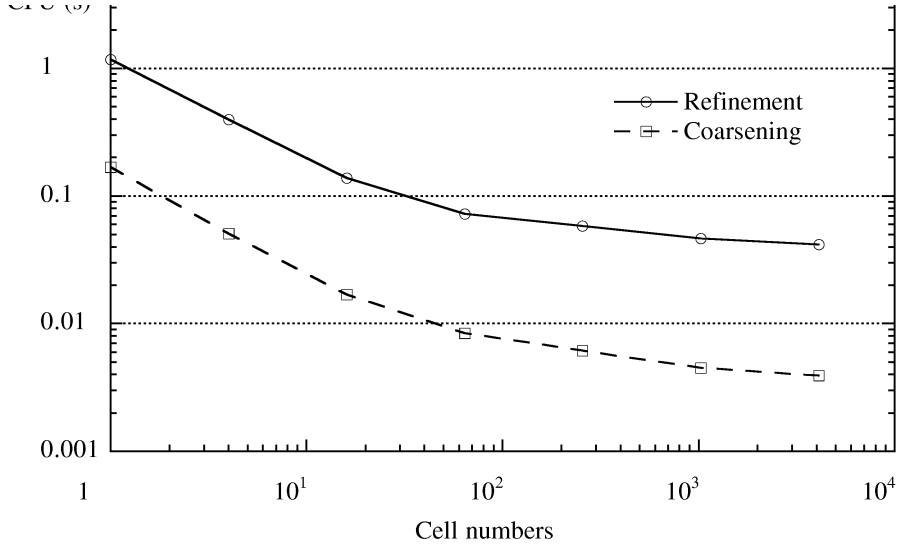


Figure 4.17. Vector efficiency of refinement and coarsening procedure

fluxes through the surface **S** and the surface sources,

$$\frac{\partial}{\partial t}\int_{\delta\Omega}\mathbf{U}d\Omega = -\int_{\delta\mathbf{S}}\mathbf{F}\cdot d\mathbf{S} + \int_{\delta\mathbf{S}}\mathbf{Q}_s\cdot d\mathbf{S}. \tag{4.5}$$

The two-dimensional Navier-Stokes equations in the form of (4.5) read

$$\begin{gathered}\mathbf{U} = \left\{\begin{array}{c}\rho\\ \rho u\\ \rho v\\ \rho E\end{array}\right\},\ \mathbf{F} = \left\{\begin{array}{cc}\rho u & \rho v\\ \rho u^2 & \rho uv\\ \rho uv & \rho v^2\\ \rho Eu & \rho Ev\end{array}\right\},\\ \mathbf{Q}_s = \left\{\begin{array}{cc}0 & 0\\ -p+\tau_{xx} & \tau_{yx}\\ \tau_{xy} & -p+\tau_{yy}\\ -pu+\tau_{xx}u+\tau_{xy}v+kT_x & -pv+\tau_{yx}u+\tau_{yy}v+kT_y\end{array}\right\},\end{gathered} \tag{4.6}$$

where total energy ρE is the sum of its internal energy and its kinetic energy

$$\rho E = \rho\int c_v dT + \frac{1}{2}\rho(u^2+v^2), \tag{4.7}$$

and for a Newtonian viscous fluid the viscous shear stress tensor is equal to

$$\tau_{ij} = \mu[(\partial_i v_j + \partial_j v_i) - \frac{2}{3}(\nabla\cdot\mathbf{v})\delta_{ij}], \tag{4.8}$$

or explicitly

$$\begin{aligned}\tau_{xx} &= \tfrac{2}{3}\mu(2u_x - v_y)\\ \tau_{xy} &= \tau_{yx} = \mu(u_y + v_x)\\ \tau_{yy} &= \tfrac{2}{3}\mu(2v_y - u_x)\end{aligned} \tag{4.9}$$

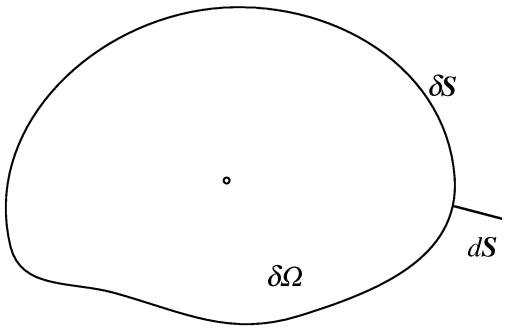


Figure 4.18. General form of the conservation laws

Velocity $\mathbf{v} = (u, v)$ is used in describing the conservation laws, however $\mathbf{v} = (v_1, v_2)$ is temporally adopted for the stress tensor (4.8) for the sake of simplicity. The conservation laws should be supplemented by the constitutive laws and by the definition of the coefficients of dynamic viscosity μ and thermal conductivity k. The state equation and the ratio of the specific heat coefficient under constant pressure, c_p, to the coefficient under constant volume, c_v, are taken as

$$p = p(\rho, T),$$

$$\gamma = \frac{c_p}{c_v}.$$

In many instances a compressible gas can be considered as a perfect gas, even if viscous effects are taken into account, and the equation of state is written as

$$p = \rho R T, \tag{4.10}$$

where R is the universal gas constant divided by the molecular mass of the gas. The perfect-gas law is accurate to $\pm 10\%$ in the range $1.8 \leq T/T_{crit} \leq 15$ (see [19]), where T_{crit} is the temperature at the critical point. For air, the range is 240K$\sim$2000K. As a direct consequence of the perfect-gas law the specific heats follows

$$c_p(T) = \frac{\gamma(T) R}{\gamma(T) - 1}$$

$$c_v(T) = \frac{R}{\gamma(T) - 1}$$

Table 4.1. Viscosity and thermal-conductivity parameters in Sutherland law for gases, $T_c = 273K$. (Transformed from White [19])

Gas	$\mu_c,\ 10^{-6}kg/(sm)$	$S_\mu,\ K$	$k_c,\ J/(smK)$	$S_k,\ K$
Air	17.16	111	0.0241	194
Ar	21.25	144	0.0163	150
N_2	16.63	107	0.0242	167
O_2	19.19	139	0.0245	222

In reality, the specific-heat ratio γ tends to decrease with temperture, but the approximation $\gamma = constant$ is accurate over fairly wide temperature variations. Under the approximation $\gamma = constant$, the total energy (4.7) becomes

$$\rho E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2). \tag{4.11}$$

The coefficients of dynamic viscosity μ and thermal conductivity k are functions of temperature T. Following Sutherland's formula they are taken as, with temperature in Kelvin (see [19]),

$$\mu = \mu_c (T/T_c)^{3/2} \frac{T_c + S_\mu}{T + S_\mu}, \tag{4.12}$$

$$k = k_c (T/T_c)^{3/2} \frac{T_c + S_k}{T + S_k}, \tag{4.13}$$

where T_c, μ_c and K_c are reference values, and S_μ, S_k are the Sutherland constants, which are characteristics of the gas. These values for some common gases are given in the table Table 4.1.

The Navier-Stokes equations are non-dimensionalized by defining some characteristic constants. We choose the characteristic density ρ_0; length L_0; temperature T_0; velocity $U_0 = (RT_0)^{1/2}$; coefficient of dynamic viscosity $\mu_0 = \mu(T_0)$; coefficient of thermal conductivity $k_0 = k(T_0)$ and the specific heat coefficient under constant pressure $c_{p_0} = \gamma R/(\gamma - 1)$. The combination of these values gives two fundamental parameters: *Reynolds number*

$$Re = \frac{\rho_0 U_0 L_0}{\mu_0} \tag{4.14}$$

and *Prandtl number*

$$Pr = \frac{c_{p0}\mu_0}{k_0}. \tag{4.15}$$

The dimensionless conservation laws are the same as the dimensional ones, but the coefficients are changed to

$$\mu = \frac{1}{Re} T^{3/2} \frac{1 + S_\mu/T_0}{T + S_\mu/T_0}, \tag{4.16}$$

$$k = \frac{\gamma}{\gamma - 1} \frac{1}{RePr} T^{3/2} \frac{1 + S_k/T_0}{T + S_k/T_0}, \tag{4.17}$$

with the dimensionless equaiton of state

$$p = \rho T \tag{4.18}$$

and the dimensionless sound speed

$$c = (\gamma T)^{1/2}. \tag{4.19}$$

The Euler equations are obtained by setting $\mu = 0$ and $k = 0$ in the Navier-Stokes equations.

4.5.2 Finite volume method

The finite volume method solves the conservation laws by directly applying them to every non-overlapping discrete volume the summation of which covers the whole computational domain. The conservation laws (4.5) are written for a discrete volume,

$$\frac{\partial}{\partial t} \int_{\Delta\Omega} \mathbf{U} d\Omega = \int_{\Delta\mathbf{S}} (\mathbf{Q}_s - \mathbf{F}) \cdot d\mathbf{S}. \tag{4.20}$$

Equation (4.20) is approximated by numerically evaluating the two integrals, then one has, for volume i,

$$\frac{\partial}{\partial t} (\mathbf{U}_i \Delta\Omega_i) = - \sum_{\text{faces}} \hat{\mathbf{F}}_k, \tag{4.21}$$

where $\hat{\mathbf{F}}_k$ is the vector of outward fluxes through face k. (4.21) states that the change of the conservative values of volume i is determined by summing its fluxes through all faces. This is the general formulation of the finite volume method. The shape of volume $\Delta\Omega_i$ is arbitrary. Modifying the shape and the location of the control volumes associated with a domain gives considerable flexibility to the finite volume method. In addition, direct discretization of the integral form of the conservation laws guarantees that the conservative quantities, namely mass, momentum and energy, will preserve conservation at the discrete level.

Remark It is seen that (4.21) numerically integrates (4.20), thus introduces the truncation errors. Let's first consider the error in evaluating the integral on the left hand of (4.20),

$$\int_{\Delta\Omega} \mathbf{U} d\Omega = \int_{\Delta\Omega} [\mathbf{U}_c + (\nabla \mathbf{U})_c \cdot (\mathbf{r} - \mathbf{r}_c)] d\Omega = \mathbf{U}_c \Delta\Omega + O(\Delta\Omega^2) \tag{4.22}$$

where the subscript c denotes at the centroid of the control volume $\Delta\Omega$. It is clear that if the variables are located at the centroid, the integration just introduces the error of $O(\Delta\Omega^2)$, then it is second order accurate. However if they are located at point i other than the centroid, a first-order error appears,

$$\int_{\Delta\Omega} \mathbf{U} d\Omega = \mathbf{U}_c \Delta\Omega + O(\Delta\Omega^2) = \mathbf{U}_i \Delta\Omega - (\nabla U)_c \cdot \mathbf{r_{ci}} \Delta\Omega + O(\Delta\Omega^2). \tag{4.23}$$

Since the error in the integration is of second order or worse, and since higher order integration becomes extremely complicated, a second-order scheme seems more appropriate on the unstructured meshes. ¶

The simplest method to calculate the integral on the right hand of (4.20) is the *trapezoidal rule*. The rule replaces the flux function by a linear one on each face, then the flux approximates to the value at the center of the face C,

$$\hat{\mathbf{F}}_k = -(\mathbf{Q}_s - \mathbf{F})_C \cdot \Delta\mathbf{S}. \tag{4.24}$$

Assuming every value at the centers is exactly known, one may get the error introduced by the trapezoidal rule is $O(\Delta\Omega^{1.5})$ (see reference [20]). However if the value at the centers of faces is unknown, it is usually linearly interpolated from point i, then additional error is also $O(\Delta\Omega^{1.5})$. Clearly for an arbitrary control volume, the finite volume method is not better than first order even with the linear reconstruction. For the scheme to be second-order accurate, either the first-order errors are cancelled by using a mesh with high quality, or high order reconstruction is required.

In the VAS2D, any domain is represented by quadrilaterals. The control volume is naturally taken to be a quadrilateral. The quadrilateral mesh cells such as these shown from Fig. 4.4 to Fig. 4.7 are all discrete control volumes. Applying (4.21) to a quadrilateral volume i, we have

$$\frac{\partial}{\partial t}(\mathbf{U}_i \Delta\Omega_i) = -\sum_{k=1}^{4} \hat{\mathbf{F}}_k, \tag{4.25}$$

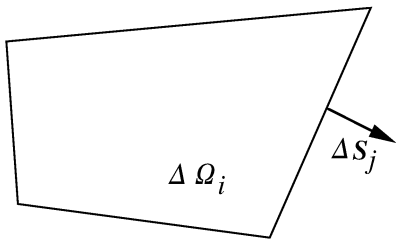


Figure 4.19. Control volume

where $\Delta\Omega_i$ denotes the volume of the cell i. The sketch of the control volume is shown in Fig. 4.19.

Following the two-step smoothing Lax-Wendroff scheme in Chapter 2, (4.25) is further written as

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i}{\Delta t}\Delta\Omega_i = -\sum_{k=1}^{4} \hat{\mathbf{F}}_k^{n+1/2},$$

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i - \frac{\Delta t}{\Delta\Omega_i}\sum_{k=1}^{4} \hat{\mathbf{F}}_k^{n+1/2}, \tag{4.26}$$

where $\hat{\mathbf{F}}_k^{n+1/2}$ are fluxes obtained by the predictor step, and located at the center of the interface or the edge. Since the variables are unknown at the center, they are interpolated from the centroid. The predictor step locally solves the modified two-dimensional Lax-Friedrichs scheme at the interface. Both the interpolation and the predictor step necessitate estimation of the gradients at every cell. The gradient evaluation will be discussed in Section 4.5.3, and the predictor step and flux evaluation in Section 4.5.4

4.5.3 Least square method

There are two methods for estimating the solution gradients within each control volume. One applys the Green's theorem to provide an evaluation of the gradients (see, for example, [2]). Another is least square method. The former will not be considered in the thesis. Least square method is sometimes referred to as the minimum energy reconstruction. It simply consists of fitting a linear function to the values of the neighboring nodes using the least-squares technique.

Suppose we are seeking the gradient $\nabla\xi = (\xi_x, \xi_y, \xi_z)$ of a scalar function ξ at node i. Values ξ_i and ξ_j at node i and its neighboring nodes j are known.

The gradient is estimated by minimizing the distance between ξ_j and a piecewise approximation

$$\xi_j' = \xi_i + \nabla\xi \cdot \Delta\mathbf{r}_{ji},$$

where $\Delta\mathbf{r}_{ji} = (\Delta x_{ji}, \Delta y_{ji}, \Delta z_{ji}) = (x_j - x_i, y_j - y_i, z_j - z_i)$. The distance is expressed as

$$\sum_j (\xi_j - \xi_j')^2 = \sum_j [\xi_j - (\xi_i + \nabla\xi \cdot \Delta\mathbf{r}_{ji})]^2,$$

where $\sum\limits_j$ represents summing all neighbor nodes of node i. Minimizing the distance leads to a linear system,

$$\begin{array}{ccccccccc}
(\sum\limits_j \Delta x_{ji}\Delta x_{ji})\xi_x & + & (\sum\limits_j \Delta x_{ji}\Delta y_{ji})\xi_y & + & (\sum\limits_j \Delta x_{ji}\Delta z_{ji})\xi_z & = & \sum\limits_j \Delta x_{ji}\Delta\xi_{ji} \\
(\sum\limits_j \Delta y_{ji}\Delta x_{ji})\xi_x & + & (\sum\limits_j \Delta y_{ji}\Delta y_{ji})\xi_y & + & (\sum\limits_j \Delta y_{ji}\Delta z_{ji})\xi_z & = & \sum\limits_j \Delta y_{ji}\Delta\xi_{ji} \\
(\sum\limits_j \Delta z_{ji}\Delta x_{ji})\xi_x & + & (\sum\limits_j \Delta z_{ji}\Delta y_{ji})\xi_y & + & (\sum\limits_j \Delta z_{ji}\Delta z_{ji})\xi_z & = & \sum\limits_j \Delta z_{ji}\Delta\xi_{ji}
\end{array}$$

or, simply

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where $\mathbf{A}$ are a matrix of the coefficients, and $\mathbf{x} = (\xi_x, \xi_y, \xi_z)^{\mathrm{T}}$. Then we have

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b},$$

where matrix $\mathbf{A}^{-1}$ is the inverse matrix of $\mathbf{A}$, and its elements satisfy

$$(A^{-1})_{ij} = \frac{\text{cofactor of } A_{ji}}{|\mathbf{A}|}.$$

The cofactor of the element A_{ij} in the square matrix $\mathbf{A}$ equals $(-1)^{i+j}$ times the determinant of the matrix which $\mathbf{A}$ becomes if its ith row and jth column are deleted. If every element in any one row (or column) of $\mathbf{A}$ is multiplied by its cofactor, and the products added together, the result is just the determinant $|\mathbf{A}|$. The elements of matrices $\mathbf{A}$ and thus $\mathbf{A}^{-1}$ are functions of coordinates, so that $\mathbf{A}^{-1}$ can be computed in a preprocessing step and stored in memory. Note that the matrix $\mathbf{A}^{-1}$ is symmetric, so six elements are enough to store it. In 2-D calculations, the linear system degenerates to

$$\begin{array}{ccccccc}
(\sum\limits_j \Delta x_{ji}\Delta x_{ji})\xi_x & + & (\sum\limits_j \Delta x_{ji}\Delta y_{ji})\xi_y & = & \sum\limits_j \Delta x_{ji}\Delta\xi_{ji} \\
(\sum\limits_j \Delta y_{ji}\Delta x_{ji})\xi_x & + & (\sum\limits_j \Delta y_{ji}\Delta y_{ji})\xi_y & = & \sum\limits_j \Delta y_{ji}\Delta\xi_{ji}
\end{array} \tag{4.27}$$

and

$$\mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \begin{pmatrix} \sum\limits_j \Delta y_{ji}\Delta y_{ji} & -\sum\limits_j \Delta x_{ji}\Delta y_{ji} \\ -\sum\limits_j \Delta x_{ji}\Delta y_{ji} & \sum\limits_j \Delta x_{ji}\Delta x_{ji} \end{pmatrix}$$

and determinant $|\mathbf{A}| = (\sum\limits_j \Delta x_{ji}\Delta x_{ji}) * (\sum\limits_j \Delta y_{ji}\Delta y_{ji}) - (\sum\limits_j \Delta x_{ji}\Delta y_{ji})^2$.

For an adaptive quadrilateral grid, some edges of a quadrilateral cell may be split, then it has two neighboring cells outside of each of the edges. In calculating coefficients in (4.27), the sum of these two cells is averaged in the VAS2D. Numerical tests show the average improves the accuracy a little.

4.5.4 Evaluation of fluxes through cell faces

The fluxes through cell faces consist of inviscid fluxes, viscous fluxes and smoothing fluxes. The inviscid fluxes are convection terms and pressure surface terms, or those in the Euler equations. The viscous fluxes are used here to describe all terms that are specific to the Navier-Stokes, i.e., viscous dissipation and thermal diffusion terms. The smoothing fluxes, which are natural extensions of those on the structure grid in Chapter 2, are added to suppress spurious oscillations. Then the vector of fluxes through an interface AB as shown in Fig. 4.9 is

$$\hat{\mathbf{F}} = \hat{\mathbf{F}}_{\text{inviscid}} + \hat{\mathbf{F}}_{\text{viscous}} + \hat{\mathbf{F}}_{\text{smoothing}}, \tag{4.28}$$

where

$$\hat{\mathbf{F}}_{\text{inviscid}} = \begin{pmatrix} \rho u_n \\ \rho u u_n + p n_1 \\ \rho v u_n + p n_2 \\ (\frac{\gamma p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2))u_n \end{pmatrix}; \tag{4.29}$$

$$\hat{\mathbf{F}}_{\text{viscous}} = \begin{pmatrix} 0 \\ -(\tau_{xx}n_1 + \tau_{xy}n_2) \\ -(\tau_{xy}n_1 + \tau_{yy}n_2) \\ -(\tau_{xy}u + \tau_{xy}v + kT_x)n_1 - (\tau_{yx}u + \tau_{yy}v + kT_y)n_2 \end{pmatrix}; \tag{4.30}$$

$$\hat{\mathbf{F}}_{\text{smoothing}} = \begin{pmatrix} -\rho_x\mu_1^* - \rho_y\mu_2^* \\ -(\rho u)_x\mu_1^* - (\rho u)_y\mu_2^* \\ -(\rho v)_x\mu_1^* - (\rho v)_y\mu_2^* \\ -(\rho E)_x\mu_1^* - (\rho E)_y\mu_2^* \end{pmatrix}, \tag{4.31}$$

and

$$u_n = un_1 + vn_2,$$

$$\Delta \mathbf{S} = (n_1, n_2) = (\Delta y_{AB}, -\Delta x_{AB}).$$

The primitive variables which appear in above fluxes are the values obtained by a locally predictor step located at the central point C. The gradients of the primitive variables are locally modified. Since only the gradients of the primitive variables have been calculated, the other gradients are locally calculated. For instance $(\rho u)_x = \rho u_x + \rho_x u$, and so on. We remark that the method of flux evaluation discussed in this section is suitable for a control volume with any shape, triangle, hexagon as well as quadrilateral. These control volumes may be defined by either cell-centered or vertex-centered data structure.

The predictor step

The predictor step locally solves the two-dimensional Euler equations at the center of an interface. Variables at the center are interpolated from one neighboring cell, for density

$$\rho_C = \rho_i + (\nabla \rho)_C \cdot \Delta \mathbf{r}_{fi}. \tag{4.32}$$

The solution is then advanced by half time step,

$$\rho_C^{n+1/2} = \rho_C + (1/2 - \mu)\Delta t (\rho_t)_C, \tag{4.33}$$

where μ is a local smoothing coefficient, and will be discussed later. Other primitive variables p,u and v are treated similarly. Time derivatives such as $(\rho_t)_C$ in (4.33) are determined by the two-dimensional quasilinear Euler equations,

$$\mathbf{M}_t = -\mathbf{A}\mathbf{M}_x - \mathbf{B}\mathbf{M}_y,$$

where $\mathbf{M}$ are primitive variables, $(\rho, u, v, p)^T$, and matrixes $\mathbf{A}$ and $\mathbf{B}$ are

$$\mathbf{A} = \begin{pmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & 1/\rho \\ 0 & 0 & u & 0 \\ 0 & \gamma p & 0 & u \end{pmatrix}; \quad \mathbf{B} = \begin{pmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & \rho \\ 0 & 0 & \gamma p & v \end{pmatrix} \tag{4.34}$$

The gradients are locally modified, and will be shown in the next subsection. After the modification, the predictor step actually solves the Euler equation by the Lax-Friedrichs scheme at the central point C. It should be emphasized that the present method to determine the state at an interface is truly multi-dimensional, since not the Riemann solver but the original Euler equations are used.

Local gradient modification

The simplest way to determine the gradients at the interface are averaging those of two neighboring cells, for density,

$$(\nabla\rho)_C^{avr} = [(\nabla\rho)_i + (\nabla\rho)_j]/2. \tag{4.35}$$

The computational stencil for the average is approximately four cells in direction $\mathbf{l}_{ij}$. The stencil for the average is always four cells on a regular grid as that shown in Fig. 4.20, and it is still approximately four cells in direction $\mathbf{l}_{ij}$ on a quadrilateral mesh. Finally the stencil for evaluating cell i will be five cells at each dimension, which is much larger than a physical wave can propagate. Therefore, special treatment is prefered to make it more compact. In the VAS2D the local gradients at the interface are calculated by a compact operator, which forces the gradient along the direction $\mathbf{l}_{ij}$ to be

$$\mathbf{l}_{ij} \cdot (\nabla\rho)_C = \frac{\rho_j - \rho_i}{\Delta l_{ij}}. \tag{4.36}$$

In practice (4.36) is easily realized by a modification of the averaged gradient

$$(\nabla\rho)_C = (\nabla\rho)_C^{avr} + [\tfrac{\rho_j-\rho_i}{\Delta l_{ij}} - (\nabla\rho)_C^{avr} \cdot \mathbf{l}_{ij}]\mathbf{l}_{ij}. \tag{4.37}$$

It is seen that the gradient $(\nabla\rho)_C$ satisfies (4.36) after the modification. Other variables, velocities and pressure, are treated similarly. Note that after local modification, the interpolation from which cell in (4.32) becomes trivial.

Remark The local modification of gradient is necessary to make the scheme compact. Because the gradient in an interior cell is obtained from four neighboring cells,the stencil of the simply averaged gradient (4.35) has eight cells, as shown in Fig. 4.20a. Since the modification (4.37) forces the gradient along direction $\mathbf{l}_{ij}$ be (4.36), it cleans up the information from cells $i-1$ and $j+1$. The stencil for this modified gradient then becomes compact as shown in Fig. 4.20b. This modification is locally conducted between two neighboring cells, so that it can be applied under an unstructured data structure without difficulty. ¶

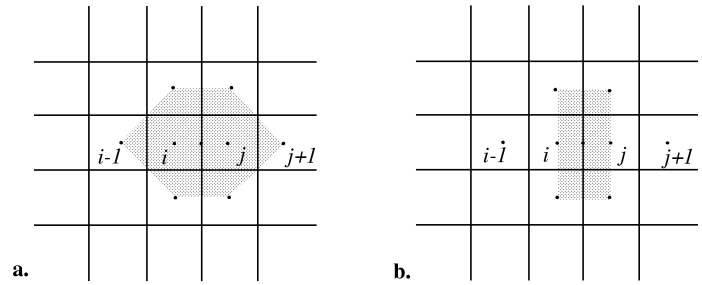


Figure 4.20. Stencil of the local gradient. **a.** the averaged gradient (4.35); **b.** the modified gradient (4.37). The modified gradient uses a more compact stencil.

Determine the smoothing coefficient

Following the method on a structured grid, the indicator located at the interface is

$$\phi_C = \max(\phi_i, \phi_j),$$

where ϕ_i and ϕ_j are calculated in an unstructured way as that in calculating the error detector in the adaptation process,

$$\phi_i = \frac{\sum\limits_k \left[|(\nabla_l M^k)_C - (\nabla_l M^k)_i| / \bar{M}_i^k \right]}{\varepsilon_0 + \sum\limits_k \left[|(\nabla_l M^k)_i| / \bar{M}_i^k \right]}, \tag{4.38}$$

$$\phi_j = \frac{\sum\limits_k \left[|(\nabla_l M^k)_C - (\nabla_l M^k)_j| / \bar{M}_j^k \right]}{\varepsilon_0 + \sum\limits_k \left[|(\nabla_l M^k)_j| / \bar{M}_j^k \right]}. \tag{4.39}$$

The smoothing coefficient along the wave direction is taken as, (see (2.61))

$$\mu = \min[\mu_0, \max(0, \mu_0 - (1 - 2\mu_0) r_C)], \tag{4.40}$$

where,

$$r_C = \frac{1 - \phi_C}{1 + \phi_C}, \tag{4.41}$$

and

$$\mu_0 = \alpha \mu_{up} \quad (1 \le \alpha \le 3/2). \tag{4.42}$$

The coefficient μ_{up} corresponds to the artificial dissipation of the first-order upwind scheme, and is determined by the local maximum CFL number,

$$\mu_{up} = \frac{\sigma^{fast}}{2(1 + \sigma^{fast})}. \tag{4.43}$$

The coefficient μ is decomposed to the Cartesian coordinates, by assuming μ is along direction of the velocity difference between i and j,

$$\mu_1 = |\frac{du}{(du^2 + dv^2)^{1/2} + 10^{-4}}|\mu,$$

$$\mu_2 = |\frac{dv}{(du^2 + dv^2)^{1/2} + 10^{-4}}|\mu.$$

Then

$$\mu_1^* = \mu_1 \frac{(\Delta x)^2}{\Delta t} n_1,$$

$$\mu_2^* = \mu_2 \frac{(\Delta y)^2}{\Delta t} n_2,$$

where the grid size is that between cell i and j,

$$\mathbf{r}_{ji} = (\Delta x, \Delta y).$$

4.5.5 Boundary conditions

In order to calculate the gradients and to solve the conservation laws by numerical methods described in the previous sections, suitable boundary conditions must be prescribed. If all variables were known at a boundary from the knowledge of the physical input, there would be no difficulty. However, this is generally not the case with the Euler and Navier-Stokes equations. At the boundaries, waves are propagating either outward or inward. The outward propagating waves have their behavior defined entirely by the solution at and within the boundary, and no boundary conditions can be specific for them. The inward propagating waves depend on the solution exterior to the computational domain and therefore require boundary conditions to complete the specification of their behavior. More general discussion of boundary conditions may be found in [21], [22].

A schematic of boundary cells is shown in Fig. 4.21. Suppose cell i is a cell with a boundary edge AB. Point m is the image of i reflected by the boundary AB; point C is the center point of edge AB. Cells j,k and h are three neighbors of cell i; some of them may not exist in the case of complicated boundaries. The reconstruction procedure for various types of boundaries depends the definition of variables at m, while the fluxes depends the definition of variables at the center C. Since under the cell-centered data structure, the reference from one edge, say AB,

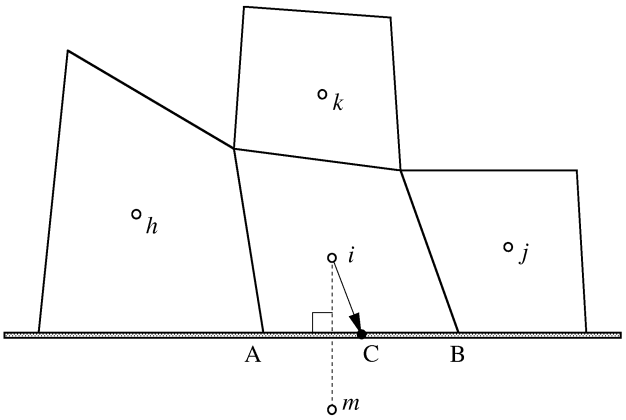


Figure 4.21. Schematic of the treatment of boundary conditions

to far interior neighboring cells, j, k and h, is associated with rather complex logic, particularly because of the possibility that some of these cells may not exist at all. It is decided only to use information obtained from neighbor i to impose all boundary conditions at the boundary edge. The solver loses little accuracy in space by the limited information to which it accesses, but the resulting simplicity and locality of the method may outweigh its drawbacks. In fact, if the order of approximation near the boundary is equal to the scheme order minus one, the overall accuracy of the scheme is not affected [21]. Therefore, in the VAS2D, flow variables that are not physically specified at boundary edges are interpolated from interior neighboring cells in space, while time accuracy is achieved by locally solving the Euler equations. In the interpolation, gradients near boundaries are modified to be compatible with the physical conditions.

In reconstruction procedure, gradients at node cell i are usually constructed by not only the interior neighbors j, k and h but also the image point m. The flow variables at m have to be imposed according to the type of boundaries. For inlet and outlet boundaries, image point m is not taken into account, i.e., gradients at cell i are calculated from other neighboring cells. For the inviscid solid wall boundaries or symmetric lines, the tangential velocity along the boundary is the same, and the normal velocity changes the sign,

$$\rho_m = \rho_i,\ p_m = p_i,\ u_m^t = u_i^t,\ u_m^n = -u_i^n.$$

However the case of viscous solid wall boundaries is more complicated. For a wall

with constant temperature T_w, or an isothermal no-slip wall, the conditions imposed are

$$\rho_m = 2p_i/T_w - \rho_i,\ p_m = p_i,\ u_m = -u_i,\ v_m = -v_i;$$

for a wall has no heat flux, or an adiabatic no-slip wall, they read

$$\rho_m = \rho_i,\ p_m = p_i,\ u_m = -u_i,\ v_m = -v_i.$$

These boundary conditions are deduced by deploying ordinary boundary conditions at the edge AB. For example, no-slip wall has $u_{AB} = 0$, and a second-order average gives $u_{AB} = (u_i + u_m)/2$, then no-slip wall gives $u_m = -u_i$. In the isothermal wall, the condition for density should be $\rho_m = 2p_w/T_w - \rho_i$; because the pressure at wall p_w is unknown, pressure at cell i is used instead. The lost accuracy of this approximation at wall is corrected by modifying the gradients there during solving the conservation laws, which will be described below.

In the flux evaluation, all smoothing fluxes are set to be zero, and other fluxes through edge AB are determined by the flow variables located at the interface center C. The primitive variables there at the time step n are interpolated from unique neighboring cell i,

$$\mathbf{M}_C = \mathbf{M}_i + (\nabla \mathbf{M})_C \cdot \mathbf{r}_{Ci}, \tag{4.44}$$

where the gradients $(\nabla \mathbf{M})_C$ are modified ones from these at cell i based on the types of boundaries. To be able to achieve second-order time accuracy, the final state at point C is given by further propagating the solution by half time step,

$$\mathbf{M}_C^{n+1/2} = \mathbf{M}_C - 1/2\Delta t[\mathbf{A} \cdot (\nabla_x \mathbf{M})_C + \mathbf{B} \cdot (\nabla_y \mathbf{M})_C],$$

where $\mathbf{A}$ and $\mathbf{B}$ are the same as that in predictor step. For the inlet or outlet boundaries, the gradients $(\nabla \mathbf{M})_C$ are the same as those at cell i,

$$(\nabla \mathbf{M})_C = (\nabla \mathbf{M})_i,$$

and the fluxes are identical to these through internal edges. In the VAS2D the flow variables at supersonic inlet boundary cells are all fixed. For inviscid solid boundary, only the gradients of density and pressure are set to be the same as these cell i,

$$(\nabla \rho)_C = (\nabla \rho)_i,\ (\nabla p)_C = (\nabla p)_i;$$

the normal component of velocity gradients $(\nabla u)_i$ and $(\nabla v)_i$ is modified such that the normal component of the velocity obtained by the interpolation (4.44) equals to zero at the boundary. Denote $\mathbf{e}^t$ and $\mathbf{e}^n$ are unit vectors on the tangential and normal directions of edge AB, one has the gradients of tangential and normal velocities

$$(\nabla u^t)_i = (\nabla \mathbf{u})_i \cdot \mathbf{e}^t, \ (\nabla u^n)_i = (\nabla \mathbf{u})_i \cdot \mathbf{e}^n.$$

Keeping the tangential gradients unchanged, and modifying the normal gradients, one gets

$$\begin{array}{l} (\nabla u^t)_C = (\nabla u^t)_i, \\ (\nabla u^n)_C = (\nabla u^n)_i - \frac{1}{\mathbf{r}_{Ci}^2}[u_i^n + (\nabla u^n)_i \cdot \mathbf{r}_{Ci}]\mathbf{r}_{Ci}; \end{array} \tag{4.45}$$

then transforming them back to the Cartesian coordinate,

$$\begin{array}{l} (\nabla u)_C = (\nabla u^t)_C e_1^t + (\nabla u^n)_C e_1^n \\ (\nabla v)_C = (\nabla u^t)_C e_2^t + (\nabla u^n)_C e_2^n. \end{array}$$

The fluxes follows

$$(0, \ p_C^{n+1/2} n_1, \ p_C^{n+1/2} n_2, \ 0)^T. \tag{4.46}$$

In solving the Navier-Stokes equations, the treatment of gradients at the inlet or outlet boundaries is the same as that in the Euler equations. By adding viscous terms in (4.46), the fluxes are

$$(0, \ (p_C - \tau_{xx})n_1 - \tau_{xy} n_2, \ (p_C - \tau_{yy})n_2 - \tau_{xy} n_1, \ 0)^T. \tag{4.47}$$

However, the viscous effects are dominated at the viscous solid walls, and the Euler equations are not supposed to be a good approximation there. Therefore the pressure at the wall is directly interpolated by following (4.44) without propagating the solution by half time step. To be consistent with no-slip conditions, the gradients of velocity should be modified such that velocity interpolated by (4.44) is zero at the boundary. For an isothermal no-slip wall, the modifications read

$$\begin{array}{l} (\nabla \rho)_C = (\nabla \rho)_i - \frac{1}{\mathbf{r}_{Ci}^2}[\rho_i - p_C/T_w + (\nabla \rho)_i \cdot \mathbf{r}_{Ci}]\mathbf{r}_{Ci}, \\ (\nabla p)_C = (\nabla p)_i \\ (\nabla u)_C = (\nabla u)_i - \frac{1}{\mathbf{r}_{Ci}^2}[u_i + (\nabla u)_i \cdot \mathbf{r}_{Ci}]\mathbf{r}_{Ci}, \\ (\nabla v)_C = (\nabla v)_i - \frac{1}{\mathbf{r}_{Ci}^2}[v_i + (\nabla v)_i \cdot \mathbf{r}_{Ci}]\mathbf{r}_{Ci}, \end{array} \tag{4.48}$$

where local density gradients are calculated for evaluation of heat flux. The mass and momentum fluxes are the same as that in (4.47), and heat flux becomes

$$kT_x n_1 + kT_y n_2, \tag{4.49}$$

where the gradients of temperature is given by

$$\nabla T = [(\nabla p)_C - T_w(\nabla \rho)_C]/\rho_C.$$

For an adiabatic no-slip wall, the modifications read

$$\begin{array}{l} (\nabla p)_C = (\nabla p)_i \\ (\nabla u)_C = (\nabla u)_i - \frac{1}{\mathbf{r}_{Ci}^2}[u_i + (\nabla u)_i \cdot \mathbf{r}_{Ci}]\mathbf{r}_{Ci}, \\ (\nabla v)_C = (\nabla v)_i - \frac{1}{\mathbf{r}_{Ci}^2}[v_i + (\nabla v)_i \cdot \mathbf{r}_{Ci}]\mathbf{r}_{Ci}, \end{array} \tag{4.50}$$

with fluxes the same as these in (4.47).

4.5.6 Flow solver under the cell-edge data structure

The flow solver can be easily constructed under the cell-edge data structure with high efficiency. Since the flow solver computes the change of the conservative values of all cells by integrating their interface fluxes. This procedure is done following a routine that consists of four basic steps:

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

The computation of fluxes through an edge requires only its two neighboring cells whose indices are explicitly saved for every edge, so that the flux evaluation is easily vectorized. Similar efficiency is also achieved for other steps by the cell-edge data structure. Adjacent connectivity information can be directly obtained. However, to get the same information, the well-known quadtree or octree structure often needs to climb up to the root of the branch and then climb down to neighboring cells, which is difficult to vectorize if not impossible. The climbing process is avoided by using the present cell-edge data structure. Note that because of step 3, step 4 simply accumulates the fluxes of four edges no mater whether the edge is split or not while preserving conservation.

The flow solver following the four steps has almost no conditional statements, which is usually very difficult for locally adaptive algorithms. Actually the only conditional statements are in step 2, where it deals with five types of boundaries. In fact, the kernel part of the solver, including gradient evaluation, integration of either the Euler or Navier-Stokes equations, and treating different types of boundaries, is just 460-Fortran-line long besides a few directives for vectorization and necessary comments. Thus, the data structure is highly efficient for solving the conservation laws, and can be conveniently applied to the modeling of other physical systems.

The time step is limited by the CFL number,

$$dt = \mathrm{CFL}\frac{L_{\mathrm{edge}}}{|\mathbf{u}| + c},$$

where L_{edge} is the length of cell edge. The practical time step is taken as the minimum one chosen from the values for edges NE1 and NE2 of all cells. Another two edges NE3 and NE4 are neglected, which is reasonable for any regular quadrilaterals.

4.6 Evaluation of the VAS2D

4.6.1 Efficiency of vector and parallel processing

Two cases are tested for measuring the efficiency of vector processing of the VAS2D. Case 1 is a planar shock wave moving in a straight tube, and Case 2 is a planar shock diffracting over a 90° corner. Both cases consist of 12–28 thousand cells. The results are shown in Fig. 4.22. The solver without adaptation (Refinement level = 0) is found to give a maximum speed of around 475 Mflops, and requires only 2.2 μs/cell/step in single processor mode. The vector length is close to 121 of maximum 128, which hardly changes with the refinement level. When the solver is coupled with adaptation (Refinement levels = 1,2,3,4), the measured speeds are ranging from 365 to 405 Mflops. Note that the speeds decrease with high refinement levels. For 4-level refinement, the solver require approximately 3 μs/cell/step. This speed is around two orders faster than an adaptive solver on the Cartesian grid conducted in a workstation using the upwind scheme with either exact or approximate Riemann solvers [11], [23]; it is much faster than an adaptive solver on a similar machine [24] [4], which uses a central difference with conventional artificial damping.

[4]The explicit data of the efficiency of the solver in the paper [24] is not available. The authors executed the solver coupled with adaptation procedure with an initial grid consisting of 60 thousand

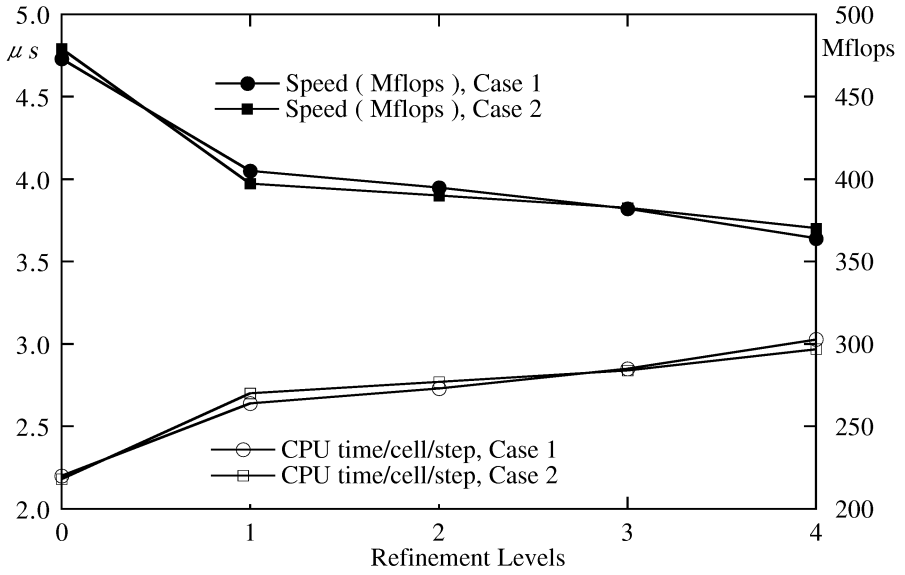


Figure 4.22. Vector efficiency of the VAS2D with the Euler solver using different refinement levels

The VAS2D performs nearly four times as fast as a well vectorized solver [25] [5] on triangular grids which permits no adaptation. It is believed that with adaptation the VAS2D even perform much better, because the VAS2D is able to efficiently vectorize the adaptation procedure. The efficiency is not only due to the well-vectorized data structure but also due to the flow solver, which is a central-difference scheme with the conservative smoothing method.

Parallel execution speed-ups of the VAS2D are shown in Fig. 4.23 as a function of number of processors (CPUs). The results are obtained by Cray software *atexpert*. It should be noted that the increase in speed is not in exact proportion to the number of processors due to the additional overheads of partitioning the work among processors. Another more important reason is the small number of cells that are to be refined or coarsened. It is quite common in 2-D calculation hundreds of cells are

points and adapted a group of 4000 cells at each time step to measure performance. The total execution time was $1.86s$ per step, then it was approximately $1.86/60,000s = 31\mu s$/cell/step. The VAS2D is ten times as fast as this speed.

[5] In reference [25], the authors gave a data on Cray Y-MP run at 150 Mflops. It requires $0.39s$/step for 15606 vertices, then the speed is $24.8\mu s$/cell/step. Since the numbers of vertices and cells are similar on a quadrilateral grid, the difference between them is not taken into account here. We assume the solver could run at 450 Mflops on Cray C90, the speed may reach $8.3\mu s$/cell/step. Therefore the present VAS2D is close to 4 times as fast as the solver without adaptation.

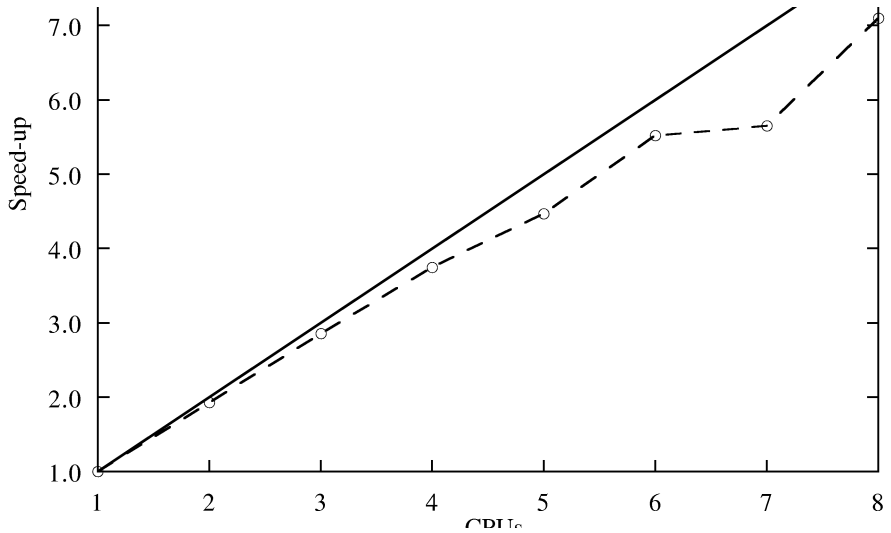


Figure 4.23. Parallel efficiency measured by Cray performance tool *atexpert*

refined in one time step, which is a job just enough for one or two CPUs with a vector length 128. The adaptation procedure conducts such a small job that most computer resources are wasted when using 8 CPUs. The fully vectorized and parallelized flow solver contributes most to the parallel speedup of the VAS2D. Actually even using one processor, the VAS2D may calculate most unsteady and steady gas-dynamic phenomena in one minute with a high resolution. However in 3-D computation an adaptation procedure has to treat thousands of cells, then parallelism will become more important.

4.6.2 CPU time distribution on subroutines

CPU time used in the main subroutines, mesh adaptation, gradient evaluation and integration of the conservation laws is measured for solving both the Euler equations and the Navier-Stokes equations. The test case computes shock diffraction with a 3-level refinement conducted at every time step. The test results are shown in Fig. 4.24. It is found that the percentage of CPU time consumption varies no more than 3% in other test cases. It is seen that the flow solver including reconstruction and integration uses around 86% and 91% of total time for the Euler and the Navier-Stokes respectively. Due to additional viscous and heat condition terms, the Navier-Stokes reasonably consumes more percentages than the Euler integration. The mesh adaptation requires less than 10% time. Except the error indicator that is calculated

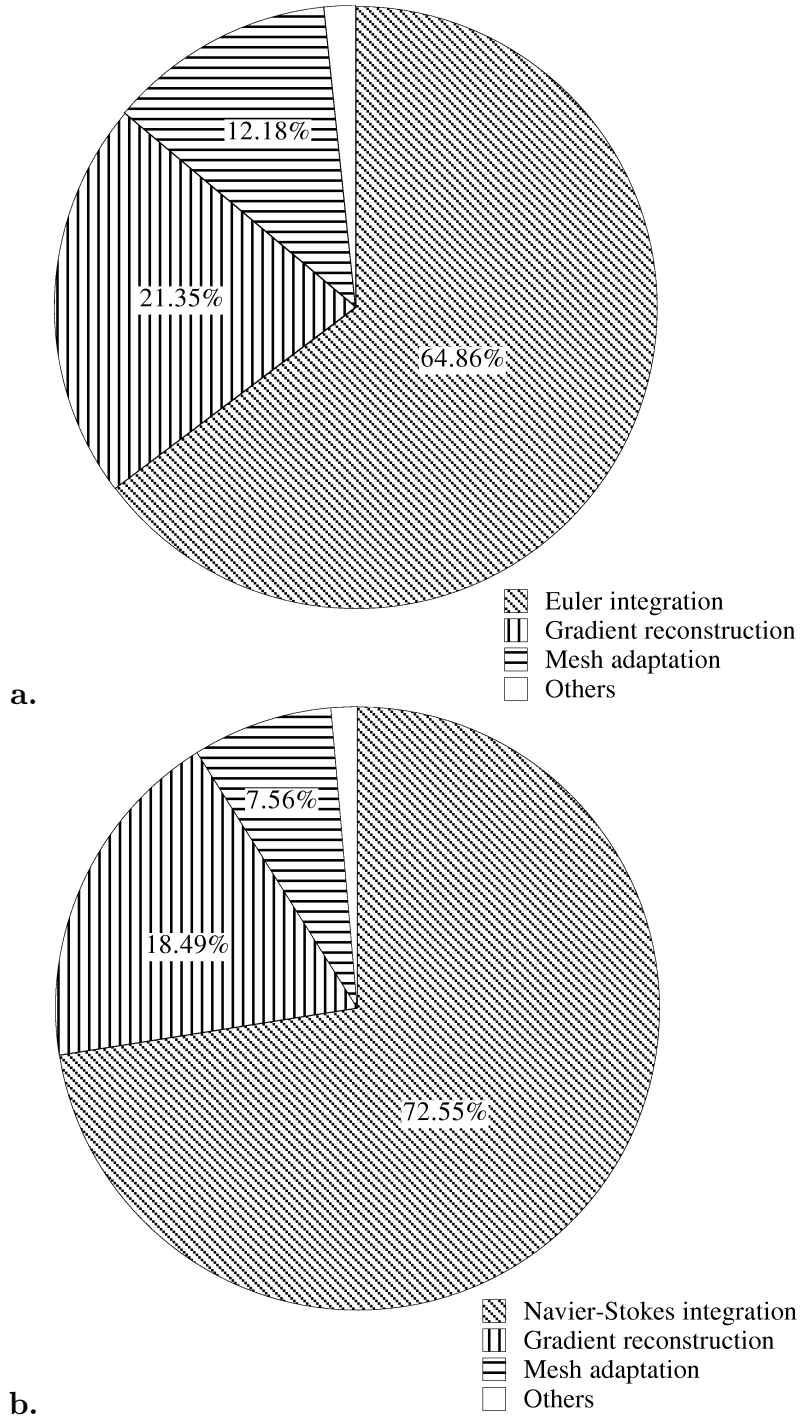


Figure 4.24. CPU time consumption by main subroutines: **a.** the VAS2D with the Euler solver; **b.** the VAS2D with the Navier-Stokes solver. The data is gathered by Cray performance tool *flowtrace*.

with the flow solver, all changes caused by mesh adaptation, such as mesh refinement and coarsening, the interpolation from the old mesh to the new mesh, and updating the volume of cells, are included in the subroutine of mesh adaptation. Clearly the overheads of mesh adaptation is very cheap. It should be mentioned that the overheads of the adaptation procedure is usually much higher. For instance, for an efficient 2-D adaptive solver on triangles [26], the overheads of the adaptation is less than 5% of total computer time on a personal computer, but it increases to around 50% on a vector machine because the adaptation procedure is poorly vectorized. Kallinderis and Vidwans's adaptation procedure [24] requires even over 80% on the connection machine CM-2. The high efficiency of present adaptation should be contributed to the greatly simplified logic under the vectorizable cell-edge data structure.

The percentage of the mesh adaptation decreases from 12% to 8%, in solving the Navier-Stokes equations. It suggests that the overheads of mesh adaptation is relatively lower with a heavy solver. Therefore the overheads of adaptation would be much cheaper in solving complicated physical systems with chemical reactions.

4.6.3 Cartesian mesh *vs.* adaptive mesh

Compared with the Cartesian structured mesh, the overheads of an adaptive mesh comes not only from the treatment of adaptation but also from its unstructured data. For instance, a solver on the Cartesian mesh need not compute any geometrical information at all; however an unstructured flow solver has to take into account cell volumes, interface orientations, and gradients. In a vector machine the overheads usually becomes very large because of the difficulties in efficiently vectorizing the unstructured solver especially the adaptation procedure. However the VAS2D employs the essential vectorizable data structure, so that both the flow solver and mesh adaptation are fully vectorized. In this section a quantitative comparison of efficiency is made between the Cartesian mesh and the adaptation mesh.

Table 4.2 records CPU time and memory requirement, compared with these on the Cartesian grid. If without adaptation but with all unstructured geometrical calculations, it is found that the VAS2D requires 2.4 times of CPU time and 5.3 times memory as much as the structured solver conducted on the same Cartesian mesh. The structured solver is computed at a speed of around 450 Mflops. With mesh

Table 4.2. Comparison of CPU time and memory requirement of the flow solver on the Cartesian mesh and the adaptive mesh. The data of the initial Cartesian mesh is set to be 1.

Refinement levels	0	1	2	3	4
CPU time: adaptive mesh	2.4	8.0	24.7	83.0	301.2
CPU time: Cartesian mesh	1	8	64	512	4096
Ratio of CPU time	2.4	1.0	0.39	0.16	0.074
Memory: adaptive mesh	5.3	8.1	12.6	22.5	43.3
Memory: Cartesian mesh	1	4	16	64	256
Ratio of memeory	5.3	2.0	0.79	0.35	0.17

adaptation, 1-level refinement is enough to compensate the overheads in CPU time, and 2-level refinement to compensate the memory overheads. When the VAS2D uses 4-level refinement, it is more than ten times as fast as that on the Cartesian grid with only about 20% memory. The CPU time and memory ratios of two meshes can be approximated described by

$$\frac{\text{Memory : adaptive mesh}}{\text{Memory : Cartesian mesh}} = C_{\text{memory overheads}} \left(\frac{C_{\text{level}}}{4}\right)^L;$$

$$\frac{\text{CPU time : adaptive mesh}}{\text{CPU time : Cartesian mesh}} = C_{\text{CPU overheads}} \left(\frac{C_{\text{level}}}{4}\right)^L,$$

where L is the level of refinement. C_{level} is the ratio of cell numbers between two consecutive refinement levels, which is a problem-dependent value. In the present test case of shock diffraction C_{level} is around 1.7. For a uniformly refined grid C_{level} is exactly 4 which appears in the denominator. Generally speaking, compared with the uniform Cartesian grid, the adaptive grid is more effective for a high refinement level. However the highest level is practically limited either by unnecessary high resolution with fine grids or by available memory and tolerable computer time. According to personal experience, a four or five level of refinement is usually fine enough to simulate flows with shock waves. Therefore by considering a 4-level refinement, it is concluded that the VAS2D requires approximately 10% computer time and 20% memory as much as a solver on the uniform Cartesian grid. In addition the VAS2D may treat complex geometry straightforwardly.

Fig. 4.25 gives a few results of the test case. It is seen that the density contours are virtually independent of the grid adaptation. Of course, shock waves become

linearly sharper for high level adaptations. Because the post-shock flow is subsonic, expansion waves propagate upstream. The incident shock wave is gradually attenuated to the downstream wall. These phenomena are resolved similarly on three different grids. The finest cells are economically distributed around shocks and vortices. This is the reason that the ratio C_{level} is around 1.7 which is much less than 4.

4.7 Numerical results

A variety of test cases have been carefully chosen to show the effectiveness of the VAS2D. The test cases cover steady/unsteady, inviscid/viscid flows with weak/strong shock waves. First subsections give the results of the Euler equations, including shock diffraction (Section 4.7.1), supersonic channel flow (Section 4.7.2), and interactions of shock / circular-cylinder (Section 4.7.3) and shock / square-cylinder (Section 4.7.4); Section 4.7.5 shows a results of the Navier-Stokes equations, shock / boundary layer interaction; last subsection (Section 4.7.6) gives a comparison of the solutions of the Euler and the Navier-Stokes solvers about shock wave moving in a nozzle. The ratio of the specific heats is always assumed to be 1.4.

4.7.1 Shock diffraction

A shock wave diffracting at a sharp convex corner has become a benchmark problem not only in the computational fluid dynamics but also for shock tube experiments [27]. In the past three decades, many experimental and numerical results have been published (for example Skews [28], Bazhenova [29], Hillier [30], Sun and Takayama [31]). This subsection is devoted to shock wave diffraction over a 90° corner for shock Mach number ranging from 1.3 to 16. To demonstrate the flexibility of the VAS2D, computations are conducted under the same numerical parameters. The geometry consists of three 1×1 squares. Every square can be divided to many fine cells. A shock wave is initially at 0.5 to the left of the corner point. The CFL number is 0.7, and it is unchanged for unsteady computations. A discontinuity may pass one cell in one time step at such a high CFL number, so that the adaptation is performed at every time step. 3-level refinement is adopted, and it is $256 \times 256 \times 3$ cells if using a uniform grid. The required CPU time is no more than 25 seconds in each case.

The results for shock Mach number 1.3, 1.6 and 2.0 are shown in Fig. 4.26. In

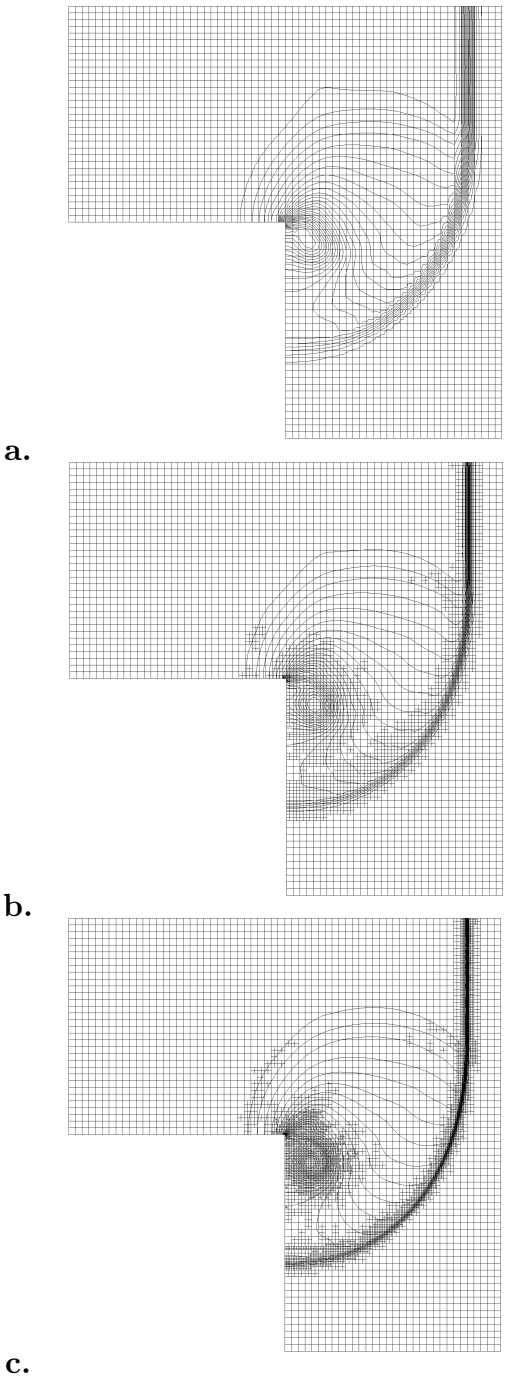


Figure 4.25. Shock diffraction using different levels of refinement: $M_s = 1.6$, CFL = 0.7, : **a.** No adaptation, 3072 cells, CPU time = 0.592s; **b.** 1-level refinement, 4773 cells, CPU time = 1.98s; **c.** 2-level refinement, 7695 cells, CPU time = 6.09s.

these cases the post-shock flow all are subsonic. The VAS2D exactly reproduces the diffracted shocks, expansion waves, vortices. The secondary shock wave which moves above the vortex is visible for $M_s = 1.6$. Note that the finest cells shown in the meshes clearly indicate these important physical configurations.

Fig. 4.27 gives strong shock diffraction with shock Mach number 4, 8 and 16. Real gas effects are not considered here, because the main object of this section is not to investigate physical phenomena. For strong shock diffraction, flow patterns close to the corner are similar. There are expansion waves, a slipstream and a system of secondary shock wave. These are well resolved for all shock strengths. Since post-shock flows are supersonic, the expansion waves can not propagate upstream and become the Prandtl-Meyer expansion fan around the corner. The vortex is not observable for very strong shocks. A clear difference between strong shocks lies in their reflection configurations on the downstream wall. The foot of $M_s = 2$ shock is almost perpendicular to the wall as seen in Fig. 4.26e; $M_s = 4$ shock forms a sharp turn there, but the waves behind it are somewhat continuous. The waves at the wall for $M_s = 8$ can be recognized as a sharp discontinuity, or a reflected shock wave, then it is a Mach reflection since a very short Mach stem is seen. For $M_s = 16$ shock the Mach stem is not visible anymore so the configuration is a regular reflection. These changes qualitatively agree with experimental observations [29]. Combined with the results in weak shock diffraction, it is clear that the smoothing flux efficiently cleans oscillations behind shock waves with quite different shock strengths.

Fig. 4.28 gives a comparison between numerical and experimental shock diffraction for $M_s = 1.475$. It is seen that the Euler solver resolves very well the main wave patterns, such as the diffraction shock and upstream moving expansion waves. However the secondary vortex which appears at the downstream of the corner in the photo is not visible in numerical result. It suggests that generation of the secondary vortex is due to the viscous effects.

4.7.2 Steady supersonic channel flows

Steady flows are computed in a channel with a compression corner, followed by an expansion corner, which is similar to that in [11]. The length of computation domain is 4. The inflow Mach number $M = 2$ and 4. Solutions are given at $t = 4$ for $M = 2$ and $t = 2$ for $M = 4$. The upper and lower surfaces of the channel are

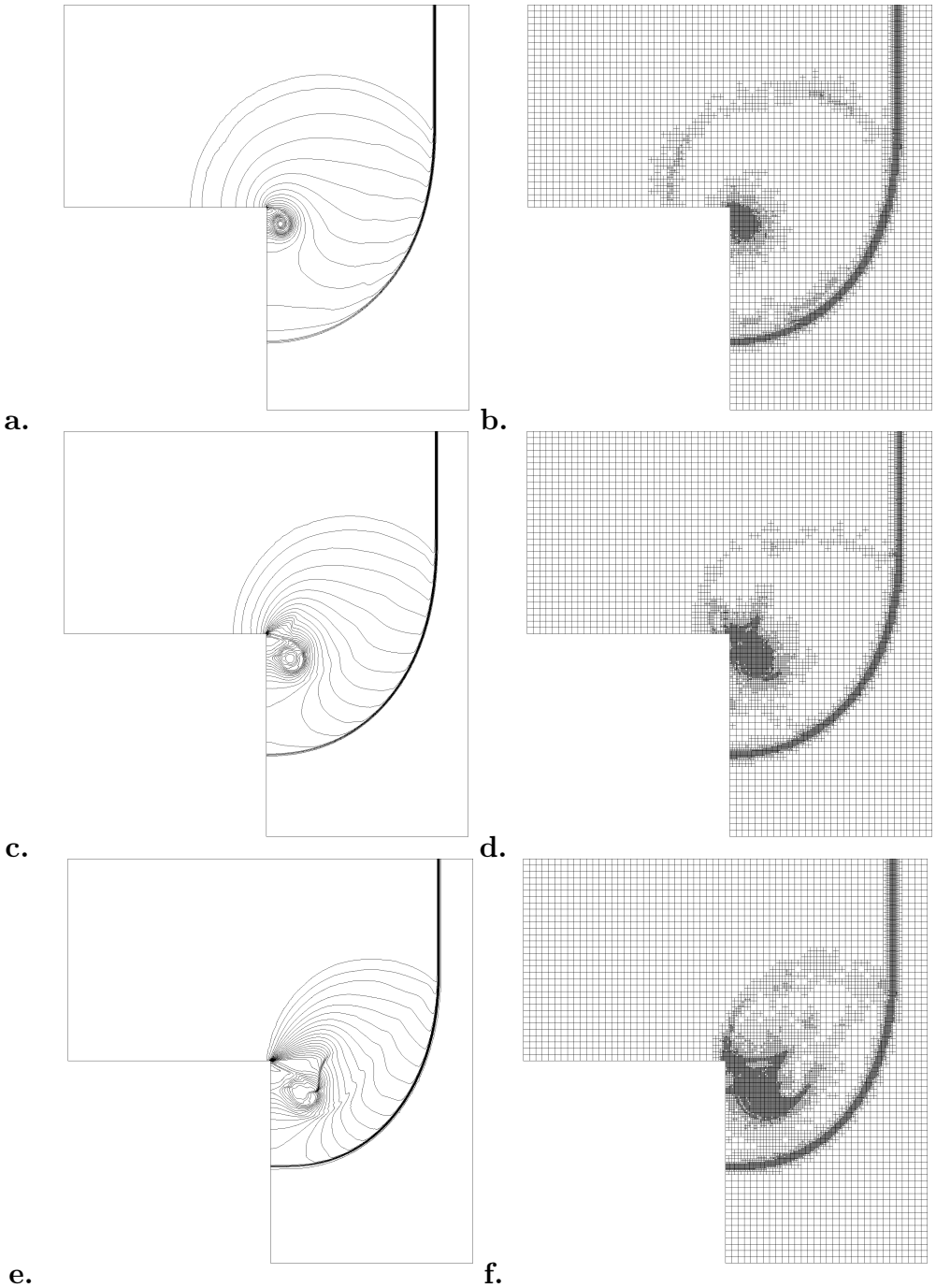


Figure 4.26. Shock diffraction, 3-level refinement, CFL = 0.7: **a.b.** $M_s = 1.3$, 12174 cells, CPU time = 18.9s, $\alpha_f = 0.02$; **c.d.** $M_s = 1.6$, 13077 cells, CPU time = 20.5s; **e.f.** $M_s = 2.0$, 16026 cells, CPU time = 22.3s.

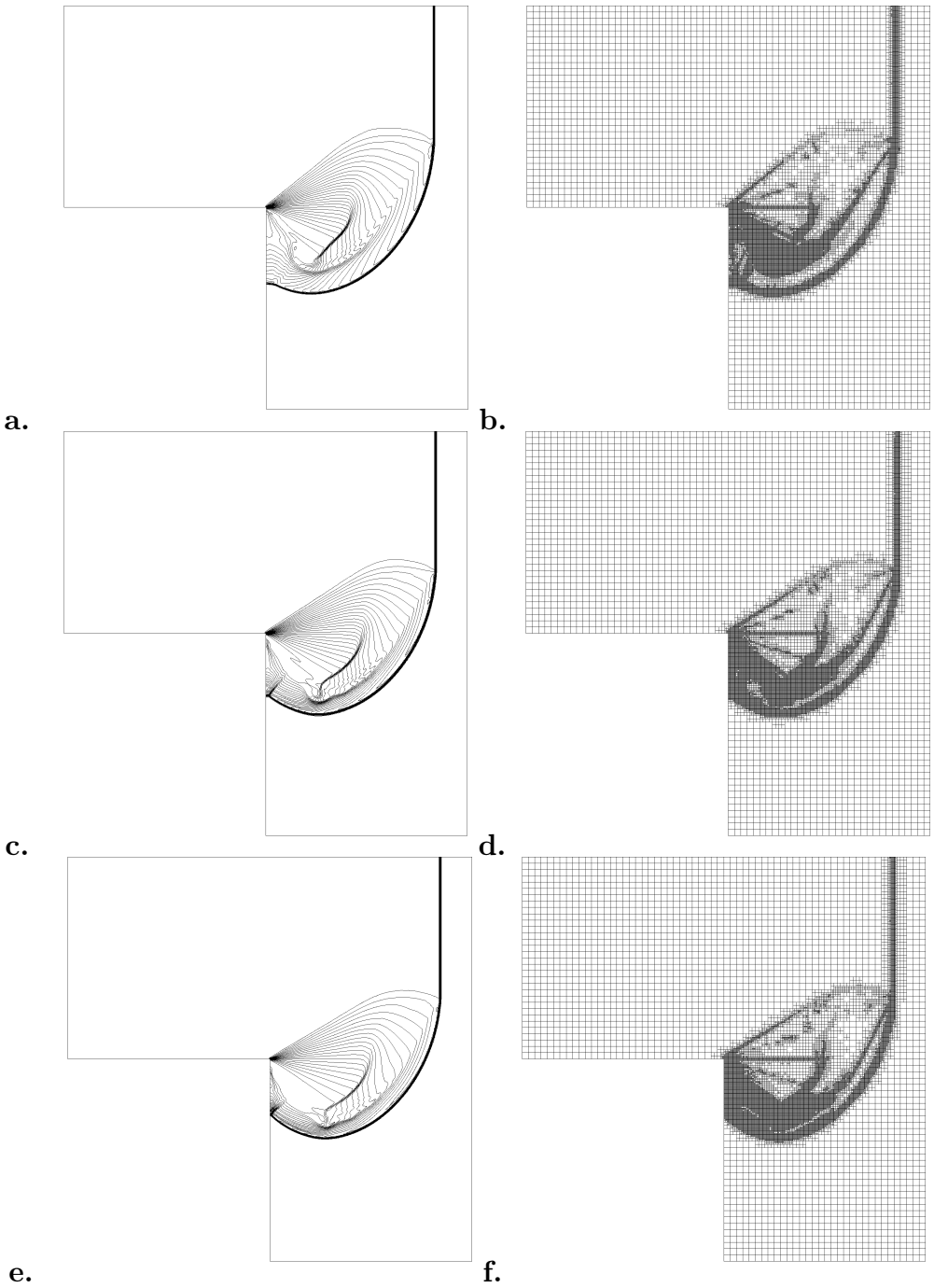


Figure 4.27. Shock diffraction, 3-level refinement, CFL = 0.7: **a.b.** $M_s = 4.0$, 20595 cells, CPU time = 23.0s; **c.d.** $M_s = 8.0$, 20817 cells, CPU time = 22.7s; **e.f.** $M_s = 16.0$, 20349 cells, CPU time = 22.4s.

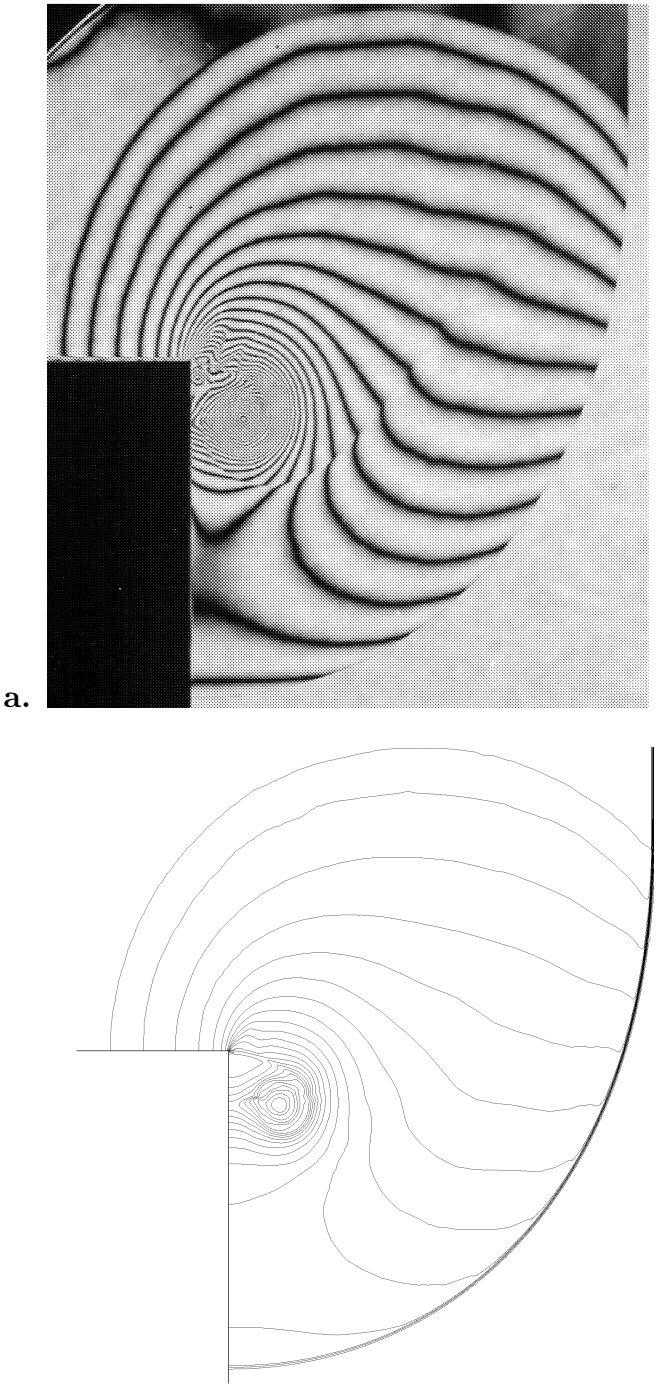


Figure 4.28. Shock wave diffraction for $M_s = 1.475$: **a.** experimental photo obtained by holographic interferometry; **b.** numerical isopycnics obtained by the VAS2D with the Euler solver.

reflecting conditions, with a supersonic inflow on the left, and outflow on the right. The grid adaptation is performed every three time steps. When using a refined grid, the VAS2D calculates for 60% of total time on a coarse grid to get a good initial state, then starts the adaptation procedure. Computer time is reduced by around half using this simple strategy.

Fig. 4.29a gives a result on a coarse grid with a 15° corner. It is seen that shock thickness is about two-cell size even when shocks are reflecting from the walls. The grid has a similar cell size to that in unsteady computation shown in Fig. 4.25a. It seems the transition region of a shock is narrower in steady computation than in the unsteady one.

Fig. 4.29bc record the results using 3-level refinement. There is an attached shock at the compression corner. The shock is reflecting from the upper surface and forms a Mach reflection. But for $M = 4$ it forms a regular reflection, as shown in Fig. 4.29de. The reflected shock interacts with the Prandtl-Meyer fan starting from the corner, and then reflects from the lower surface as well. For $M = 2$ case, the slipstream creates a discontinuity in isopycnics, emanating from the triple point and nearly going parallel to the upper surface, as shown in Fig. 4.29b. It is recognizable even in the very coarse grid in Fig. 4.29a.

A shortcoming of the smoothing or filter in computing unsteady [32] and steady flows [33] is the fluctuation of contours. The present approach has successfully overcome this. It is seen that density contours are smooth in expansion wave regions even on a grid with different sized cells.

4.7.3 Shock / circular-cylinder interaction

Shock wave propagation over an obstacle is of interest in predicting the dynamic response of structures to an internal or external explosion. A circular-cylinder is one of the simplest obstacles with only one geometrical length scale, say its radius. Unlike self-similar shock diffraction shown in Section 4.7.1, shock wave moving over a circular-cylinder is truly unsteady. This subsection tries to show the ability of the VAS2D in simulating such truly unsteady flows and to demonstrate the treatment of smooth boundary such as a circle.

The computation is conducted on the initial mesh shown in Fig. 4.5 with 4-level

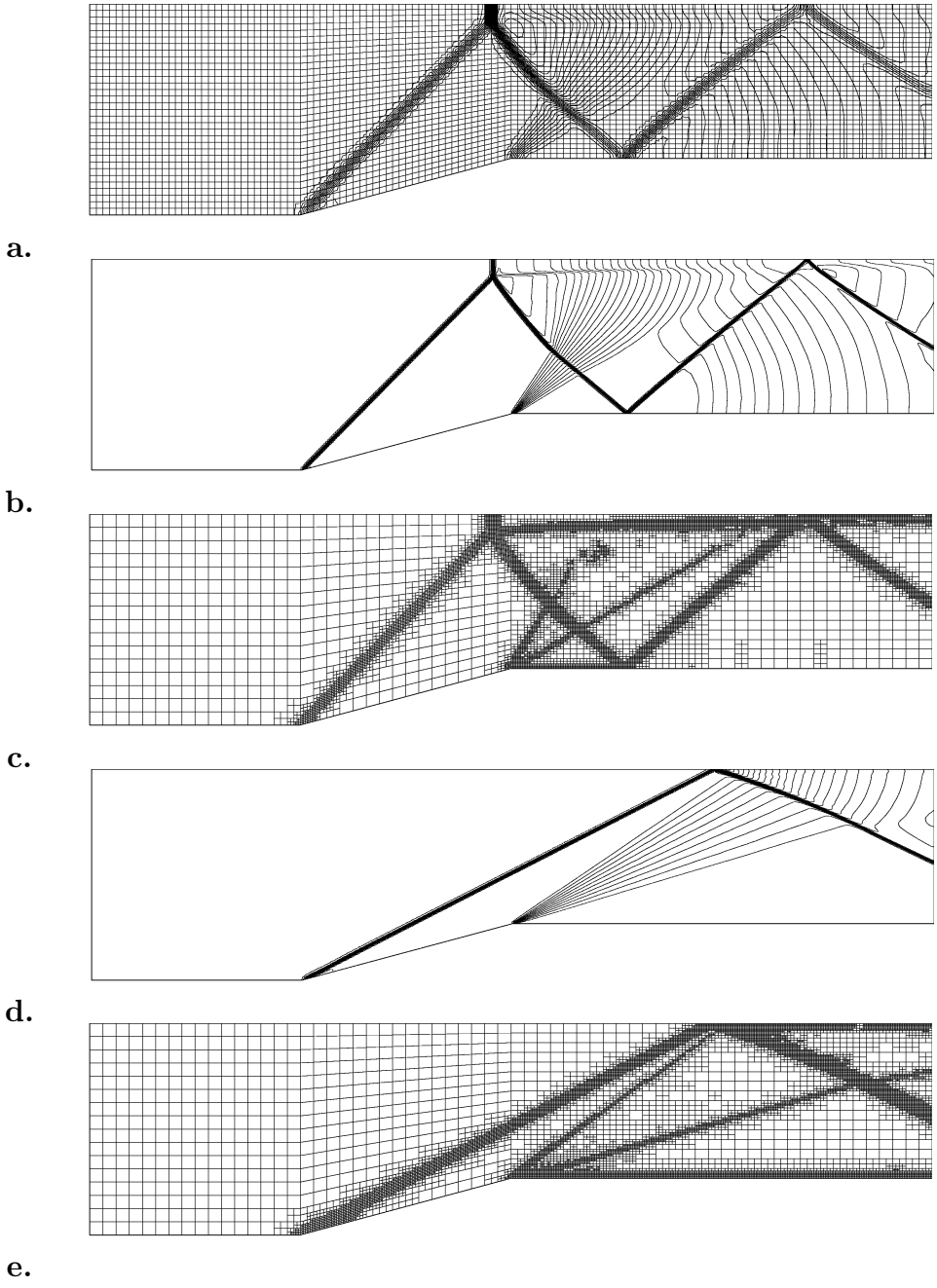


Figure 4.29. Steady flow in a tunnel with 15° bump: **a.** $M = 2$, $t = 4$, with 1-level uniform refinement; **b.c.** $M = 2$, $t = 4$, 3-level refinement, 11338 cells, 1670 steps, CPU time = 47.8s; **d.e.** $M = 4$, $t = 2$, 3-level refinement, 11386 cells, 1381 steps, CPU time = 47.4s.

refinement. The radius of the half circle is equal to 1. The results for $M_s = 1.7$ are shown in Fig. 4.30 to Fig. 4.31. Fig. 4.30ab correspond to the instant when the shock wave hit the cylinder. There are some refined cells visible behind the shock wave, which are the vestiges of an initially discontinuous shock approaching to a continuous shock. This phenomenon can also exist in the results using some other schemes.

The shock wave configuration over the cylinder forms a Mach reflection, which consists of the incident shock, reflected shock, Mach stem, and slipstream, as shown in Fig. 4.30cd. The Mach stem moves over the cylinder surface, and then reflects from the symmetric line. In Fig. 4.31ef, the Mach stem has been reflected and start to create a new triple point there. The new triple point grows up, then a new slipstream is generated stemming from it. Fig. 4.31gh clearly records the two triple shock systems. The reflected shock that moves along the cylinder surface induces the separation of a vortex bubble behind the cylinder. The present numerical results agree with these computed by a second order Godunov-type scheme [34], and agree well with the experimental observations except the region close to the cylinder where the boundary layer is not negligible.

4.7.4 Shock / square-cylinder interaction

In numerical simulation, the initial mesh of shock / square-cylinder interaction is shown in Fig. 4.7. The distance from the square center to its tip is taken as 1, and the square is oblique to the horizontal line by 30°. The distances from the square center to the upper and lower surface are 2 and 1.3 receptively. The asymmetric setup is intentionally designed to generate complicated flows. The results for incident shock Mach number $M_s = 2$ are shown in Fig. 4.32 to Fig. 4.33. The shock configuration in this test is actually a combination of most basic phenomena of compressible flows, such as shock diffraction, reflection, vorticity generation, and interactions of shock/expansion waves, shock/vortex. The VAS2D is able to capture the flows without difficulty. On the other hand, it suggests that the scheme that combines the Lax-Wendroff with conservative smoothing is robust.

It should be mentioned that the scheme tolerates irregular meshes. The mesh cells around the square are very irregular, as shown in Fig. 4.32b. Some nodes have fives neighbors, some have only three neighbors. From Fig. 4.32c and Fig. 4.33eg,

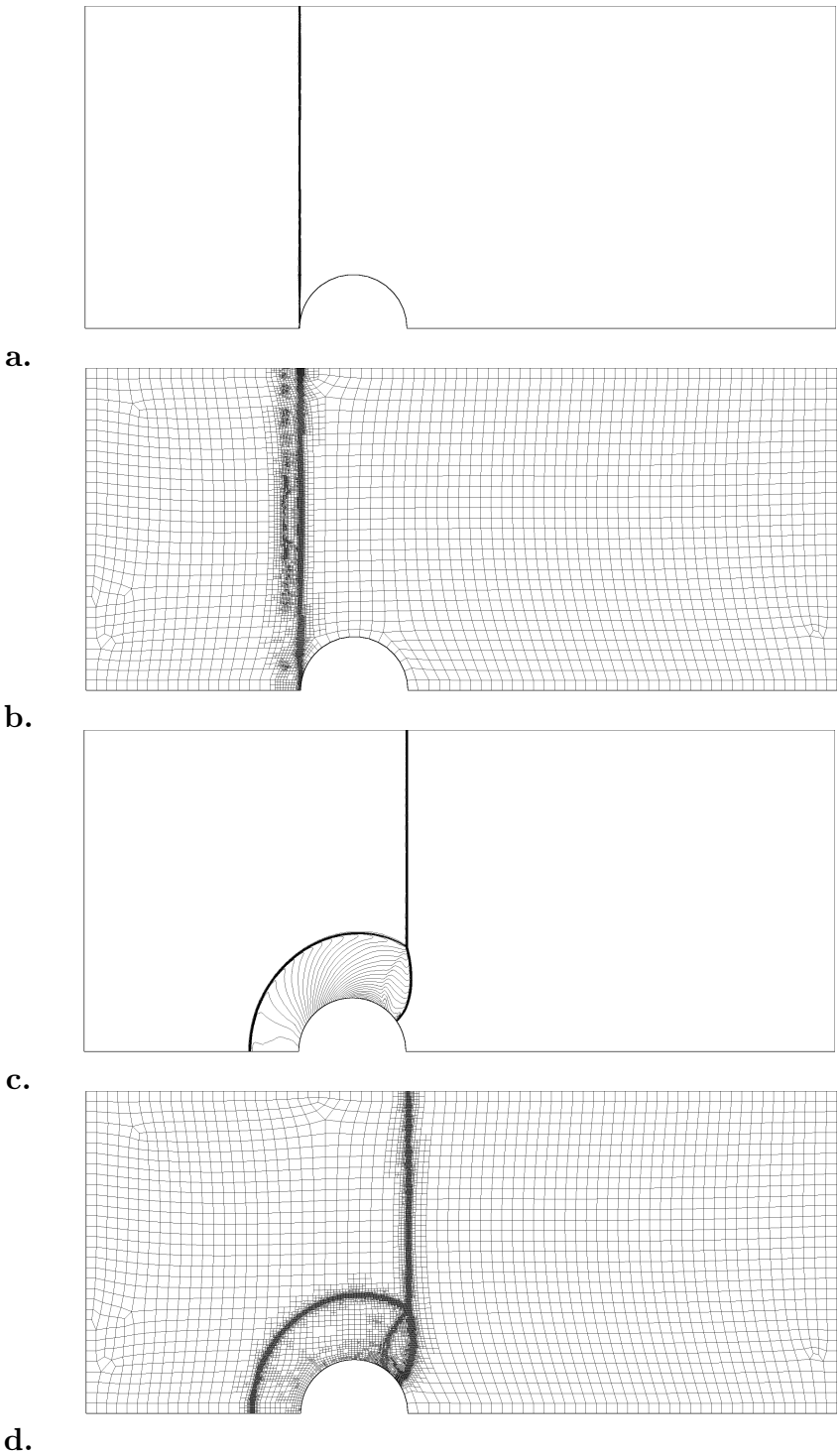
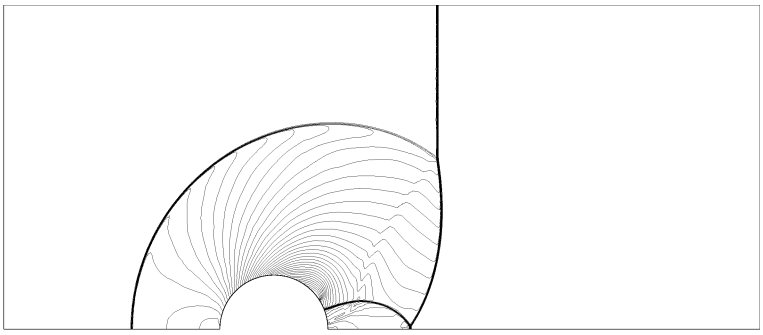
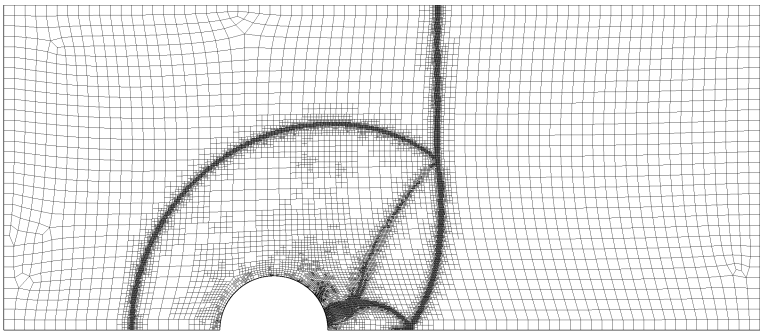


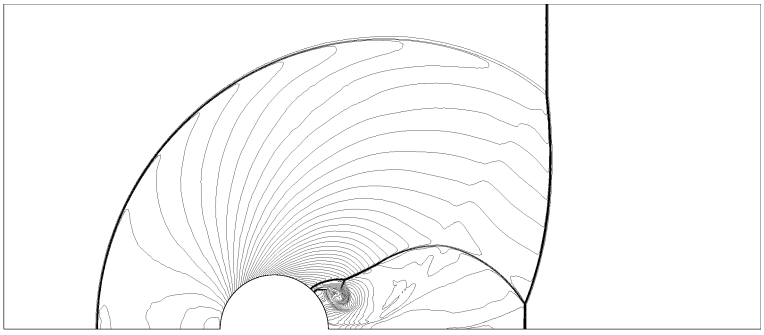
Figure 4.30. Shock / circular-cylinder interaction, isopycnics and corresponding adaptive meshes. Incident shock Mach number $M_s = 1.7$, 4-level refinement, CFL = 0.7. **a.b.** $t = 0.0$, 9735 cells, CPU time = 3.0s; **c.d.** $t = 1.0$, 12471 cells, CPU time = 18.6s.



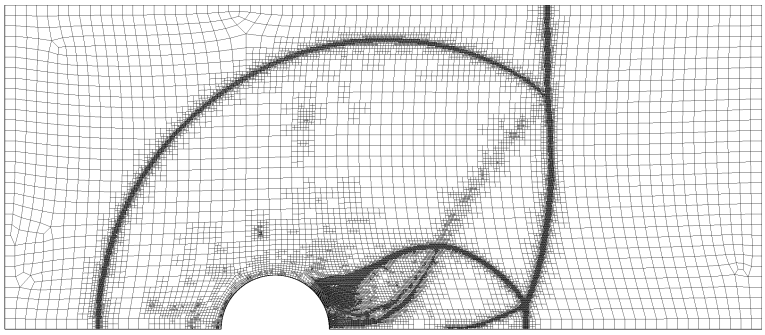
e.



f.



g.



h.

Figure 4.31. Shock / circular-cylinder interaction, isopycnics and corresponding adaptive meshes (Continued). **e.f.** $t = 2.0$, 20253 cells, CPU time = 47.4s; **g.h.** $t = 3.0$, 29790 cells, CPU time = 101.5s, 1800 steps.

density contours there are reasonably smooth, and no clear grid-dependent error is visible. This property can also be seen in the results of shock / circular-cylinder interaction, where the mesh cells around the circle is rather skewed as seen in Fig. 4.30b.

Fig. 4.34 gives a result of shock wave interacting with a rectangle with the ratio of length and width 4. By comparing with the experimental photo, it is seen that primary wave patterns are well captured by the Euler solver. The shape and location of main vortices agree very well. The disagreement is observable close to the back wall of the rectangle, especially the system of secondary vorticies as has been mentioned in shock diffraction pheonmenon.

4.7.5 Shock / boundary layer interaction

This section gives a result of the Navier-Stokes solver, shock/boundary layer interaction. The accuracy of the Navier-Stokes solver is first tested with the computation of the viscous flow around an isothermal flat wall at zero incidence. The free-stream Mach number is equal to 1, and the Reynolds number Re based on a unit length is equal to 1.01×10^4. The theory of boundary layer shows that for the coefficients of viscosity and thermal conductivity are linear functions of temperature, local skin friction coefficient remains the same as in incompressible flow (see, [35]):

$$\tau_w = 0.332\sqrt{u^3/xRe},$$

where u is the non-dimensional free-stream speed and x the distance from the tip of wall. Numerical results of the ratio of τ_w to $\sqrt{u^3/xRe}$ along the flat wall are plotted in Fig. 4.35. It is seen that the numerical results agree very well with the boundary layer solution 0.332, but great discrepancy exists close to the tip. In the region close to the tip of the wall, the variations normal to the wall become the same order as those along the wall, so that the boundary layer assumption is not valid there. This explains the discrepancy between the numerical results obtained by the full Navier-Stokes equations and the approximate solution predicted by the theory of boundary layer. The agreement in the regions far from the tip indicates the excellent accuracy of the present Navier-Stokes solver.

In simulating shock / boundary layer interaction, an incident shock moves from left to right, and reflects from the end wall. The reflected shock propagates over and interacts with the boundary layer which is created behind the incident shock.

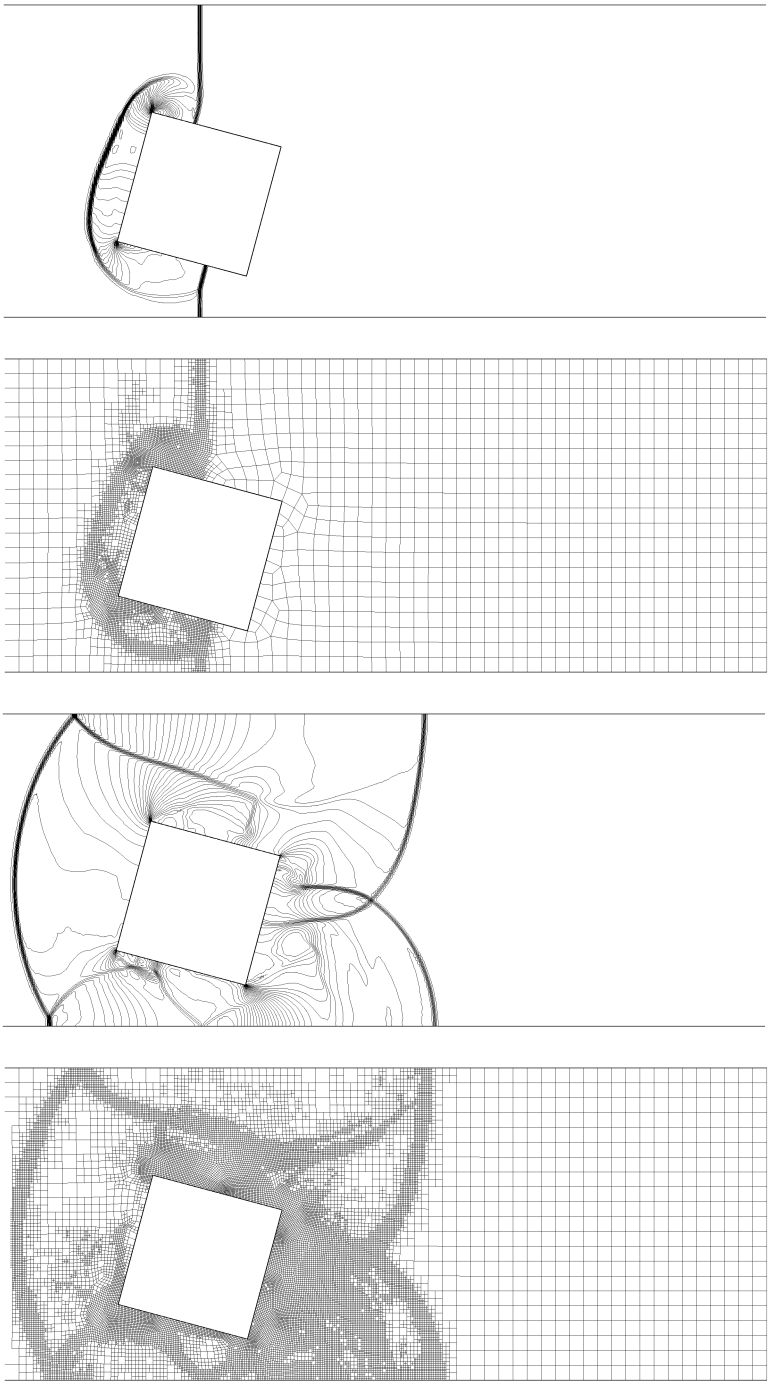
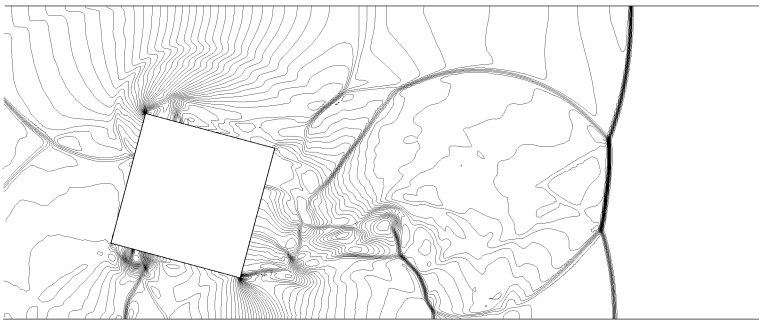
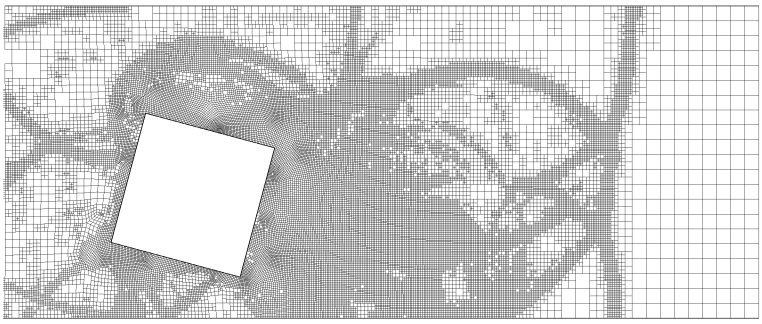


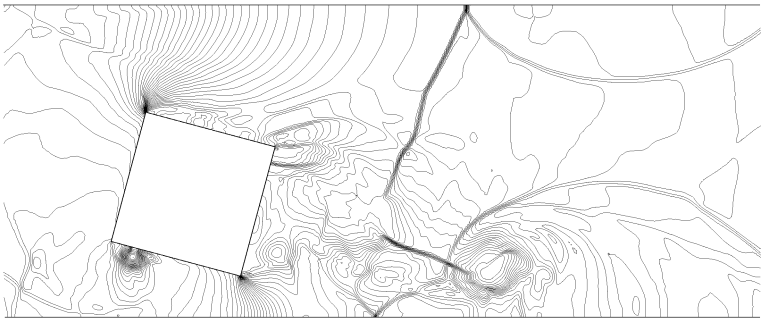
Figure 4.32. Shock / square-cylinder interaction, isopycnics and corresponding adaptive meshes. Incident shock Mach number $M_s = 2$, 3-level refinement, CFL = 0.7. **a.b.** $t = 0.0$, 8149 cells, CPU time = 4.3s; **c.d.** $t = 1.0$, 22981 cells, CPU time = 29.6s.



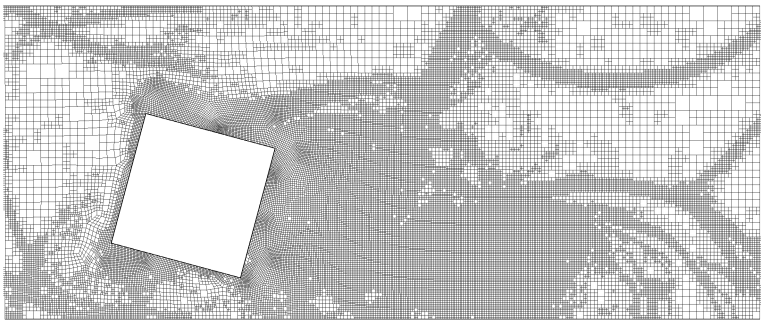
e.



f.



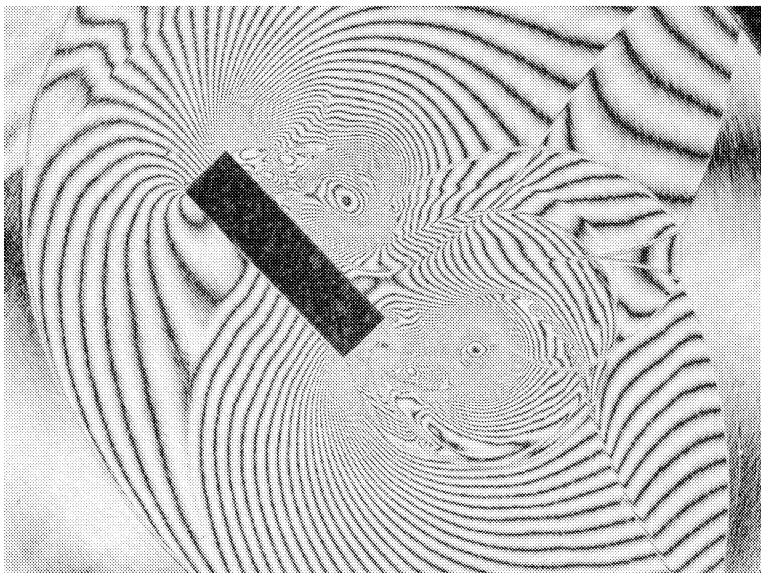
g.

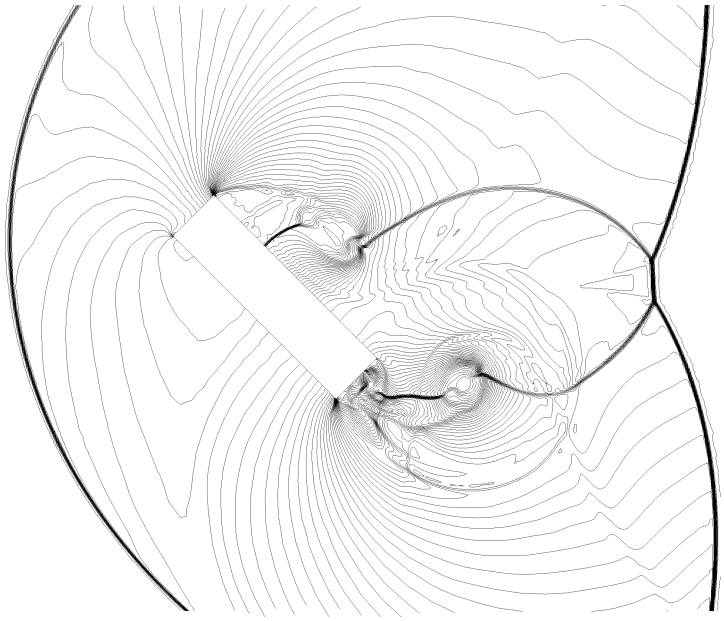


h.

Figure 4.33. Shock / square-cylinder interaction, isopycnics and corresponding adaptive meshes (Continued). **e.f.** $t = 2.0$, 38383 cells, CPU time = 82.0s; **g.h.** $t = 3.0$, 48334 cells, CPU time = 147.6s, 1767 steps.

a.





b.

Figure 4.34. Shock wave interaction with an oblique rectangle. **a.** experimental photo obtained by holographic interferometry; **b.** numerical isopycnics obtained by the VAS2D with the Euler solver.

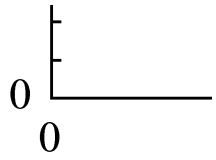


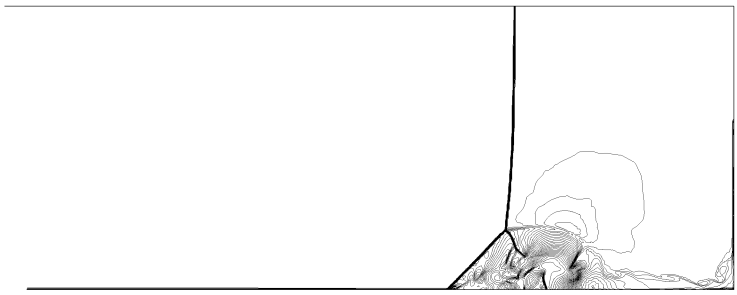
Figure 4.35. Validation of the Navier-Stokes solver. The theory of boundary layer states that for a flat wall $\frac{\tau_w}{\sqrt{u^3/xRe}}$ is 0.332. The numerical values along the flat wall are shown by circles. x is a distance mesured from the tip of the flat wall.

The flow speed behind the incident shock is very high, but the flow speed in the boundary layer is much lower. When the reflected shock propagates in this non-uniform flows, it goes faster in the boundary layer than in the internal region. A complex configuration then appears as shown in Fig. 4.36.

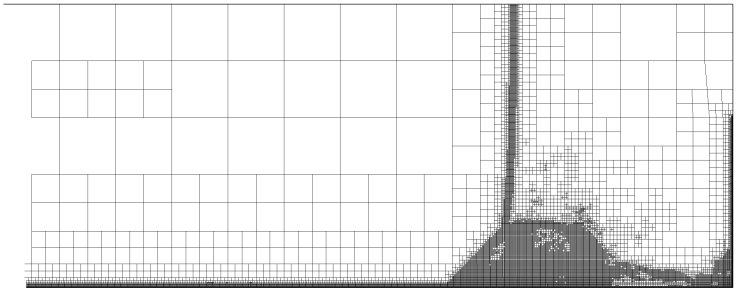
By comparing with experimental photos obtained in the Shock Wave Research Center, the agreement is however quite poor, most possibly due to the fact that the effects of real gases are not taken into account. Actually for the gas behind the incident shock wave the ratio of specific heats is far below 1.4. However, it agrees well with other numerical computations.

4.7.6 Shock wave moving in a nozzle

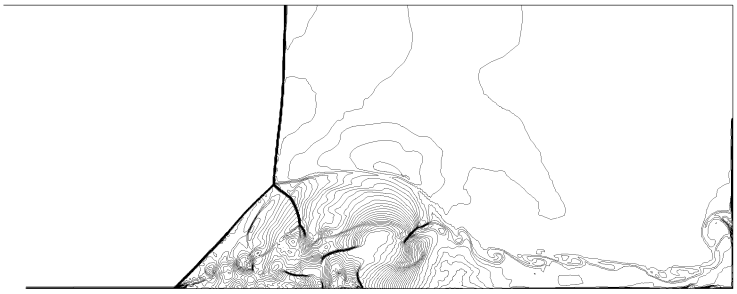
Shock wave motion in a nozzle is demonstrated in this subsection. Shock wave moving a nozzle is interested in some practical applications such as shock tunnel operation. The initial grid for the computation is shown in Fig. 4.6. The nozzle is 14cm long. Fig. 4.37 draws time sequences of isopycnics at every $50\mu s$. Fig. 4.37a is the instant when the incident shock wave arrives at the inlet of the nozzle. 50μ later, a part of the shock wave goes through the throat, and most is reflected by the wall, as shown in Fig. 4.37b. With the elapse of time, the shock wave propagating



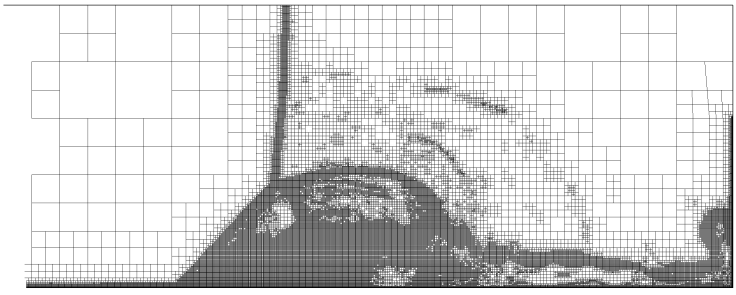
a.



b.



c.



d.

Figure 4.36. Shock / boundary layer interaction, $M_s = 5$, $T_w = 298k$; 6-level refinement, CFL = 0.5, $\alpha_{bl} = 1.4$: **a.b.** 32502 cells, 18832 steps, $t = 25\mu s$; **c.d.** 59205 cells, 25853 steps, $t = 50\mu s$.

through the throat becomes weaker and weaker due to the expanding nozzle, and a contact region is then formed behind it. The reflected shock wave is repeatedly reflected in the upstream part of the nozzle, and forms a multi-reflection pattern. These are observable in Fig. 4.37c. The multi-reflection shocks are attenuated by the "leak" from the throat, they cannot sustain for a long time and finally disappear, as shown in Fig. 4.37de.

Fig. 4.38 compares the experimental photo with the numerical results obtained by the Euler and the Navier-Stokes solver. Both solvers resolve the primary waves correctly. The main difference between two solvers is the secondary shock wave in the expanding part of the nozzle. In the viscous solution, the secondary shock wave interacts with the boundary layer and creates a separation bubble, as shown in Fig. 4.38c, which is not observable in inviscid flows as shown in Fig. 4.38b. Comparing with the secondary shock in the photo, the Navier-Stokes solution is clearly closer to the real gas flows. It suggests that the viscous effect in the nozzle flows is an important factor.

4.8 Remarks on three-dimensional extension

It has been shown in the previous sections that the VAS2D is a general-purposed vectorizable solver for two-dimensional applications. The solver consists of mesh generation, flow solver, mesh adaptation and post-processing procedures. To extend the solver to three dimensions, one needs to modify all procedures. The modifications are sometimes extremely complicated and difficult; time and work for conducting all modifications are certainly beyond what one doctoral student can afford in three years. This section will describe an efficient way to extend the VAS2D to vectorizable adaptive solver for three dimensions (VAS3D).

Any control volume in the VAS3D is a hexahedron, which is the 3-D counterpart of a quadrilateral. Automated hexahedral meshing is much more challenging than quadrilateral meshing. Fortunately many efforts have recently been made on hexahedral meshing (see, [5]). The post-processing of 3-D unstructured data may be performed by the commercial softwares which are installed in the supercomputer center of the Institute of Fluid Science. Therefore the key technique of the VAS3D is the extension of the flow solver and mesh adaptation.

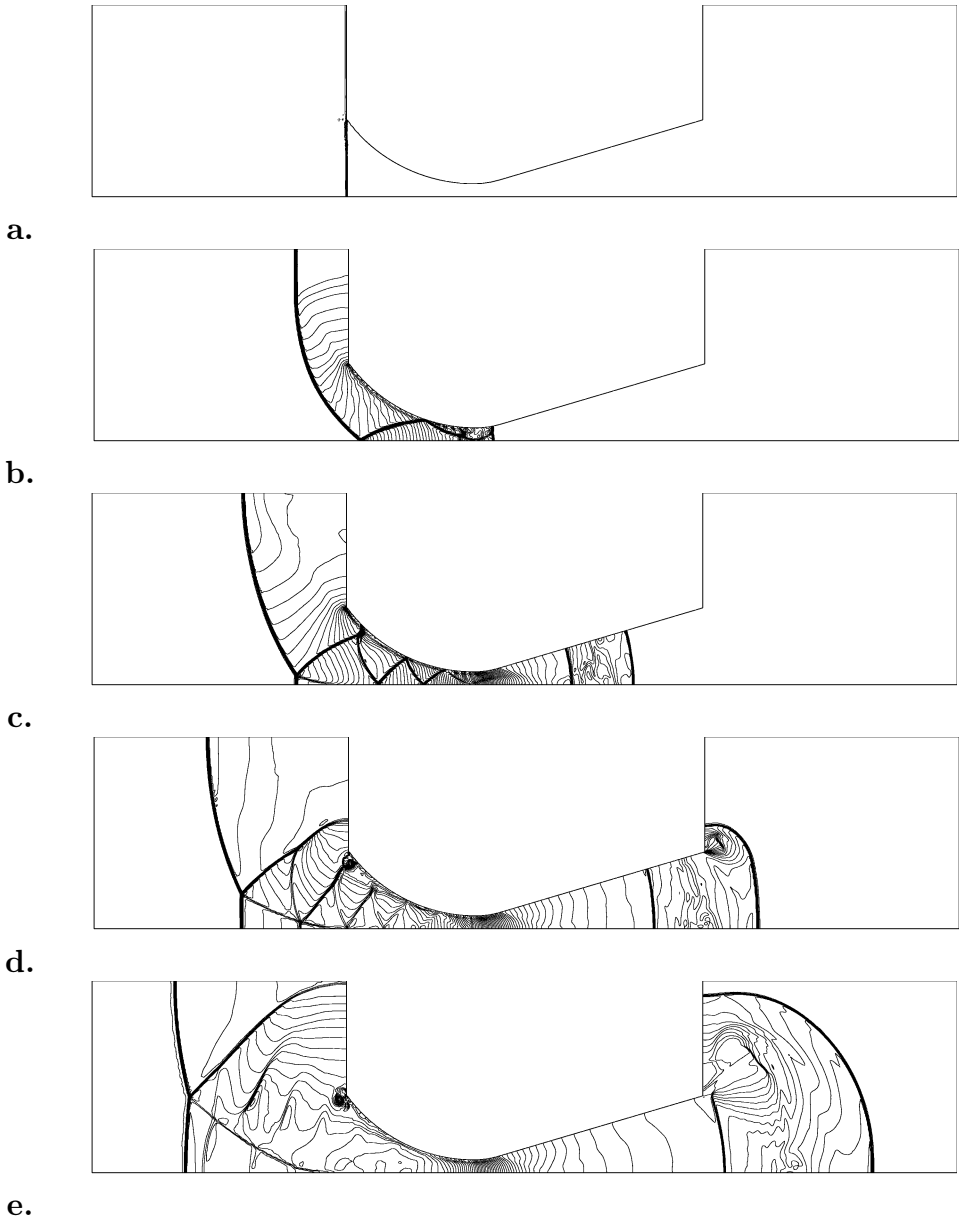
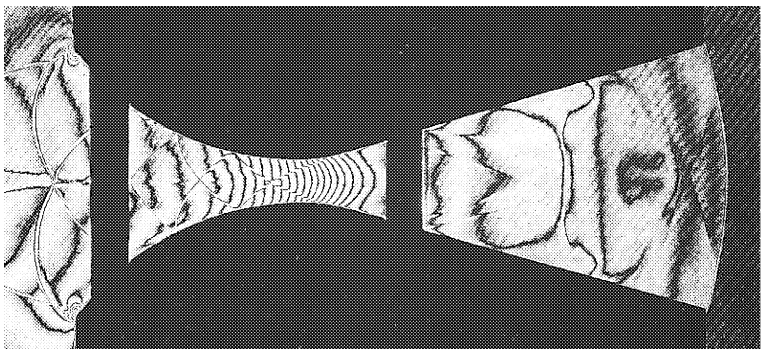
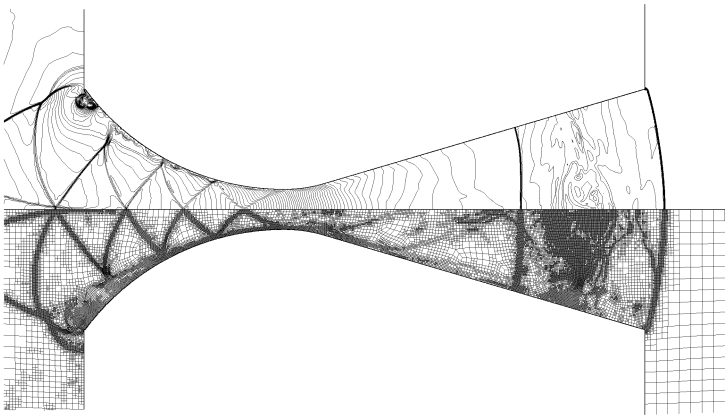


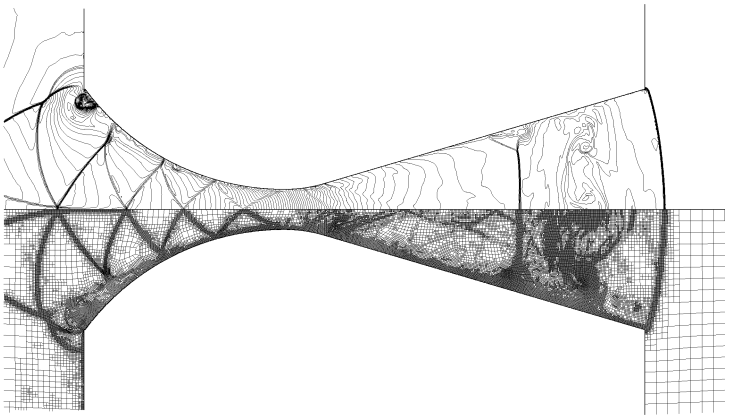
Figure 4.37. Shock wave motion in a nozzle for $M_s = 2.8$. CFL = 0.7, **e** consists of 36591 cells.



a.



b.



c.

Figure 4.38. Shock wave motion in a nozzle for $M_s = 2.8$: **a.** experimental photo obtained by holographic interferometry; **b.c.** numerical isopycnics and the corresponding grids obtained by the VAS2D. **b** is the result of the Euler solver, and **c** the Navier-Stokes solver.

The extension of the flow solver requires changes of the gradient evaluation and flux evaluation. The least square method discussed in Section 4.5.3 is actually suitable for three dimensions as well as two dimensions. The kernel part of the flux evaluation is the predictor step, which locally solves the two-dimensional Euler equations, and it can be done similarly by using three-dimensional Euler equations. The local gradient modification, which makes the stencil compact, has already been written in the vector form in Section 4.5.4, and so it is able to be applied to three dimensions without difficulty.

The extension of vectorizable mesh adaptation for a hexahedral grid is the most difficult part, but key problems have been solved and briefly described in Chapter 3. For instance, data structure is the cell-face one, and to label six neighboring cells follows the path which goes through all faces. The advantages of the cell-face structure for hexahedral are similar as the cell-edge structure for quadrilaterals, such as full vector processing, logic-free connectivity information for flow solver.

4.9 Summary

The smoothing Lax-Wendroff scheme, proposed in Chapter 2 has been applied to arbitrarily quadrilaterals by the finite volume method. Extensive numerical tests show that the scheme is indeed problem independent. The strengths of the approach may be summarized as follows. First, it is more general. Because no Riemann solver is involved in computation, the approach can be directly applied to the modeling of other systems of conservation laws. Second, it is a truly multidimensional method. The numerical flux is predicted by solving the multidimensional Euler equations without one-dimensional assumption there.

An essentially vectorizable data structure, which is one special case of those discussed in Chapter 3, has been realized on arbitrarily adaptive quadrilaterals. The data structure provides logic-free topological adjacency for the solver with negligible overheads. Its vectorization efficiency is comparable with that of a solver on structured meshes.

In solving the Navier-Stokes equations, unlike the hybrid methods which combines structured quadrilaterals around body with unstructured triangles outside, the present adaptation method adopts uniformly unstructured quadrilaterals. It neatly

generates a layer of body-fitted orthogonal mesh without losing generality, and it is a real bonus that quadrilateral mesh generation is readily automated.

References

[1] Venkatakrishnan V (1995) A perspective on unstrutured grid flow solvers, AIAA paper 95-0667.

[2] Aftosmis MJ, Gaitonde D, Sean Tavares T (1994) On the accuracy, stability, and monotonicity of various reconstruction algorithms of unstructured meshes, AIAA paper 94-0415.

[3] Nakahashi K, Egami E (1991) An automatic Euler solver using the unstructured upwind method, Computer and Fluids, 19 : 747-771.

[4] Malanthara A, Gerstle W (1997) Comparative study of unstructured meshes made of triangles and quadrilaterals, Proc. of 6th Intl Meshing Roundtable, Park City, Utah, USA.

[5] Proc. of 6th Intl Meshing Roundtable, Park City, Utah, USA, 1997.

[6] Hoffmann KA, Chiang ST (1993) Computational fluid dynamics for engineers, V1,V2, Publication of Engineering Education System^{TM}.

[7] Blacker TD, Stephenson MB (1991) Paving: a new approach to automated quadrilateral mesh generation, *Intl. J. for Numerical Methods in Engineering*, 32 : 811-847

[8] Zhu JZ, Zienkiewicz OC, Hinton E, Wu J (1991) A new approach to the development of automatic quadrilateral mesh generation, *Intl. J. for Numerical Methods in Enginearing*, 32 : 849-866.

[9] Kinney P (1997) Cleanup: improving quadrilateral finite element meshes, Proc. of 6th Intl Meshing Roundtable, Park City, Utah, USA.

[10] CF77 optimization guide, SG-3773 6.0, Cray Research, Inc. 1993.

[11] Zeeuw DD, Powell KG (1993) An adaptively refined Cartesian mesh solver for the Euler equations, *J. of Comput. Phys.* 104 : 56-68.

[12] Zienkiewicz OC, Zhu JZ (1991) Adaptivity and mesh generation, *Intl. J. for Numerical Methods in Enginearing* 32 : 783-810.

[13] Roe PL (1991) Modern shock-capturing schemes, In: K. Takayama (eds), Proc 18 th ISSW, Sendai, Springer: pp. 29-40.

[14] Zeeuw DD (1993) A quadtree-based adaptively-refined Cartesian-grid algorithm for solution of the Euler equations, Ph.D. thesis, The University of Michigan.

[15] Batina JT, (1989) Unsteady Euler airfoil solutions using unstructured dynamic meshes, AIAA paper 89-0115.

[16] Hase JE, Anderson DA and Parpia I, (1991) A Delaunay triangulation method and Euler solver for bodies in relative motion, AIAA Paper 91-1590.

[17] Singh KP, Newman JC and Baysal O, (1995) Dynamic unstructured method for flows past multiple objects in relative motion, AIAA J. 33, No.4.

[18] Trepainier JY, Reggio M, Paraschivoiu M and Camarero R, (1993) Unsteady Euler solutions for arbitrarily moving bodies and boundaries, AIAA J. 31, No. 10 : 1869-1876.

[19] White FM (1974) Viscous fluid flow, McGraw-Hill, Inc.

[20] Jeng YN, Chen JL (1992) Truncation error analysis of the finite volume method for a model steady convective equation, *J. of Comput. Phys.* 100 : 64-76.

[21] Poinsot TJ, Lele SK (1992) Boundary conditions for direct simulations of compressible viscous flows, *J. of Comp. Phys.* 101 : 104-129.

[22] Hirsch C. (1990) Numerical computation of internal and external flows, V1,V2, John Wiley & Sons.

[23] Giraud L, Manzini G (1996) Parallel implementations of 2D explicit Euler solvers, *J. of Comput. Phys.* 123 : 111-118.

[24] Kallinderis Y, Vidwans A (1994) Generic parallel adaptive-grid Navier-Stokes algorithm, AIAA J. 32 : 54-61.

[25] Venkatakrishnan V, Simon H, Barth TJ (1991) A MIMD implementation of a parallel Euler solver for unstructured grids, RNR-91-024, NASA Ames Research Center.

[26] Voinovich PA (1993) Two-dimensional locally adaptive unstructured unsteady Euler code, Advanced Technology Center, St. Petersburg, Russia (unpublished).

[27] Takayama K, Inoue O (1991) Shock wave diffraction over a 90 degree sharp corner, *Shock Waves* 1 : 301-312.

[28] Skews BW (1967) The perturbed region behind a diffracting shock wave, *J. Fluid Mech.* 29 : 705-719.

[29] Bazhenova TV, Gvozdeva LG, Zhilin YV (1979) Change in the shape of the diffracting shock wave at a convex corner, Acta Astronautica. 6 : 401-412.

[30] Hillier R (1991) Computation of shock wave diffraction at a ninety degree convex edge, *Shock Waves* 1: 89-98.

[31] Sun M, Takayama K (1997) The formation of a secondary shock wave behind a shock wave diffracting at a convex corner, *Shock Waves* 7:287-295.

[32] Fursenko AA, Sharov DM, Timofeev EV and Voinovich PA (1992) Numerical simulation of shock wave interactions with channel bends and gas nonuniformities, *Computers Fluids*, 21 : 377-396.

[33] Engquist B, Lotstedt P and Sjogreen B (1989) Nonlinear filters for efficient shock computation, *Mathematics of computation*, 52 : 509-537.

[34] Ofengeim DKh, Drikakis D (1997) Simulation of blast wave propagation over a cylinder, *Shock Waves*, 7: 305-317.

[35] Loitsyanskii LG (1966) Mechanics of liquids and gases, Pergamon Press.

Chapter 5

Experimental and numerical study of shock wave focusing

5.1 Introduction

A shock wave the shape of which is concave to the direction of its propagation can converge to a focal region and will produce locally a high pressure and temperature there. This shock wave focusing has been applied to various scientific and engineering applications. Guderley [1] was the first to derive the self-similar solution for symmetrically converging cylindrical and spherical shock waves and showed that the pressure at the center of convergence becomes infinite. Sturtevant and Kulkarney [2] pointed out that, in the case of the focusing of a reflected shock wave from concave wall, the ratio of the pressure amplification of a sonic wave is infinite at focal point. It seems that the focusing of a concave converging shock wave is one of the most effective ways in creating a high pressure and temperature.

Guderley's theoretical work of the convergence of cylindrical shock waves has been experimentally examined by Watanabe and Takayama [3]. Milton and Archer [4] proposed the introduction of a duct having a logarithmic spiral configuration in order to achieve the focusing of a planar shock wave. Milton [5] reviewed the focusing of a planar shock wave in shock tubes. Very high pressures and temperatures, however, have never been obtained by the focusing of a planar shock wave reflected from concave walls. Sturtevant and Kulkarny [2], based on shadowgraph observation of the reflected shock wave focusing, conjectured that diffracted expansion

waves created from the edge of the concave reflector corners overtake the converging shock wave and hence reduce the pressure amplification at the focal point. Geometrical shock dynamics explains, in the other hand, that nonlinear effects will make the shape of a shock wave flat so that the peak pressure is somewhat restricted. This is a multi-dimensional effect of shock wave focusing, which is deviated from a symmetrical convergence of shock waves as discussed by Guderley [1].

Milton [5] found that, in the real procedure of planar shock wave focusing along the logarithmic spiral duct, the triple point appeared at the final stage of the focusing. This implies that the nonlinear character of focusing shock wave is dominating at the final stage. Watanabe and Takayama [3] investigated the stability of converging cylindrical shock waves and found that the small initial irregularity in the shape of a shock wave developed eventually creating Mach reflections which accompanied triple points and vortices and smaller the mode number of shock wave irregularities is, more significant the final distortion of the shock wave is. These observations indicate that the contribution of the nonlinear evolution of a shock wave plays an intriguing role in the shock wave focusing.

In the earlier holographic interferometric study of shock wave focusing from a circular concave wall (Takayama and Ben-Dor [6]), it is revealed that the shock wave focusing is strongly affected by the transition of the shock wave reflection from Mach to inverse Mach reflection over the curved wall and by the motion of the resulting triple points. Izumi and Nishida [7] also observed this phenomenon over parabolic shaped reflectors. In general, the detailed understanding of the effect of shock wave reflection on the formation of focusing is, unfortunately, not yet completed experimentally and numerically.

The present study explores the final stage of the shock wave focusing inside a reflector, which has a similar circular reflector as that used by Takayama and Ben-Dor [6], connected to a diaphragmless shock tube of 60 mm in height and 150 mm in width. Observations were carried out mostly by double exposure holographic interferometry. Attention was focused on the motion of the triple points which were created by the termination of the inverse Mach reflections.

It was found that, for building up the high pressure and temperature at the focal point, the collision of the two triple points which were created by the inverse

Mach reflection and developed with propagation along the curved wall has a more dominating effect than the convergence of the curved reflected shock wave which is predicted by acoustics or geometrical shock dynamics. The triple points always collide with each other before the curved shock wave coalesces into the geometrical focal point. The result of the quantitative analysis of these interferograms confirmed that the peak density was a maximum immediately after the time instant of the shock wave focusing and the location in which the peak density appeared was slightly deviated from the geometrical focus. This trend was also verified by a finite volume numerical simulation using adaptive unstructured grids.

5.2 Experiment and computation

5.2.1 Experimental setup

Experiments were conducted using a 60 mm $\times$ 150 mm diaphragmless shock tube in the Shock Wave Research Center, Institute of Fluid Science, Tohoku University (Yang and Takayama, [8]). Fig. 5.1 shows the shock tube consisting of a driven channel of 60 mm $\times$ 150 mm, and a driver chamber of 290 mm i.d. and 2 m in length. Upon the quick movement of a rubber membrane that seals a driver chamber and a driven channel, shock waves were readily generated in the driven channel. The movement of the rubber membrane could be controlled so accurately that the generated shock waves became very repeatable under identical initial conditions. Therefore, the scatter of shock wave Mach numbers was found to be within $\pm 0.25\%$ for more than 500 runs (Yang and Takayama [8]). This repeatability enabled to record shock wave configurations at the focal point sequentially and much more accurately. The diaphragmless shock tube allowed very repeatable observations of the flow field near the focal point at every 10, 20 or 25 microsecond time interval for incident shock Mach numbers $M_s = 1.03$, 1.09, 1.28, 1.50, and 1.74 in Nitrogen.

Fig. 5.2 shows the geometry of the test section, which had a circular shape of 387 mm i.d. and 150 mm in width, and was connected to the shock tube. The observation windows were made from an acrylic plate of approximately 40 mm in thickness. Due to double exposure holographic interferometric observation, thick plastic glass windows were still acceptable for quantitative observation of density. For creating weak shock waves of $M_s = 1.03$ and 1.09, the initial pressure in the driver channel was adjusted slightly higher than atmospheric pressure, since the

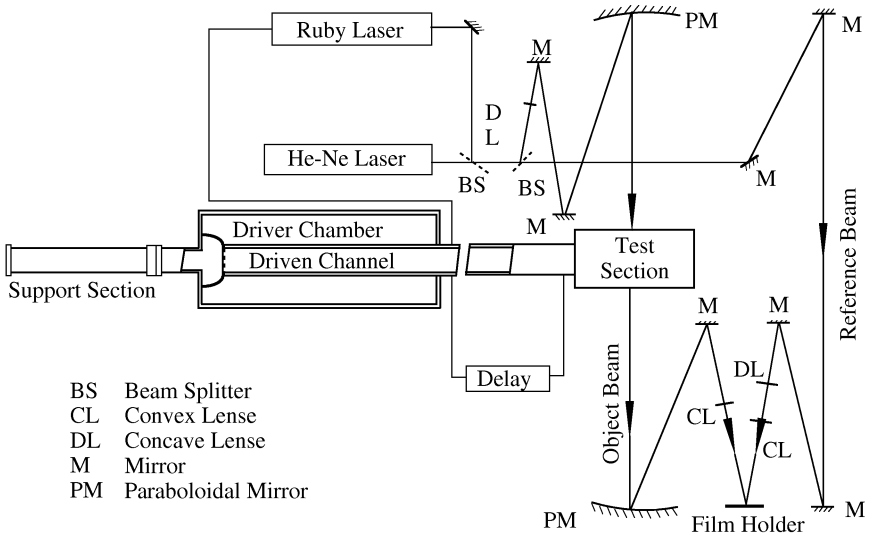


Figure 5.1. Experimental setup and optical arrangement

possible resolution of interferometric fringes is proportional to the initial pressure or density. This enhancement of the initial pressure was only limited because of the large diameter of the plastic windows. On the other hand, in order to create very weak shock wave but with possibly larger density jump across them, it is advantageous to keep the ratio of pressures between the driver chamber and driven channel closer to unity, but to make their difference large. Room temperature is at about $25 \pm 2^{\circ}C$ during all experiments. Most experimental results were summarized in [9], and only the results of $M_s = 1.5$ are discussed and compared with numerical results in this chapter.

Fig. 5.1 also shows the schematic diagram of double exposure holographic interferometric arrangement. Paraboloidal schlieren mirrors of 500 mm dia. and 5 m focal length was used to collimate the object beam so that this optical arrangement is similar to a conventional shadowgraph optic except that the reference light beam was added to it. A double pulse ruby laser (Apollo Laser Inc. 22DH, 25 ns pulse duration and 2 J/pulse) was used as light source. The light path length of the test section was 150 mm so that the sensitivity is increased by a factor 2.5 to the test section of a shock tube with the light path length of 60 mm (Yang and Takayama

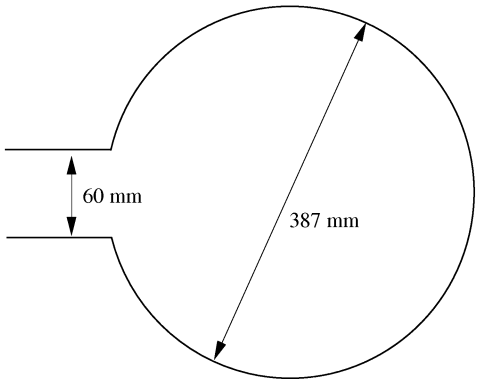


Figure 5.2. A 150 mm wide circular reflector connected to the 60 mm × 150 mm shock tube

[8]). The object and reference beams were superimposed on a 100 mm × 125 mm sheet film (Agfa Gevaert 10E75). Holograms were reconstructed by an Argon-Ion laser (wavelength $\lambda = 514.5$ nm and 1 W). The distortion of images between the experiment and reconstruction was negligibly small.

5.2.2 Numerical methods

At the final stage of shock wave focusing, shock waves and slip lines change their configurations not only in very short time duration but also in a very confined focal region. Therefore, for simulating these shock wave interactions, a sophisticated numerical scheme is needed to achieve such high spatial and temporal resolutions. Although it is very straightforward distributing fine meshes everywhere in the computational zone, this can not be very efficient.

The Euler equations were numerically solved on an adaptive quadrilateral grid by the VAS2D, which uses the Lax-Wendroff scheme with a smoothing flux and efficiently removes numerical oscillations (see Chapter 2 for details). The initial quadrilateral grid is shown in Fig. 5.3. Only half of the test section is calcualted because of symmetry. Another code is also chosen in this study for comparison. The code uses a triangular grid, which was originally constructed by Voinovich (1993). The scheme is a second-order Godunov-type one using a TVD limiter and the exact Riemann solver.

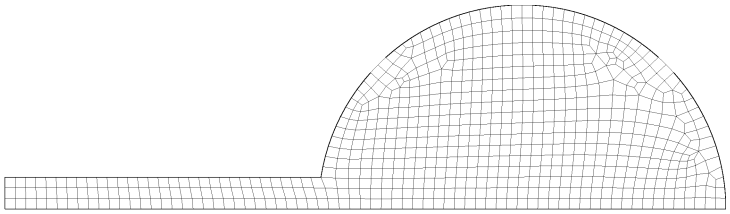


Figure 5.3. The initial quadrilateral grid for the VAS2D.

5.2.3 Quantitative image data analyses of interferograms

The shock wave focusing in a shock tube was usually visualized by using shadowgraph or schlieren method and the pressure at the focal regions were well measured (Sturtevant and Kulkarny [2]). However, precise sequential optical flow visualizations were found to be difficult so far in a conventional shock tube. On the other hand, pressure measurements can provide only the flow information at individual spots. In addition to this, the pressure transducers fail, due to their finite size, to respond a very steep pressure rise at a shock wave. Therefore, the time variation of pressures at a very narrow focal region has never be resolved properly.

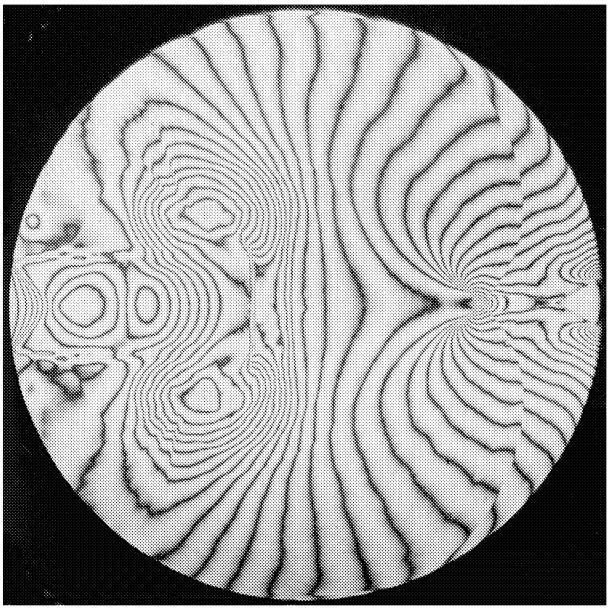
Holographic interferometry can measure quantitatively and non-invasively high-speed flows. In two-dimensional flows and if the real gas effect is neglected, the fringes on interferograms correspond to isopycnics so that this technique is useful to determine the structure of the focusing shock wave. However, it is not straightforward to evaluate the density jump at the shock waves, because fringes there are so densely packed, sometimes beyond the limit of photographic resolution, that the collimated object beam was often refracted and the image of the fringes at the shock wave appear to be blurred or even gray. There is no difficulty in resolving fringe spacing in a region where densities vary continuously. A possible way of evaluating the density jump at the shock wave is to adopt the result of reliable numerical simulations.

Sun and Takayama [10] tried to combine the quantitative density distribution estimated by simply counting fringes on interferograms with a numerical result. The comparison between an interferogram and its corresponding numerical result of the present adaptive unstructured Eulerian solver is shown in Fig. 5.4. The interferogram was taken for $M_{\rm s} = 1.502 \pm 0.002$ at $t = 1114 \pm 8\ \mu s$ where the elapsed

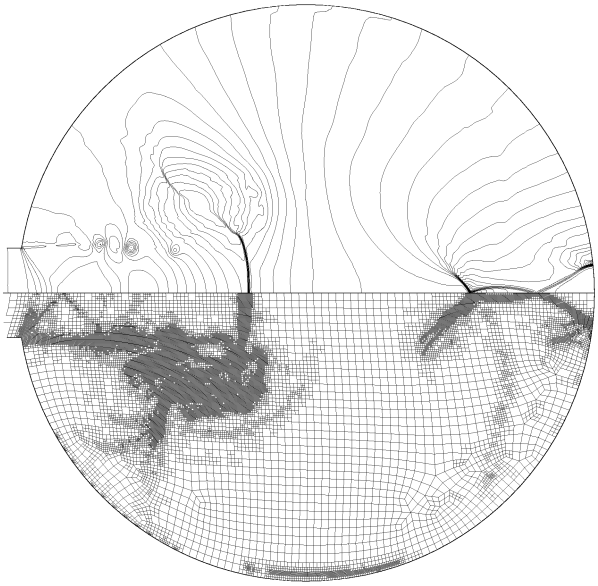
time t was measured from when the incident shock wave entered the test section. The scatter of the elapse of time was due to a slightly inaccurate measurement the time instant when a shock wave exactly entered the test section. This was attributable to the finite size of the pressure transducers.

At the time instant when the shock wave focusing occurs, four shock waves and two slipstreams intersect simultaneously at one point $x = 0.786$, where x is a dimensionless distance normalized by the reflector diameter so that the center of the reflector is at $x = 0.5$. These complex wave patterns are identified exactly by corresponding numerical results. As seen in Fig. 5.4b, the same wave pattern is observable in the numerical simulation at t = 1113 μs and $x = 0.782$. The upper half of Fig. 5.4b shows the isopycnics and lower half does the corresponding grid for $M_{\mathrm{s}} = 1.50$. No compensation was sought for the possible phase shift between the isopycnics and the fringes on the interferogram. The observable discrepancies of fringes are estimated to be less than those created during measurements. The spatial and temporal agreement of fringes between the interferogram and the numerical result reveals that the VAS2D result is accurate enough to describe not only the shape of shock waves but also their strength. It must be emphasized, although it is not a main topic of the present study, that the interaction between vortexes generated from the entrance of the reflector and the secondary shock wave is very sharply resolved.

Taking into account this agreement, the quantitative interpretation of interferograms can be done readily by using these numerical results. The value of the density jump at the shock waves is given numerically, and hence the density in other parts are measured simply by counting the fringe number on the interferogram since the jump of the fringe number on the interferogram has a known value (see [9]). The density distribution along the centerline is later summarized in Fig. 5.8b. Very good agreement is obtained over the entire flow field except the flow region which is likely to be affected by the real gas effect. This quantitative analysis is successfully applied to the density measurement near the focal region for $M_{\mathrm{s}} = 1.5$.



a.



b.

Figure 5.4. Comparison between an interferogram and the corresponding numerical results of the VAS2D: (a) interferogram for $M_s = 1.502 \pm 0.002$ and at $t = 1114 \pm 8 \mu s$; and (b) numerical isopycnics (upper half) and the corresponding adaptive grid (lower half) for $M_s = 1.5$ and at $t = 1113$ μs.

5.3 Formation of reflected shock wave

A planar shock wave is gradually curved if propagating in a non-uniform medium or moving along a non-uniform boundary. The non-uniformity simply means the inhomogeneity of the medium such as a medium having a temperature gradient or a stratified layer. If a cylindrical shock wave converges toward its center of convergence, it is gradually perturbed with propagation, although it appears to be cylindrical and has apparently no disturbances behind it. Watanabe and Takayama [3] experimentally proved that these initial perturbations developed to form the irregularity of the shape of the shock wave, which eventually became Mach reflection while moving very close to the center of convergence. Evidently, the shape of curved reflected shock wave is a dominating factor in shock wave focusing. This section describes, therefore, the process of the deformation of curved shock waves from the circular reflector.

The formation of a curved shock wave which was discharged from an inlet to the circular reflector is shown in Fig. 5.5 for $M_s = 1.5$ in air. The initially planar shock wave diffracts at the exit of the shock tubes, resulting in a curved shock wave, the foot of which is perpendicular to the wall as seen in Fig. 5.5a. The foot of this curved shock wave tends to lean forward with its propagation and eventually becomes Mach reflection. With further propagation of the shock wave, as seen in Fig. 5.5b, the Mach reflection transits to the regular reflection. The triple point of the Mach reflection is succeeded by a second triple point on the reflected shock wave of the regular reflection. The transition was well documented as that from the direct Mach reflection to the inverse Mach reflection by Courant and Friedrichs [11] and was well demonstrated by Takayama and Ben-Dor [6]. A full developed regular reflection is shown in Fig. 5.5c.

The second triple point consists of three shock waves and a slip line which successively develops to a vortex on the wall. The planar incident shock wave reflects from the curved wall resulting in a reflected shock wave which is concave toward the direction of its propagation. Simultaneously the normal shock wave and the slip line grow with the elapse of time. These waves are clearly shown in Fig. 5.5d. It is, therefore, this wave system that contributes to the shock wave focusing.

The process of the formation of a curved reflected shock waves is observed for

other shock wave Mach numbers and the trend of the process is similar to other M_s. The same conclusion was derived experimentally by Takayama and Ben-Gor [6] and also by Izumi and Nishida [7] who tested with paraboloidal reflectors. The present observation as seen in Fig. 5.5 and also previous observations showed that the curved reflected shock wave bounded by the two triple points which were created along the upper and lower walls and were moving toward the center line. When the curved reflected shock wave coalesced at the centerline, the triple points simultaneously merge.

Counting the contour numbers in Fig. 5.5d, the density behind the triple points is found to be higher than that behind reflected shock wave. The higher density is maintained until these two triple points collide with each other. For stronger shock waves, it will be shown that the highest density that occurs immediately after the collision at the focal point can be estimated from the value of the density behind the triple points. The formation procedure of reflected shock wave agrees very well with experimental observations [9].

5.4 Shock wave focusing and collision of triple points

Sequential interferograms and numerical results, particularly enlarged view of the focal point, are shown in Fig. 5.6 for $M_s = 1.50$. Over a concave wall, the transition of the pattern of a reflected shock wave from a direct Mach reflection to an inverse Mach reflection takes place. The two reflection points of these resulting regular reflections which consist of the regular reflection and a triple point on its reflected shock wave merge forming a concave shaped reflected shock wave.

Fig. 5.6a shows the interferogram at -25 μs from the moment of the shock wave focusing. The triple points move toward the centerline. The normal shock waves at the second triple points were gradually curved with their propagation and crossed. The triple points collided each other in Fig. 5.6c completing the shock wave focusing. At 10 μs after the focusing, the slip lines coalesced, resulting in a very steep increase of density at the focal point. The curved slip line merged and appeared to be a loop in Fig. 5.6e. The curved shock waves crossed each other forming another three shock wave confluence. Numerical results shown in Fig. 5.6bdf exactly reproduce the shock wave configurations during focusing.

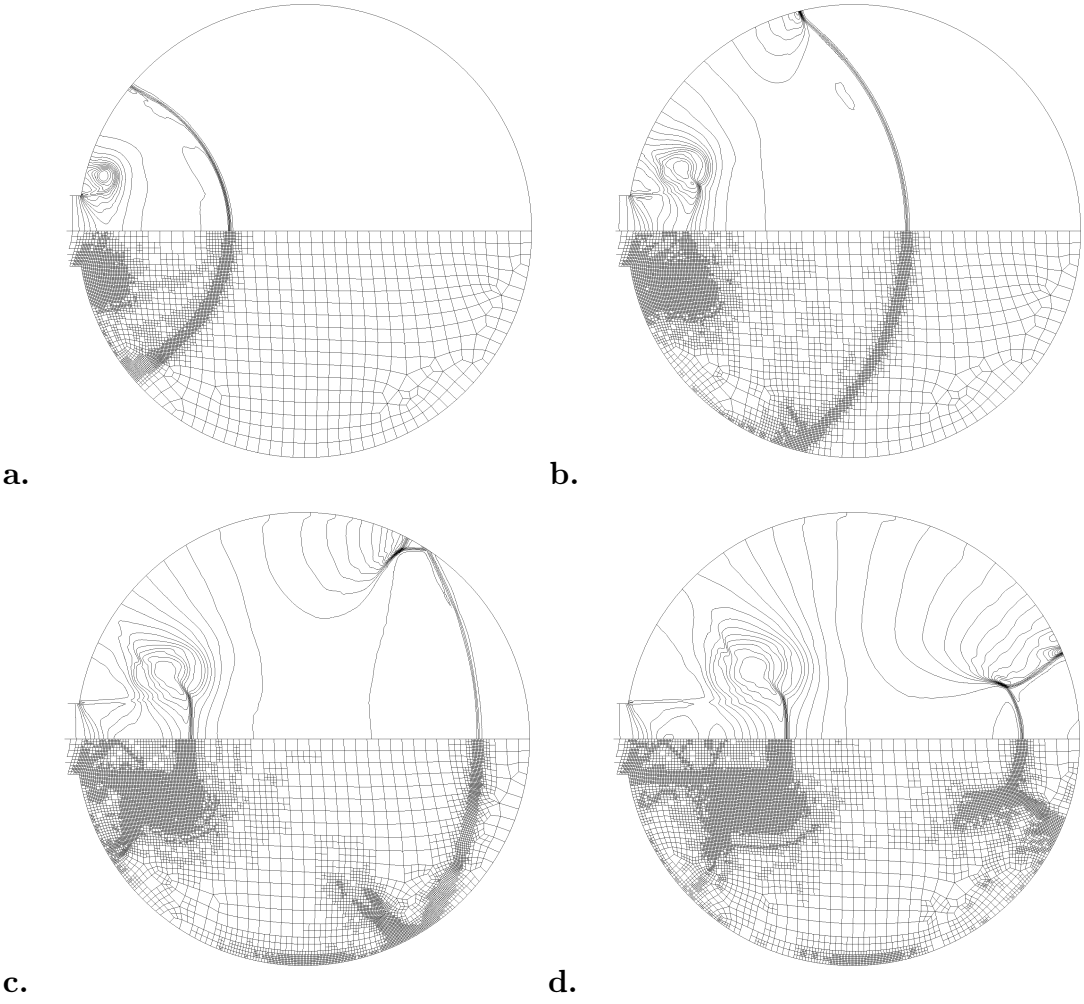


Figure 5.5. Sequential numerical isopycnics and adaptive grids of shock wave propagation in the circular reflector for M_s=1.5.

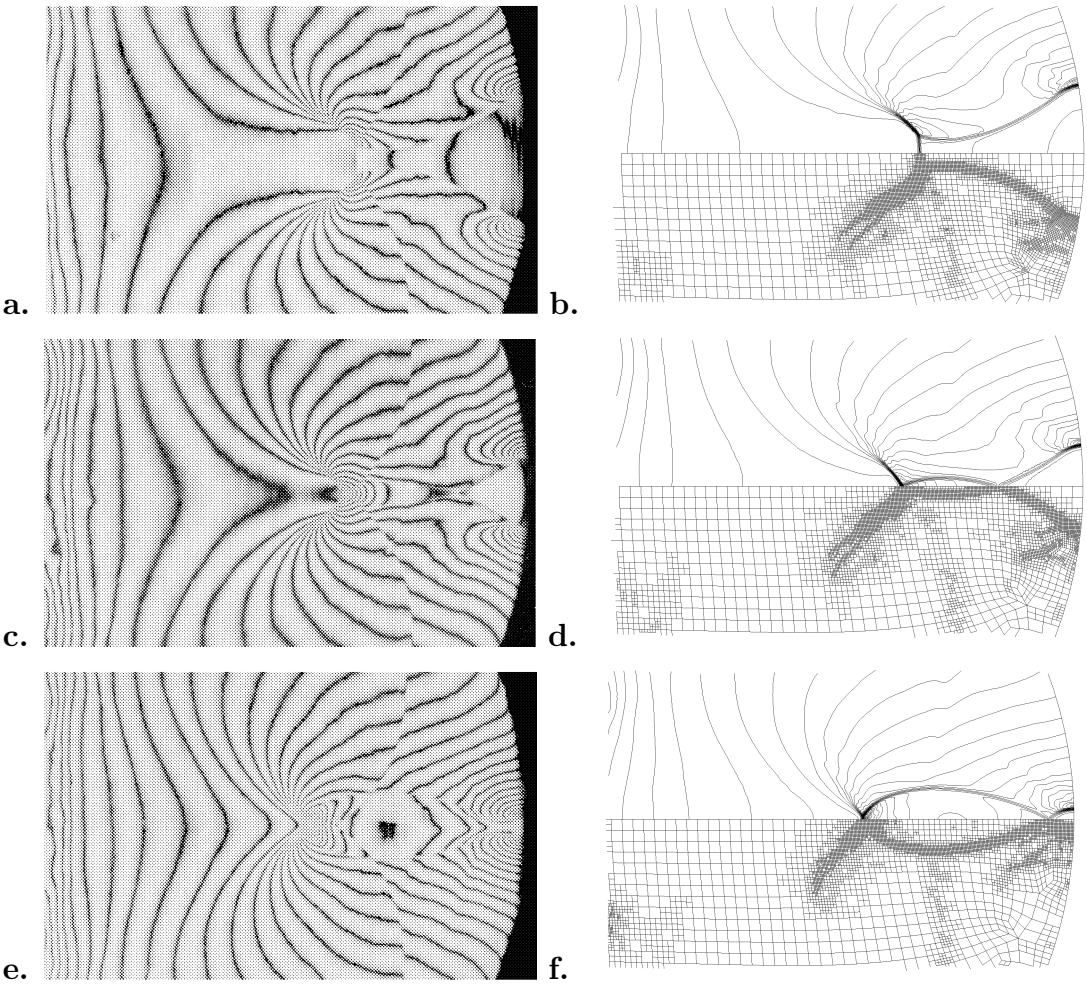


Figure 5.6. Shock wave focusing for $M_s = 1.50$, $P_0 = 26.7$ kPa: **a.b.** -25 μs; **c.d.** 0 μs; **e.f.** 25 μs : **a.c.e.** are experimental interferomgrams, and **b.d.f.** are isopycnics and corresponding adaptive grids obained by the VAS2D

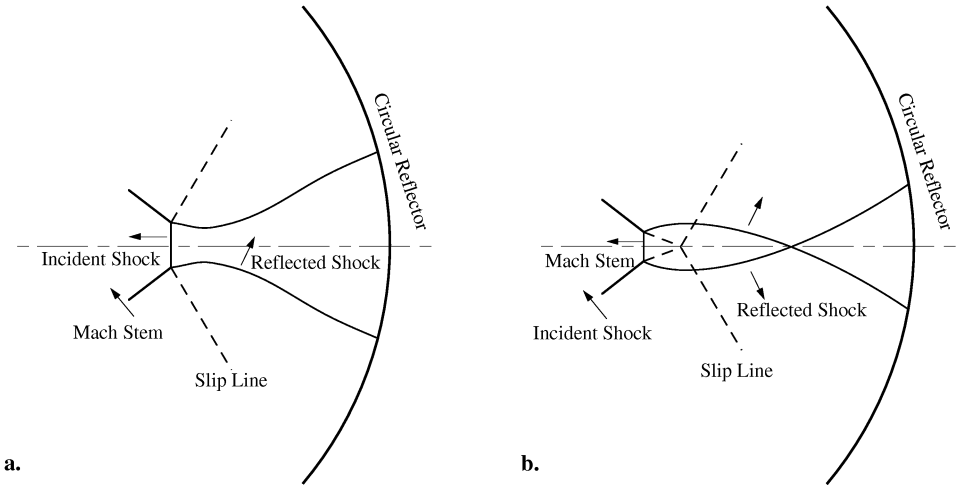


Figure 5.7. Sketch of the shock wave system in Fig. 5.6ae, corresponding to Fig. 5.7a and b respectively.

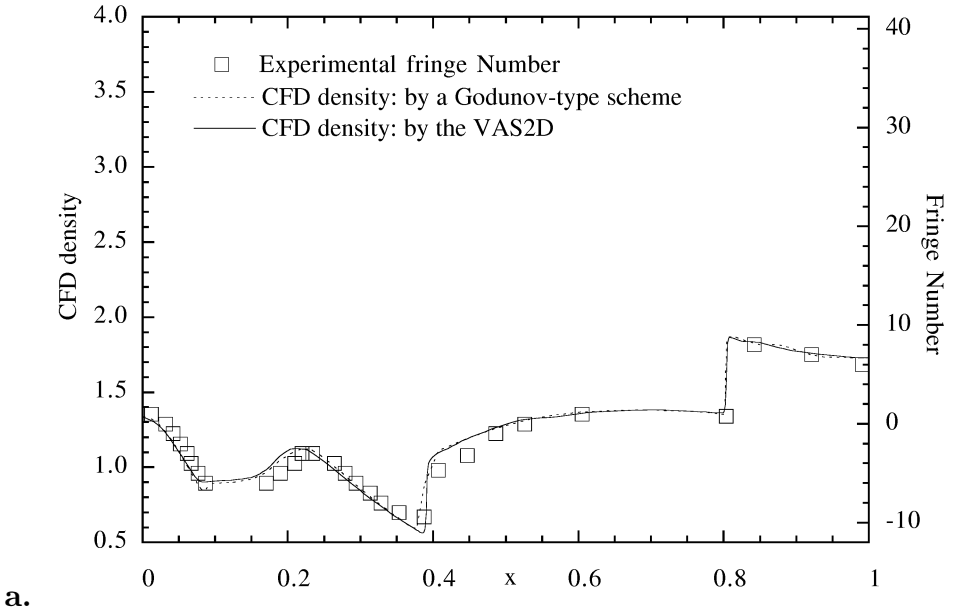
Fig. 5.7 illustrates the renaming the shock wave system in the vicinity of the focal point: the sketches of Fig. 5.6ae are illustrated in Fig. 5.7a and b respectively. At the triple point in Fig. 5.7a, the incident shock wave, the reflected shock wave, the Mach stem, and the slip line are defined so far according to the understanding of the shock wave dynamics. However, as the slip line already overtook the curved shock wave in Fig. 5.7b, the incident shock wave in Fig. 5.7b is replaced by the Mach stem in Fig. 5.7a and the Mach stem in Fig. 5.7b is also replaced by the incident shock wave in Fig. 5.7a. The reflected shock wave and the slip line remain unchanged. Eventually the shape of these wave systems appear to be a loop shape as seen in Fig. 5.6e. The reflected compression waves from the curved boundary coalesced into a shock wave eventually forming the reflected shock wave. This coalescence of fringes into a reflected shock wave is clearly observable in Fig. 5.6. The loop shaped shock wave interacted at first with these reflected waves, which are now renamed as Mach stem, and generated triple points. However, with propagation, these reflected shock waves no longer remain as shock waves but become compression waves. Hence, the triple points vanish and the loop shaped shock wave becomes smoothly rounded.

The density enhancement behind converging shock waves is confirmed in experiment. Only one or two fringes are found to be different from the instant $-50\mu s$ to $-25\mu s$. A quantitative measurement shows that the fringe number difference is about 16 immediately after focusing for $M_s = 1.5$. The focusing of a curved shock wave never increases the fringe number. Detailed experiment indicates that the fringe number is increased due to the interaction of the Mach stems. The fringes are densely packet immediately after the interaction of the Mach stem. The high density region moves toward the point at which the reflected shock waves merge as shown in Fig. 5.6c.

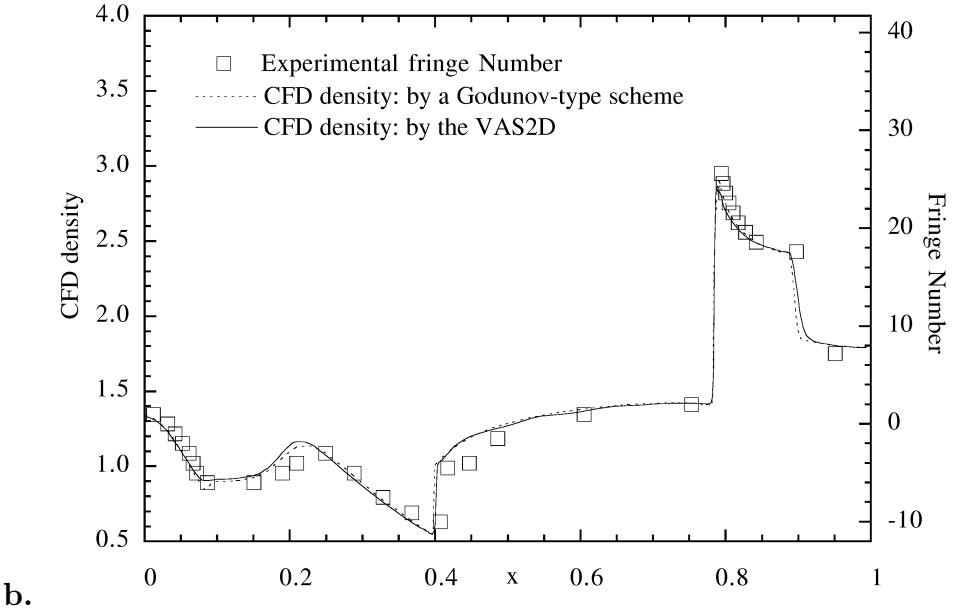
Fringe numbers on the interferograms in Fig. 5.6ace are counted along the centerline. The comparison with these experimental results with numerical results is shown in Fig. 5.8ab, and Fig. 5.9c: open squares denote the experimental data; and solid and dashed lines denote the numerical results of two different methods. Since the fringes behind the loop shaped shock wave is so densely concentrated that the corresponding density gradient deteriorates the resolution of the fringe spacing. The peak density seen in Fig. 5.9c, therefore, should be obtained by the extrapolation. Good agreement is obtained between experiments and numerical results. The density increase of approximately 50% at 25 μs is observed between Fig. 5.8a and Fig. 5.8b. The interaction of these two incident shock waves further increase of the peak density as shown in Fig. 5.9c. The density increase is more clearly shown in Fig. 5.9d by plotting three density distributions together.

It is seen that in three quantitative comparisons with experiments, the numerical results obtained by the VAS2D agree well with these by the Godunov-type upwind scheme. This suggests that the VAS2D can provide numerical results for complex shocked flows as accurate as the upwind scheme.

Mach stems and slip lines are observable in Fig. 5.6e. The shape of the reflected shock wave after the collision of triple points is almost self-similar. This, based on the wave pattern at the focal point, makes it possible to estimate the highest density near the focal region. Hence, the shock wave system in Fig. 5.6c is equivalent to the shock wave reflecting over a wedge with inclination angle of 35°. The density ratio normalized by the initial density is 1.42 in front of the reverse moving shock wave. This value is obtained by the VAS2D. The fringe jump corresponding to the density

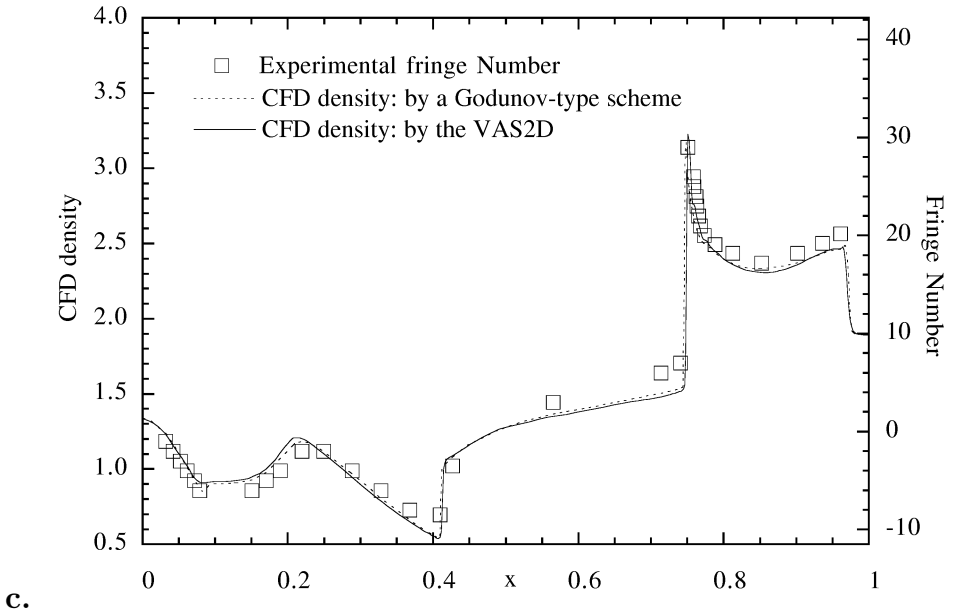


a.

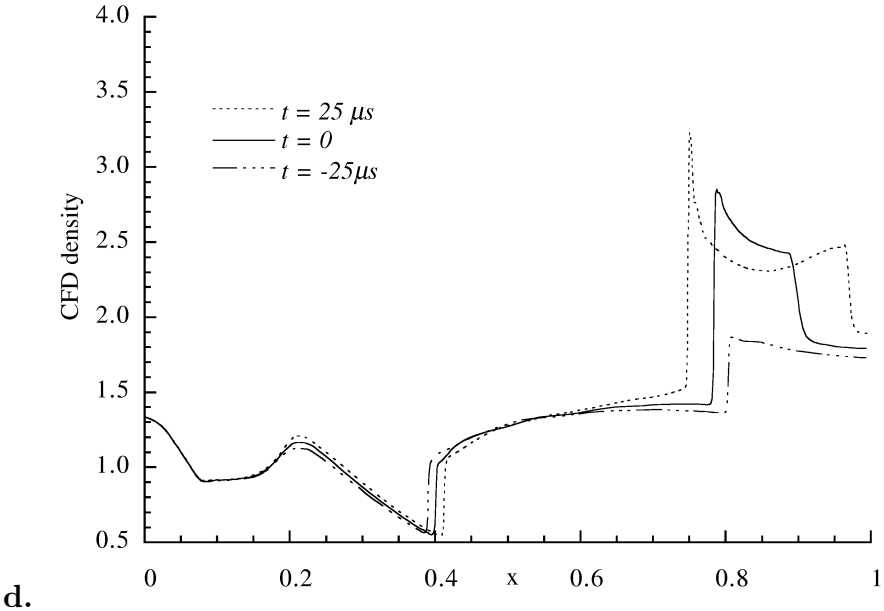


b.

Figure 5.8. Comparison of density distributions along the centerline: **a.** $t = -25\mu s$, corresponding to Fig. 5.6a; **b.** $t = 0$, corresponding to Fig. 5.6b.



c.



d.

Figure 5.9. Comparison of density distributions along the centerline (Continued): **c.** $t = 25\mu s$, corresponding to Fig. 5.6c; **d.** numerical density distributions at the three time instants.

Table 5.1. Estimation of triple point trajectory and highest pressure and density after the collision of the triple point by the geometrical shock dynamics theory

	theory	interferogram	CFD
χ	5.1°	5°	
P_{max}	6.31		6.1
ρ_{max}	3.47		3.3

jump across the incident shock waves is 16, and 1.73 in terms of the density ratio, which was obtained by counting fringes on the interferogram. Thus the incident shock wave Mach number $M_{\rm s}$ is eventually determined to be 1.425.

In the case of the reflection of a shock wave of $M_{\rm s} = 1.425$ over a wedge of inclination angle of 35°, the reflection pattern is Mach reflection and its physical quantities are tabulated in Table 5.1. The triple point trajectory was solved by geometrical shock wave dynamic theory [12]. The theory can well predict experimental results. Quantitative agreement of these quantities indicates that higher density or pressure is mainly caused by the interaction of shock waves.

The collision of the shock wave is assumed to be the same as the termination of a triple point that occurs in the inverse Mach reflection. Let call the regular reflection which appears when the inverse Mach reflection terminates as a transitional regular reflection (TRR) consisting basically of RR followed by MR (see Ben-Dor and Takayama [13]). Then the wave configuration that appears after the collision is regarded as TRR. The RR which was first mentioned by Sturtevant and Kulkarny [2] is indeed TRR. The wave configuration that appears in the present experiment after the collision of triple points is MR. This Mach reflection appears when the inverse Mach reflection InMR terminates so that this may be called a transitional Mach reflection, TrMR. The shape of the transition to RR or MR after the shock wave focusing belongs to either TRR or TrMR.

5.5 Summary

Widening the width of the test section of the diaphragmless shock tube very reliable interferograms were obtained with a high degree of repeatability. The adaptive unstructured grid numerical simulation was conducted in order to interpret the interferograms quantitatively. The results obtained are summarized as follows:

1. Apparent shock wave focusing is created by the collision of triple points. However, the peak density appears at the time instant after the triple points merge at the center line.

2. The process of focusing of reflected shock waves from circular concave wall is governed by the entire process of shock wave systems moving over the circular concave wall.

3. The numerical results obtained by the VAS2D on unstructured quadrilateral grids agree very well with the present finite fringe interferograms. The image data processing partially supported by the numerical simulation has been successfully applied to the shock tube experiment, in which very delicate shock wave interactions were well resolved.

4. A transitional Mach reflection may appear after the termination of an inverse Mach reflection as well as a transitional regular reflection. The patterns of shock waves after focusing correspond to the transitional regular reflection and the transitional Mach reflection.

References

[1] Guderley G (1942) Starke kugelige und zylindrische Verdichtungstösse in der Nähe des Kugelmittelpunktes bzw der Zylinderachse, Luftfahrt forschung 19,302.

[2] Sturtevant B, Kulkarny VA (1976) The focusing of weak shock waves, *J. Fluid Mech.* 73 : 651-671.

[3] Watanabe M and Takayama K (1991) Stability of converging cylindrical shock waves, *Shock Waves* 1 : 149-160.

[4] Milton BE and Archer DR (1975), Proc. 10th Intl. Symp. on Shock Tube and Waves.

[5] Milton BE (1989), The focusing of shock waves in two-dimensional and axisymetric ducts, In: Takayama K (Ed), Proc Int. Workshop on Shock Wave Focusing, Shock Wave Research Center, Inst. Fluid Sci., Tohoku University.

[6] Takayama K and Ben-Dor G (1986) Reflection and diffraction of shock waves over a circular concave wall, Rep. of Inst. High Speed Mech., Tohoku Univ 51 : 43-87.

[7] Izumi K, Aso S, Nishida M (1994) Experimental and computational studies focusing processes of shock waves reflected from parabolic reflectors, *Shock Waves*, 3 : 213-222.

[8] Yang JM and Takayama K (1994) Holographic interferometric study of transition of reflected shock waves in a diaphragmless shock tube, In: Proc 3rd Asian Symp on Flow Visualization, pp 779-785.

[9] Sun M, Takayama K (1996) A holographic interferometric study of shock wave focusing in a circular reflector, *Shock Waves* 6: 323-336

[10] Sun M and Takayama K (1996) Quantitative analysis of shock wave focusing in a circular reflector, In: JSME Spring Annual Meeting, Tokyo, Japan.

[11] Courant R, Friedrichs KO (1948) Supersonic Flow and Shock Waves, Wiley Interscience, New York.

[12] Han ZY and Yin XZ (1993) Shock Dynamics, Kluwer Academic Publishers/Science Press.

[13] Ben-Dor G, Takayama K (1992), The phenomena of shock wave reflection - a review of unsolved problems and future research needs, *Shock Waves*, 2 : 211-223.

Chapter 6

Conclusions

The highlights of the thesis are summarized here:

- The flow solver is a novel central difference scheme, which is the Lax-Wendroff scheme with conservative smoothing. Extensive numerical tests show that the scheme is independent of flow conditions. The strengths of the approach may be summarized as follows. First, it results in a compact computational stencil, which consists of three nodes in each dimension. Second, it is a truly multidimensional method. The numerical flux is predicted by solving the multidimensional Euler equations without any one-dimensional assumption there. Finally it does not involve the Riemann solver in the flux evaluation, so that it can be directly applied to other systems of conservation laws.

- An essentially vectorizable data structure is designed and has been applied to arbitrarily adaptive quadrilaterals. The data structure provides logic-free topological adjacencies for the solver with negligible overheads. It makes not only the flow solver but also mesh adaptation easily vectorized. The efficiency of vector processing is comparable with that of a structured solver.

- In solving the Navier-Stokes equations, unlike the hybrid methods which combine structured quadrilaterals around body with unstructured triangles outside, the present unstructured adaptation method adopts uniformly unstructured quadrilaterals. It neatly generates a layer of body-fitted orthogonal mesh by repeatedly refining the cells in the boundary layer. It is a real bonus that

quadrilateral mesh generation is readily automated.

- The image data processing partially supported by the numerical simulation has been successfully applied to the shock tube experiment, in which very delicate shock wave interactions are well resolved. The numerical results obtained by the vectorizable adaptive solver on unstructured quadrilateral grids agree very well with the experimental finite fringe interferograms.

Acknowledgement

I would like first to thank Professor Kazuyoshi Takayama, Director of the Shock Wave Research Center, Institute of Fluid Science, Tohoku University, who provided the author with, besides Tatami and bread, inspiration and vision. I am also grateful to Mrs. Chieko Takayama. She often on holidays threw a party chaired by Professor Takayama. The parties I attended in three years gave me warmness as well as delicious Japanese food. Their support started with offering me the first Sashimi and never ceased. I do hope that I could make up for those other than just acknowledging it here.

I would like to thank Professor Kazuhiro Nakahashi, Professor Keisuke Sawada, Dr. Satoru Yamamoto and Dr. Akihiro Sasoh of Tohoku University for their helpful suggestions and comments. I am particularly grateful to Dr. Osamu Onodera, Mr. Hidenori Ojima, Mr. Toshihiro Ogawa and Mr. Kikuo Takahashi of the Institute of Fluid Science for their assistance in conducting experiments. I did benefit from their extensive experience which is never taught in classroom, such as how to search for a leak, how to search for the source of noise on a photo.

I especially thank Dr. Zonglin Jiang and Dr. Jiming Yang for their constant help not only in academic research but also in other matters. I have also the pleasure to thank Professor Peter Voinovich of Advanced Technology Center in Russia, Professor Joseph Falcovitz of Hebrew University in Israel, Dr. Tsutomu Saito of Cray Research Japan and Dr. Eugene Timofeev of the Institute of Fluid Science for their discussions and criticisms on numerical work.

I would like to express thanks to my former advisors and teachers in University of Science and Technology of China, Professor Zhaoyuan Han, Professor Xieyuan Yin, Professor Xiezhen Yin and many others for their teachings. With the knowledge learned in their classes, I always feel confident in challenging the unknown in these years.

I will remain indebted to many friends and students, Dr. Tetsuya Kodama, Dr. Toshikatsu Meguro, Dr. Kiyoshi Yamada, Mrs. Chikako Okamoto, Mr. Jurgen Schumacher, Miss Naoko Yagi, Miss Takako Iijima, Mr. Kensuke Koremoto, Mr.

Kazuyuki Yada, Mr. S.H.R. Hosseini, Mr. Satoshi Nonaka, Mr. Dan Igra and other people in the Shock Wave Research Center.

Lastly, I thank my parents for their pride and faith in my accomplishments and abilities, it is what kept me going when many times I didn't think I could.