Numerical Simulations of the Steady Euler Equations on Unstructured Grids

HU Guanghui

A thesis submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Principal Supervisor: Prof. TANG Tao

Hong Kong Baptist University

October 2009
Declaration

I declare that this thesis has been composed by myself under the guidance of my principal supervisor Prof. TANG Tao, and co-supervisor Prof. WU Xiaonan. The thesis has not previously included in any thesis, dissertation or report submitted to any institution for a degree, diploma or other qualification. All sources of information have been acknowledged by means of references to the relevant publications.

Signature:____________________

October 2009
Abstract

This thesis is concerned with effective and robust numerical schemes for solving steady Euler equations. For solving the nonlinear system resulting from the discretization of the steady Euler equations, we employ a standard Newton method as the outer iterative scheme and a linear multigrid method as the inner iterative scheme with the block lower-upper symmetric Gauss-Seidel iteration as its smoother. The Jacobian matrix of the Newton-iteration is regularized by the local residual, instead of using the commonly adopted time-stepping relaxation technique based on the local CFL number. The local Jacobian matrix of the numerical fluxes are computed by using the numerical differentiation, which can significantly simplify the implementations by comparing with the manually derived approximate derivatives.

In the reconstruction step, the linear reconstruction and the quadratic reconstruction are studied respectively. For the linear reconstruction, the approximate polynomial in each cell is obtained by using the WENO reconstruction method. The numerical results show that the algorithm works very well with the WENO reconstruction. Compared with the results given by using the Venkatakrishnan limiter, the WENO reconstruction method gives superior convergence order, and non-oscillatory and sharp shock profiles. Although the WENO method works very well for the linear case, the convergence to the steady state of the algorithm is affected if the WENO method is extended to the quadratic case directly. So for the quadratic reconstruction, a new hierarchical WENO reconstruction method is introduced to improve the convergence to steady state and also to preserve the formal order of accuracy. The efforts are made to balance the convergence order of the numerical


discretization, the ability of avoiding the non-physical oscillations, and the efficiency of the Newton-iteration.

The last part of the thesis is concerned with using the $h$-adaptive technique to enhance the performance of the proposed numerical algorithms. Numerical results show that with the $h$-adaptive methods the grids around the shock regions are locally refined successfully, which can save a large amount of computational time and memory.
Acknowledgements

First and foremost, I am particularly grateful to my supervisor, Prof. Tao Tang, for his inspiring guidance and enthusiastic support. His superb intuition, his broad knowledge and experience, and his continuous encouragement throughout this research have all been indispensable for whatever I have achieved. I would also like to thank my co-supervisor, Prof. Xiaonan Wu, for his encouragement and support.

I would like to express my deep gratitude to Prof. Ruo Li in Peking University. He inspired my work by numerous fruitful discussions, taught me practices and skills and gave me continuous support throughout my PhD education. I still remember the excitement of solving a million order linear system within a few seconds just on my notebook. His unfailing support is a major factor in the completion of my doctoral studies.

I would like to thank Prof. Paul Zegeling of Utrecht University in the Netherlands. During my three months visit, I had benefited a lot from the discussion with him on the moving mesh methods. Also I would like to thank Dr. Arthur van Dam for his enthusiastic help on my research and my daily living during my visit. Thanks are also given to Ms. Alexandra Babenko and Mr. Dmytro Makogon for their help. With these friends, I had a very wonderful experience in the Netherlands.

I would like to thank Prof. Vít Dolejší of Charles University in Czech Republic. His expertise on numerical simulations helps me to much better understand many details of my research projects. I also would like to thank Dr. Václav Kučera for his useful sharings on research.

My thanks extend to all staffs and postgraduate students in our department for
their help. In particular, I wish to give special thanks to my fellow postgraduates, Dr. Zhonghua Qiao, Dr. Heyu Wang, Mr. Yubo Zhang, Dr. Xianliang Hu, Dr. Jiwei Zhang, Dr. Leihong Zhang, Dr. Changqiu Jin, Dr. Qiaolin He, for sharing with me some of their profound knowledge on scientific computing. I would like to thank the departmental secretaries and technicians, Ms. Claudia Chui, Ms. Susanna Sze, Ms. Corinna Yu, Ms. LAW, Natalie, Ms. TSE, Amanda, Ms. Tammy Lam, Mr. Cheong-Wing Yeung, for their kind assistance.

I would like to give special thanks to my fiancée, Dr. Lei Yang, for her great love and unlimited support.

Finally, I would like to thank my parents. Their love always accompanies me.
Table of Contents

Declaration i

Abstract ii

Acknowledgements iv

Table of Contents vi

List of Tables ix

List of Figures x

Chapter 1 Introduction 1

1.1 The Euler Equations 2

1.2 Mesh Generation 5

1.3 The Finite Volume Method 7

1.3.1 Solution Reconstruction 11

1.3.2 Slope Limiter Methods 13

1.3.3 ENO/WENO Reconstruction 14

1.3.4 The HLLC Riemann Solver 16

1.4 The Residual Distribution Method 18

1.5 Temporal Discretization 19

1.6 Some Acceleration Techniques 19

1.6.1 Local Time-Stepping 20

1.6.2 The Multigrid Method 20
Chapter 2  The Linear Finite Volume Solver 24

2.1 The Finite Volume Discretization for the Steady Euler Equations 27
2.2 The Reconstruction and Limiting Strategies 28
  2.2.1 The Linear Reconstruction 29
  2.2.2 The Venkatakrishnan Limiter 30
  2.2.3 The WENO Reconstruction 33
2.3 Newton Iteration 35
  2.3.1 The Jacobian of the Numerical Flux 35
  2.3.2 Regularization 36
2.4 Multigrid Method 38
  2.4.1 The Projection Operator 39
  2.4.2 The Smoother 42
2.5 Boundary Condition 44
  2.5.1 Solid Wall Boundary Condition 44
  2.5.2 Farfield Boundary Condition 45
2.6 Numerical Results 46
  2.6.1 Numerical Convergence Tests 47
  2.6.2 The Robustness of the Algorithm 62
  2.6.3 Remarks on the Efficiency of the Algorithm 63
2.7 Conclusion Remarks 68

Chapter 3  The Quadratic Solver Based on the Residual Distribution Schemes 69

3.1 High order reconstruction 73
3.2 The WENO hierarchical limiting strategy 75
  3.2.1 Summarization of the Hierarchical Reconstruction 75
3.2.2 Implementation of The Hierarchical Reconstruction for Steady Problems ........................................... 78

3.3 Remarks on the Curved Boundary .............................................. 81

3.4 Numerical Results .................................................................. 84
  3.4.1 Numerical Convergence Tests .............................................. 85
  3.4.2 Robustness .................................................................. 99

3.5 Conclusion Remarks ................................................................ 103

Chapter 4 Combination with the Adaptive Techniques 106
  4.1 Moving Finite Element Method for the Simulation of Gravity Fingers in Porous Media ......................................................... 107
    4.1.1 Finite Element Discretization .............................................. 108
    4.1.2 Moving Mesh Strategy ......................................................... 109
    4.1.3 Numerical Results .......................................................... 111
  4.2 h-Adaptive Method ................................................................. 114
    4.2.1 Mesh Refinement and Coarsening ........................................ 115
    4.2.2 Remarks on the Weight in the Reconstruction ................. 120
    4.2.3 Numerical Results .......................................................... 120
  4.3 Conclusion Remarks .............................................................. 122

Chapter 5 Concluding Remarks 134

Bibliography 136

Curriculum Vitae 147
List of Tables

3.1 $L_2$ error and convergence order of quadratic reconstruction with hierarchical limiting strategy. .................................................. 81

3.2 The $L_2$ entropy errors and orders of convergence for the linear reconstruction and quadratic reconstruction computations for the flow around a circle on four successive grids; errors are measured in regions $0.5 \leq r \leq 2.0$. "L-lim" and "Q-lim" stand for linear and quadratic reconstruction with limiting procedure respectively. "Q." stands for quadratic reconstruction without limiting procedure. .................. 90

3.3 Same as the last columns in Table 3.2, except that the curved boundary is approximated by the linear segments. ......................... 91

4.1 The data file of the mesh shown in Fig. 4.3. .............................. 116
List of Figures

1.1 Definition of a finite control volume. ........................................ 3
1.2 Structured meshes (left) and unstructured meshes (right)............. 6
1.3 Cell-centred scheme (left) and Cell-vertex scheme with dual control volume: (middle) and overlapping control volume (right). .......... 8
1.4 Left and right state at the cell edge. The cell-centred schemes is used. 11
1.5 Unlimited (left) and limited (right) linear reconstruction. .......... 13
1.6 The comparison of the reconstruction between the ENO reconstruction (solid line) and the Lax-Wendroff reconstruction (dashed line). . 15
1.7 HLL (left) and HLLC (right) approximate Riemann solvers. .......... 17

2.1 Left: the reconstruction patch $P(K)$ used for the least square reconstruction. Right: four patches, $\{K, K_0, K_1, K_2\}$, $\{K, K_0, K_{0,0}, K_{0,1}\}$, $\{K, K_1, K_{1,0}, K_{1,1}\}$, and $\{K, K_2, K_{2,0}, K_{2,1}\}$, which are used for the WENO reconstruction in the cell $K$. ................................. 30
2.2 The $\min(1, x)$ function used in Barth limiter, and its smooth Venkatakrishnan approximation. .................................................... 32
2.3 Results of themesh-refinement study for the linear reconstruction without limiting procedure, the linear WENO reconstruction and the linear reconstruction with Venkatakrishnan limiter. The initial mesh contains 64 cells, and the reconstruction is implemented on six successively refined meshes. ................................. 34
2.4 Representation of a fine mesh with meshsize $h$, and of two coarse meshes with meshsizes $2h$ and $4h$ respectively for the structured meshes case. ................................................................. 39
2.5 Element patches near the airfoil body of NACA 0012 generated by the aggregation from a quasi-uniform mesh. It shows the element patches on four continuous levels. .................................................. 41
2.6 Demonstration of physical domain and boundaries used in numerical simulations. ................................................................. 45
2.7 Mesh used in the simulation of subsonic flow around a disk. The left one is entire mesh, the right one is the detail of mesh around the disk. 48
2.8 The Mach isolines (bottom) obtained with mesh size $16 \times 12$ (top). 49
2.9 The Mach isolines (bottom) obtained with mesh size $32 \times 24$ (top). 50
2.10 The Mach isolines (bottom) obtained with mesh size $64 \times 48$ (top). 51
2.11 The Mach isolines (bottom) obtained with mesh size $128 \times 96$ (top). 52
2.12 Results of mesh-refinement study for linear reconstruction without limiting procedure, linear WENO reconstruction and linear reconstruction with Venkatakrishnan limiter. The error is given as the $L_2$ norm of the entropy production. ................................................................. 53
2.13 The mesh grids generated using EasyMesh. Left: the mesh around the NACA 0012 airfoil, and the whole mesh containing 2662 cells. Right: the mesh around the RAE 2822 airfoil, and the whole mesh containing 3444 cells. .................................................. 54
2.14 Free-stream configuration: Mach number 0.8, attack angle 1.25°. The airfoil is NACA 0012, and the mesh contains 2662 cells. Top: Mesh details around the airfoil. Bottom: The Mach isolines. ............................... 55
2.15 Same as Fig. 2.14, except that the mesh contains 10648 cells. .... 56
2.16 Same as Fig. 2.14, except that the mesh contains 42592 cells. .... 57
2.17 The comparison between the Mach isolines obtained with the linear
WENO reconstruction (top) and the reconstruction with Venkatakrishnan limiter (bottom). There are 2662 cells in the domain. . . . . . 58
2.18 Same as Fig. 2.17, except that there are 10648 cells in the domain. . 59
2.19 Same as Fig. 2.17, except that there are 42592 cells in the domain. . 60
2.20 Comparison of pressure distribution along the surface of the airfoil
between the results obtained with linear WENO reconstruction (left)
and reconstruction with Venkatakrishnan limiter (right). The mesh
contains 2662 cells (top), 10648 cells (middle), and 42592 cells (bottom). 61
2.21 The meshes for two airfoils in the flow fields. Left: the mesh around
a NACA 0012 airfoil and a RAE 2822 airfoil, and the whole mesh
contains 7870 cells. Right: the mesh around two NACA 0012 airfoils,
and the whole mesh contains 4360 cells. . . . . . . . . . . . . . . . . . 63
2.22 Results of Mach isolines (top), pressure distribution along the surface
of the airfoil (middle), and convergence history (bottom) obtained using
the algorithm with the linear WENO reconstruction. Left: NACA
0012 airfoil with Mach number 0.3 and attack angle 3.0°. Right: RAE
2822 airfoil with Mach number 0.75 and attack angle 1.0° . . . . . . 64
2.23 Same as Fig. 2.22, except for the free-stream configurations: NACA
0012 airfoil with Mach number 0.85 and attack angle 1.0° (left), and
Mach number 0.99 and attack angle 0.0° (right). . . . . . . . . . . . . 65
2.24 Same as Fig. 2.22, except for the free-stream configurations: one
NACA 0012 airfoil and one RAE 2822 airfoil in the flow fields with
Mach number 0.75 and attack angle 1.0° (left), and two NACA 0012
airfoils in the flow fields with Mach number 0.99 and attack angle
0.0° (right). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 66
3.1 The reconstruction patches \( \mathcal{P}(\mathcal{K}) \) used for the least square reconstruction.
Left: linear case. Right: quadratic case . . . . . . . . . . . . . . 74
3.2 The computational domain (dashed line) and its polygon approxima-
tion (solid line). .......................................................... 82
3.3 Quadratic isoparametric element (left) and simplified quadratic ele-
ment (right). .............................................................. 83
3.4 Quadrature points for Simpson formula. In our algorithm, points A,
B, C are used. .............................................................. 84
3.5 The Mach isolines obtained with linear WENO reconstruction (top)
and with quadratic reconstruction using WENO hierarchical limiting
procedure (bottom). The mesh size is 16 × 12. .......................... 86
3.6 Same as Fig. 3.5, except that the mesh size is 32 × 24. ............... 87
3.7 Same as Fig. 3.5, except that the mesh size is 64 × 48. ............... 88
3.8 Same as Fig. 3.5, except that the mesh size is 128 × 96. ............... 89
3.9 The Mach isolines obtained with quadratic reconstruction using WENO
hierarchical limiting procedure, the mesh size is 64 × 48. The curved
boundary is approximated with the straight line. ........................ 91
3.10 Mach isolines around the whole airfoil (top) and around the shock
profiles (bottom) obtained with the quadratic hierarchical WENO
reconstruction. The free-stream condition is Mach number 0.8 and
attack angle 1.25°. The mesh contains 2662 cells. ......................... 93
3.11 Same as Fig. 3.10, except that the mesh contains 10648 cells. ....... 94
3.12 Same as Fig. 3.10, except that the mesh contains 42592 cells. ....... 95
3.13 Comparison of the pressure distribution along the airfoil surface ob-
tained with two methods: the algorithm with quadratic reconstruc-
tion where no limiting procedure is used (top), and with quadratic
hierarchical WENO reconstruction (bottom). ........................... 96
3.14 Same as Fig. 3.11, expect that the limiting strategy is not used. ... 97
3.15 Convergence history of the algorithm with quadratic WENO recon-
struction (top) and with the hierarchical WENO reconstruction. The
mesh contains 42592 cells. ............................................... 98
3.16 The contours of Mach number (top left), pressure (top right), surface pressure profiles of the airfoil (bottom left), and convergence history (bottom right) for NACA 0012 airfoil with free-stream configuration: the Mach number is 0.3 and the attack angle is 3°.

3.17 Same as Fig. 3.16 except with Mach number 0.73 and attack angle 1.0°, and the airfoil is RAE 2822.

3.18 Same as Fig. 3.16 except with Mach number 0.99 and attack angle 0°.

3.19 Same as Fig. 3.17 except with two airfoils in the flow field: NACA 0012 and RAE 2822.

3.20 Convergence history for NACA 0012 airfoil in the flow field with free-stream configuration: Mach number 0.3, attack angle 3° (left), and Mach number 0.99, attack angle 0° (right). The parameter \( \tau = \tau_i = 0.6 \) is used.

4.1 Computational region for single finger.

4.2 Numerical solutions of Nonequilibrium Richard equation with initial condition: \( K(S) = S^2 \), \( D(S) = S^2 \), \( \tau = 0.5 \), \( S_1 = 0.7 \) and \( S_0 = 0.06 \).

4.3 A mesh with four elements.

4.4 A uniform and successive refinement of the element \( K \).

4.5 The mesh before (left) and after (right) the implementation of the refining and coarsening.

4.6 Element Patches near the airfoil body of NACA 0012 generated by Algorithm 2. It shows the element patches on four successive levels when \( h \)-adaptive technique was adopted.

4.7 The Mach isolines (top) and mesh details (bottom) around the NACA 0012 airfoil which obtained with linear WENO reconstruction, and \( h \)-adaptive method is used. There are 2662 cells in the domain.

4.8 Same as Fig. 4.7, except that there are 4555 cells in the domain.

4.9 Same as Fig. 4.7, except that there are 8746 cells in the domain.

4.10 Same as Fig. 4.7, except that there are 15031 cells in the domain.
4.11 Same as Fig. 4.10, except that just the Mach isolines and the meshes shown in the box in Fig. 4.10 (top) are demonstrated. . . . . . . . . . 128

4.12 The convergence history of the algorithm which is implemented on four successively refined mesh with $h$-adaptive method. . . . . . . . . . 129

4.13 The pressure distribution along the NACA 0012 airfoil obtained from algorithm which is implemented by using four successively refined mesh with $h$-adaptive method. Top left: 2662 cells; Top right: 4555 cells; Bottom left: 8746 cells; Bottom right: 15031 cells. . . . . . . . . 130

4.14 The Mach isolines (top) and mesh details (bottom) around two NACA 0012 airfoils which obtained with quadratic hierarchical WENO reconstruction, and $h$-adaptive method is used. There are 4360 cells in the domain. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131

4.15 Same as Fig. 4.14, except that there are 7240 cells in the domain. . . 132

4.16 Same as Fig. 4.14, except that there are 13339 cells in the domain. . . 133
Chapter 1

Introduction

Computational Fluid Dynamics (CFD) has been one of the most important methods for understanding the fluid flows phenomenon since early 1970s. There are many disadvantages of the traditional methods which are used to investigate the fluid dynamics. Since the mathematical models of fluid dynamics have high nonlinearity, most of the analytical solutions for these problems cannot be obtained; for more and more practical problems, the expense for each physical experiment is too high, and for some problems such as the high temperature flows or/and the high speed flows around aircrafts, it is impossible to give results with physical experiments. Thanks to the rapid development of CFD, all these drawbacks are avoided. Nowadays, it was possible to compute the inviscid flows past the complete aircraft configurations or inside of turbomachines.

One of the most interesting phenomena in the fluid dynamics is the steady state of the flows. For example, when the aircraft is in the cruise status, the variables such as the density, the velocity, the pressure, and the temperature around the aircraft are unchanged. By solving the temporal systems up to sufficiently large time, such steady states can be obtained approximately. Due to the restrictions of CFL condition for the stability of the numerical schemes, the length of the time step cannot be too large. Consequently, it may take a very long time for the numerical simulations to achieve the steady state. This is not endurable for the research of the shape optimization of the aircraft which is formed by a series of
steady state problems. As a result, the efficient and robust solvers for the steady state flows become necessary. So far many classical acceleration techniques have been presented such as local time stepping, enthalpy damping, residual smoothing, and multigrid method. With these modern techniques, the nonlinear systems can be solved with residual to machine accuracy on desktop computers in a few minutes.

The main purpose of this thesis is to develop an effective and robust framework of the numerical algorithm for the 2D steady Euler equations. In this chapter, some background materials will be presented, which will be useful for the following chapters. We first briefly derive the Euler equations. Then several useful methods will be reviewed, including the numerical discretization methods, the classical reconstruction and limiting techniques, and the implicit method and its convergence acceleration strategy. In the last section, a brief description of this thesis will be given.

1.1 The Euler Equations

The principal equations of fluid dynamics are derived based on the fact that the dynamical behaviour of a fluid is determined by the conservation laws,

1. the conservation of mass;
2. the conservation of momentum;
3. the conservation of energy.

The conservation of the flow quantity means that the total variation inside an volume can be expressed as the net effect of the amount of the quantity being transported across the boundary, of any internal forces and sources, and of external forces acting on the volume [8]. To describe the conservation laws mathematically, first we give the demonstration of a control volume in Fig. 1.1, where Ω is the control volume and ∂Ω is its boundary. The surface segment dS is also introduced, and \( \mathbf{n} = (n_x, n_y) \) denotes the outward pointing unit normal with respect to dS. The flow velocity is denoted by \( \mathbf{v} = (u, v) \).
Figure 1.1: Definition of a finite control volume.

The conserved quantity in the conservation of mass is the density $\rho$. To derive the continuity equation, we consider the control volume shown in Fig. 1.1. The time rate of change of the total mass inside the control volume $\Omega$ is

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \, d\Omega,$$

while the convective flux across the surface element $dS$ is written as

$$\rho (\mathbf{v} \cdot \mathbf{n}) dS.$$

So the conservation law of mass is presented by the following equation

$$\frac{\partial}{\partial t} \int_{\Omega} \rho d\Omega + \oint_{\partial \Omega} \rho (\mathbf{v} \cdot \mathbf{n}) dS = 0.$$

In addition to the conservation law of mass (1.3), we also need an equation for the velocity. The velocity itself is not the conserved quantity, but the momentum is. The product $\rho \mathbf{v}$ denotes the density of momentum in the control volume $\Omega$, which means the total momentum in $\Omega$ is given by the integral of $\rho \mathbf{v}$ over the control volume $\Omega$. So the time rate of change of the total momentum inside the control volume $\Omega$ is

$$\frac{\partial}{\partial t} \int_{\Omega} \rho \mathbf{v} d\Omega.$$

There are two ways to change the total momentum in the control volume $\Omega$. The first one is that there is momentum carried past the surface $dS$ along with the moving fluid which can be written as $\rho \mathbf{v} (\mathbf{v} \cdot \mathbf{n}) dS$, while the other one is due to the pressure
\( p \) of the fluid, \( pI \cdot \mathbf{n}dS \) where \( I \) is the unit tensor. Consequently, the conservation law of momentum can be modeled as

\[
\frac{\partial}{\partial t} \int_{\Omega} \rho \mathbf{v} d\Omega + \oint_{\partial \Omega} \rho \mathbf{v} \cdot \mathbf{n} dS + \oint_{\partial \Omega} p \mathbf{n} dS.
\] (1.5)

Finally, the energy equation is derived from the fist law of thermodynamics. The total energy \( E \) is often decomposed as

\[
E = \rho e + \frac{1}{2} \rho \mathbf{v} \cdot \mathbf{v},
\] (1.6)

where \( \rho \mathbf{v} \cdot \mathbf{v}/2 \) denotes the kinetic energy, and \( \rho e \) denotes the internal energy, and \( e \) is the specific internal energy. The total energy advects with the flow, leading to the macroscopic energy flux term \( E(\mathbf{v} \cdot \mathbf{n})dS \) on the surface. In addition, the pressure \( p \) also leads to a flux in kinetic energy \( p(\mathbf{v} \cdot \mathbf{n})dS \). The final conservation law of energy is given as

\[
\frac{\partial}{\partial t} \int_{\Omega} E d\Omega + \oint_{\partial \Omega} (E + p)(\mathbf{v} \cdot \mathbf{n}) dS = 0.
\] (1.7)

Note that we have assumptions for the derivations, say, the flows are inviscid, and there is no source term inside the control volume \( \Omega \) and along the surface \( \partial \Omega \). Such assumptions actually give us the inviscid Euler equations, which describe the pure convection of flow quantities.

The compact form of Euler equations which are given by the three conservation laws can be written as

\[
\frac{\partial}{\partial t} \int_{\Omega} U d\Omega + \oint_{\partial \Omega} F(U) \cdot \mathbf{n} dS = 0,
\] (1.8)

where \( U \) is the so-called conservative variables, and has form in 2D case as

\[
U = \begin{bmatrix}
\rho \\
\rho u \\
\rho v \\
E
\end{bmatrix}.
\] (1.9)
The convective flux $F$ is given by

$$
F = \begin{bmatrix}
\rho (v \cdot n) \\
\rho u (v \cdot n) + n_x p \\
\rho v (v \cdot n) + n_y p \\
(E + p) (v \cdot n)
\end{bmatrix}.
$$

(1.10)

There are five unknowns, the density $\rho$, the velocity $v = (u, v)$, the energy $E$ and the pressure $p$ in the system, but just four equations. To enclose the system, the so-called equation of state has to be introduced. The common form of the equation of state for an ideal gas is read as

$$
E = \frac{p}{\gamma - 1} + \frac{1}{2} \rho v \cdot v,
$$

(1.11)

where $\gamma = 1.4$ is the ratio of specific heats of the ideal gas.

The system (1.8) presents the 2D unsteady Euler equations. It is a hyperbolic system, and so the properties propagate along the characteristic lines of the system. However, if the temporal term is dropped, the type of the system will be changed with the Mach number. For example, the system is elliptic when Mach number is smaller than 1.0, and is hyperbolic when Mach number is bigger than 1.0. This results in the difficulties to solve the steady Euler equations directly. The popular way to obtain the steady state of the flows is to solve the unsteady system up to the point that all time derivatives reach some tolerance criteria.

The numerical simulations of the Euler 1.8 equations contain two main ingredients: the mesh generation with which the finite number of geometrical elements are used to cover the flow field, and the flow solver which is used to generate the solutions of the governing equation over the meshes. In the following subsections, these two parts will be introduced respectively.

### 1.2 Mesh Generation

For classical numerical schemes such as the Finite Difference Methods, the Finite Volume Methods and the Finite Element Methods, the quality of the meshes is a
critical issue for obtaining high quality of the numerical results. There are two different types of the mesh grids, and we state them in the 2D case as following.

- Structured meshes. For each mesh grid in the domain, it can be accessed through its indices $i$, $j$ and the corresponding coordinates $x_{i,j}$ and $y_{i,j}$. The demonstration of the structured meshes is given in Fig. 1.2 (left).

- Unstructured meshes. The information of the meshes such as the indices of the nodes, the neighbours of certain node is totally unordered. The demonstration of the unstructured meshes is given in Fig. 1.2 (right).

For the structured meshes, the information of the nodes is stored in a linear way in the computer memory. This results in that one can access the neighbours of certain node very flexibly. For example, for the current node $(2, 2)$ in Fig. 1.2 (left), the right neighbour can be easily accessed by simply adding 1 to the $x$-component of the corresponding index, so its index is $(3, 2)$. Besides this advantage, the well-ordered indices of the nodes also give a banded flux jacobian matrix when the implicit method is used, and the treatment of the boundary condition is also simplified greatly by using the structured meshes. However, the disadvantage of the structured meshes is also obvious: for the complex domain, it is not easy to
give the high quality structured meshes. In [8], there is a summarisation about the methods which can be used to generate high quality structured meshes for the complex domains.

Compared with the structured case, the unstructured manner gives very large flexibility to generate meshes for very complex domains. It also gives the flexibility to use different mesh sizes, or even the different element shapes in meshes. This is good for generating high quality of meshes for certain special problems. For example, for the problems with boundary layers, the quality of numerical solutions can be improved significantly by using mixed meshes which include the rectangular elements used nearby the solid walls, and the triangular elements inside the domain. The further advantage of unstructured meshes is that the solution depending on grid refinement and coarsening can be handled in a relatively native and seamless manner [8]. Of course, there are disadvantages about the unstructured meshes. For example, to obtain the neighbours of the node with the index 8 in Fig. 1.2, additional memories in the computer will be needed to restore the indices of the current node and its neighbours. However, the flexibility of generating high quality of meshes for the complex domains results in the wide usage in most of commercial CFD softwares. In this thesis, the unstructured meshes are used, and they are generated by the powerful software EasyMesh [68].

In the simulations, the finite volume scheme is adopted for the linear case, while the Residual Distribution Scheme is used for the quadratic case. Although they are two different numerical schemes, there are still some common parts for both schemes. In the following sections, the finite volume scheme is summarised first. Then the comment on the difference and relations between the finite volume scheme and the residual distribution scheme is given later.

1.3 The Finite Volume Method

The finite volume method has been widely used since it was first employed by McDonald [65]. One feature of the finite volume method is that it discretizes the
conservation laws directly on the physical domain, and then the surface integral in
governing equations is approximated by the sum of the fluxes crossing the individual
faces of the control volume. So there is no transformation between the computational
domain and physical domain, and there is no limit for the shape of the control
volume in the physical domain theoretically, which makes the finite volume method
very flexible - both structured and unstructured meshes can be adopted. It allows
the finite volume method to simulate flows in very complex geometries.

The further advantage of the finite volume method is that, since the conservation
laws are discretized directly, mass, momentum and energy are also conserved by the
numerical schemes. This gives the finite volume method the ability to calculate the
*weak solutions* of the governing equations correctly.

In this thesis, we focus on the unstructured meshes. There are two main types
of the control volume,

- **Cell-centred** scheme, which is shown in Fig. 1.3 (left). The place to store
  the conserved variables for Cell-centred scheme is the centroid of the control
  volume.

- **Cell-vertex** scheme, which is shown in Fig. 1.3, with dual control volume
  (middle) and with overlapping control volume (right). The conserved variables
  are often stored at the grid points.

Each scheme has advantages and disadvantages. For example, for Cell-vertex
scheme with dual control volume, the difficulty arises for the domain with curved boundaries. For this case, boundaries of the computational domain will not agree with the physical domain. Consequently, the mismatch will introduce the discretization error compared with the other two schemes. Although the cell-vertex scheme with overlapping control volume could remedy this discretization error, it cannot be combined with the popular upwind methods such as TVD, AUSM and CUSP. For details about properties of these schemes, we refer to [8] and references therein. In this thesis, the cell-centred scheme is adopted.

Now we are ready to give the discretized form of (1.8). Suppose the physical domain is \( \Omega \). Let \( T \) be a triangular partition of \( \Omega \), and \( K_i \in T \) be one cell in the partition. We assume the intersection of two different cells can only be either an edge or a vertex. Let \( n_{ij} \) denote the outer unit normal on the edge \( e_{ij} \), pointing from \( K_i \) to \( K_j \). In cell \( K_i \in T \), the system (1.8) can be written as

\[
\frac{\partial}{\partial t} \int_{K_i} U_i d\Omega + \sum_{e_{ij} \in \partial K_i} \int_{e_{ij}} F(U_i, U_j) \cdot n_{ij} dS = 0. \tag{1.12}
\]

In what follows, the numerical schemes which are used to evaluate the convective fluxes will be introduced. For the cell-centred scheme, there exist two main schemes to implement the evaluation. One is the central schemes, which was first implemented for the Euler equations on structured meshes by Jameson et al. [39], then it was extended to the unstructured meshes [38, 58]. The basic idea of the central schemes is to use the arithmetic average of the conservative variables on both sides of the edge of the cell as the convective fluxes at this edge. As well known, the central schemes are computationally considerably cheaper than other schemes. However, for preventing the odd-even decoupling of the solution and wiggles at shocks, the artificial dissipation has to be added. Consequently, the quality of the numerical solutions obtained with central schemes around the shocks and boundary layers is lower than that obtained with other schemes, say, the upwind schemes.

In practices, the upwind schemes are more popular than the central schemes. The basic idea of the upwind schemes is to split the information into different parts according to the characteristic variables of the system, then use these detailed infor-
mation to generate much more accurate numerical fluxes. There are many different implementations for the upwind schemes such as the *Flux-vector Splitting*, *Flux-difference splitting*, *Total Variation Diminishing*, and *Fluctuation-splitting* schemes. All these schemes were first proposed for the structured meshes, and can be applied to the unstructured meshes without modifications to the basic methodology. In [8], Blazek gave the summarisation of these schemes.

Among these schemes, the flux-difference splitting schemes are derived based on the solutions of the locally one-dimensional Euler equations for discontinuous states at an edge. Such solutions can be obtained by solving the *Riemann* (shock tube) problem. Godunov [25] gave the first implementation of flux-difference splitting method, and his method can be summarised as the following *Reconstruction-Evolve-Average* (REA) algorithm [45].

\begin{algorithm}
\textbf{Algorithm 0: [REA]}

1. Reconstruct a piecewise polynomial $h_{i,t_n}$ in each cell $\mathcal{K}_i$, from the cell averages $\bar{U}_{i,t_n}$ in every cell. Make sure the variables are conserved in the cell, say,

   $$\int_{\mathcal{K}_i} h_{i,t_n} d\Omega = \bar{U}_{i,t_n}.$$

2. Evolve the hyperbolic equation exactly (or approximately) with the reconstructed data to obtain $h_{i,t_{n+1}}$, where $t_{n+1} = t_n + \Delta t$.

3. Average $h_{i,t_{n+1}}$ over the cell $\mathcal{K}_i$ to obtain the updated cell averages

   $$\bar{U}_{i,t_{n+1}} = \frac{1}{|\mathcal{K}_i|} \int_{\mathcal{K}_i} h_{i,t_{n+1}} d\Omega.$$

\end{algorithm}

In [25], the simplest piecewise constant polynomial was used, say, the *Left State* $U_L$ and *Right State* $U_R$ (Fig. 1.4) of the common edge of cells $\mathcal{K}_i$ and $\mathcal{K}_j$ were equal to the cell averages $\bar{U}_i$ and $\bar{U}_j$ respectively. This results in a spatial discretization which is only first-order accurate. Such piecewise constant approximation is too diffusive. The higher order reconstruction can be implemented to obtain higher order numerical accuracy.
1.3.1 Solution Reconstruction

To achieve the second-order accuracy, there are many reconstruction methods. The direct way is to extend the MUSCL method [44], which was proposed for the structured meshes, to the unstructured meshes [17]. However, additional storages are needed for the phantom nodes which are used to evaluate the left and right states, and such approximation method does not work smoothly for the boundary cells since the phantom points lie outside the physical domain.

The other way is to use the piecewise linear reconstruction. The essential is to evaluate the gradient of the solution at the reference points. To obtain the evaluation of the gradient, the Green-Gauss approach can be adopted, which uses the following relation to approximate the gradient

$$\nabla U \approx \frac{1}{|\Omega|} \int_{\partial \Omega} U n dS, \quad (1.13)$$

where $\Omega$ is a closed domain, and $n$ is the unit out normal vector on the surface of the $\Omega$. For the cell-centred schemes, the gradient on the centroid on the cell $K_i$ in Fig. 1.4 can be obtained as

$$\nabla U_i \approx \frac{1}{|K_i|} \sum_j \frac{1}{2} (\bar{U}_i + \bar{U}_j) n_{ij} |S_{ij}|, \quad (1.14)$$

where $|S_{ij}|$ is the length of the common edge of the cell $K_i$ and $K_j$, and summation extends over all edges of the cell $K_i$. The Green-Gauss method can be implemented efficiently. However, this approach becomes unreliable when the highly stretched meshes are used.
The $k$-exact Reconstruction

In [2], Barth presented the so-called $k$-exact reconstruction for the cell-vertex scheme with dual control volume. Then Mitchell and Walters [67] gave the implementation of the $k$-exact reconstruction for the cell-centred schemes. The idea of the $k$-exact reconstruction is concise. For certain cell $\mathcal{K}_i$, suppose there is an approximate polynomial $h_i$ for the solution in this cell. The polynomial $h_i$ can be obtained by using Taylor Series at the reference point, and the expansion is truncated after the $k$-th order term. For the cell-vertex scheme, the reference point is the node of the meshes, while for the cell-centred scheme, it is the centroid of each cell. For conserving the variables in the cell $\mathcal{K}_i$, the $k$-exact reconstruction required that the cell average of the approximate polynomial $h_i$ was equal to the original cell average $\bar{U}_i$, say,

$$\frac{1}{|\mathcal{K}_i|} \int_{\mathcal{K}_i} h_i d\Omega = \bar{U}_i,$$

(1.15)

where $|\mathcal{K}_i|$ is the area of the cell $\mathcal{K}_i$. Beside the cell $\mathcal{K}_i$, $k$-exact method also required that the polynomial $h_i$ conserved the variables in the neighbours of $\mathcal{K}_i$. Suppose cells $\mathcal{K}_j, j = 0, 1, 2, \cdots, m$ are the neighbours of the cell $\mathcal{K}_i$, the following equations can be obtained

$$\frac{1}{|\mathcal{K}_j|} \int_{\mathcal{K}_j} h_i d\Omega = \bar{U}_j, j = 0, 1, 2, \cdots, m.$$ (1.16)

If $m > k$, the equations (1.15) and (1.16) are an over-constrained system. Such system actually minimized the difference between the cell averages of the approximate polynomial $h_i$ over these cells and the original cell averages of these cells. By solving this system, the $k$-th order approximate polynomial of the solution is obtained in the cell $\mathcal{K}_i$. Note that the obtained $h_i$ does not conserve the variables in the cell $\mathcal{K}_i$ strictly, so certain adjustment is needed for the constant term of $h_i$. After implementing the reconstruction in cells $\mathcal{K}_i$ and $\mathcal{K}_j$ (Fig. 1.4), the $U_L$ and $U_R$ can be obtained from the polynomials $h_i$ and $h_j$ respectively.

Based on the description above, we can see that with the $k$-exact reconstruction, the conservation of the variables such as the density, the momentum, and the energy in each cell can be guaranteed. The further advantage is that such approach supplys
Figure 1.5: Unlimited (left) and limited (right) linear reconstruction.

a uniform framework to reconstruct polynomial with arbitrary order. In this thesis, the $k$-exact reconstruction is used for the linear case.

### 1.3.2 Slope Limiter Methods

For linear or higher order reconstruction, the limiting procedure is necessary for preventing the non-physical oscillations. What expected is a monotonicity preserving method which means that the maxima of the solution must be non-increasing, minima non-decreasing, and no new local extrema is created during the time evolution.

The effect of the limiting procedure can be demonstrated by Fig. 1.5. It shows the slope reduction at the cell $i$ and the change of the slope at the cells $i - 2$ and $i - 1$. With the limiting procedure, the solution becomes monotone.

For the structured meshes, there are many well-established slope limiters such as the *Minmod* function, the *Superbee* function and the *MC* function. All these limiters are summarised in [45].

For the unstructured meshes, Barth and Jespersen gave the first implementation of a limiter function in [4]. Such limiter function can be used for both cell-vertex and cell-centred schemes. With limiter of Barth and Jespersen, the non-physical oscillations are removed effectively. However, it is rather dissipative and tends to
smear discontinuities. Furthermore, the usage of the min function adversely affects the differentiability of the numerical schemes, and so the convergence of steady state of the algorithm. To avoid these disadvantages, Venkatakrishnan [83] proposed a limiter which the min function was replaced by a smooth function. The revised limiter demonstrates the superior convergence properties. But it can not guarantee the strictly monotonicity of the solution, and the numerical accuracy in the smooth region is also affected since the limiter can be enacted even in the region with smooth solution.

All limiter functions introduced above actually constrain the polynomial obtained by using a fixed stencil, so the convergence order for the smooth solution must be adversely affected. To keep the desired numerical accuracy and also remove the non-physical oscillations, the Essential Non-Oscillatory (ENO) and Weighted Essential Non-Oscillatory (WENO) schemes are introduced.

1.3.3 ENO/WENO Reconstruction

The ENO scheme was first presented by Harten et al.[28, 29]. Different from the reconstruction methods which use a fixed stencil, the ENO method uses the "smoothest" stencil of the current cell to reconstruct the polynomial. Such idea can be demonstrated by Fig. 1.6. This figure shows the linear reconstruction of the solution in the region with shock. If the Lax-Wendroff schemes are used, the stencil for the reconstruction in the cell $i$ should be the cells $\{i, i + 1\}$, then the reconstructed linear function is shown by the dashed line. Obviously the numerical oscillations are introduced. With the ENO method, two approximate polynomials $h_{i-1,i}$ and $h_{i,i+1}$ are generated in stencils $\{i - 1, i\}$ and $\{i, i + 1\}$ respectively, and the smoother one $h_{i-1,i}$ is selected as the final approximate polynomial in the cell $i$. Similarly, for the quadratic case, the stencils $\{i - 2, i - 1, i\}, \{i - 1, i, i + 1\}$, and $\{i, i + 1, i + 2\}$ will be used to generate polynomials $h_{i-2,i-1,i}, h_{i-1,i,i+1}$ and $h_{i,i+1,i+2}$, and the smoothest one $h_{i-2,i-1,i}$ will be used as the final polynomial. Such procedure can be straightly extended to arbitrary high order case.
With ENO reconstruction, uniform accuracy can be obtained in the smooth region, and the oscillations can also be prevented by selecting the appropriate stencil. However, due to the inherent non-differentiability of this process, convergence of the solution to steady state is not possible[66]. Such drawback can be removed or reduced by using the WENO schemes. The WENO scheme was introduced in [52, 40] for the structured meshes, then was extended to the unstructured meshes in [31].

Different from the ENO scheme which just uses the smoothest polynomial among those candidates, the WENO method uses certain convex combination of all candidates. For example for the quadratic reconstruction which was described in the first paragraph of this subsection, there are three candidates, say, $h_{i-2,i-1,i}$, $h_{i-1,i,i+1}$ and $h_{i,i+1,i+2}$. According to certain criterion, three positive weights $\omega_{i-2,i-1,i}$, $\omega_{i-1,i,i+1}$ and $\omega_{i,i+1,i+2}$ can be supplied to these candidates respectively. Note that $\omega_{i-2,i-1,i} + \omega_{i-1,i,i+1} + \omega_{i,i+1,i+2} = 1$. Finally, $\sum \omega h$ is used as the approximate polynomial in the cell $i$.

In Subsections 1.3.1, 1.3.2 and 1.3.3, the solution reconstruction methods and the limiting methods are introduced. Now suppose in each cell there is an approximate polynomial, and we are ready to implement the evolution step in Algorithm 0. For
the Godunov method in [25], the Riemann problem was solved exactly. It is more efficient if the approximate Riemann Solver is used.

There are many classical approximate Riemann solvers such as the HLL[30] and HLLC Riemann solvers [81], the Riemann solver of Roe[72], and the Riemann solver of Osher [70]. The summarizations and applications of these Riemann solvers can be found in [79]. In this thesis, the HLLC Riemann solver is used.

1.3.4 The HLLC Riemann Solver

For the exact Riemann solution, there are a large amount of details, which may be either shocks or expansion waves. To solve the problem efficiently, Harten, Lax and Van Leer [30] presented the well-known HLL solver for the Riemann problem.

Suppose we have the following 1D Riemann problem

\[ U_t + F(U)_x = 0, \]
\[ U(x, 0) = \begin{cases} 
U_l, & \text{if } x < 0, \\
U_r, & \text{if } x \geq 0, 
\end{cases} \tag{1.17} \]

where \( U \) and \( F \) are selected similar to (1.9) and (1.10) respectively. If the flow is subsonic, a simple Riemann solution can be given by ignoring the contact discontinuity and assuming a single, averaged intermediate state \( U^* \), between two acoustic waves with velocities \( S_l \) and \( S_r \). The structure of the solution is demonstrated in Fig. 1.7 (left). The HLL numerical flux can be given as

\[ F_{hll} = \begin{cases} 
F_l, & \text{if } S_l > 0, \\
F^*, & \text{if } S_l \leq 0 \leq S_r, \\
F_r, & \text{if } S_r < 0, 
\end{cases} \tag{1.18} \]

where \( F^* \) is given as

\[ F^* = \frac{S_r F_l - S_l F_r + S_l S_r (U_r - U_l)}{S_r - S_l}. \tag{1.19} \]

The HLL flux satisfies the conservation laws by construction. However, the ignoring of the contact discontinuity limited its practical applications. To remedy
such drawback, Toro et al. [81] proposed the much more accurate HLLC solver for the Riemann problem. HLLC solver is a modification of the HLL solver, whereby the missing contact and shear waves are restored. The solution structure is demonstrated in Fig. 1.7 (right). The HLLC flux can be written as

$$
F_{hlle} = \begin{cases} 
F_l, & \text{if } S_l > 0, \\
F_{sl}, & \text{if } S_l \leq 0 \leq S_m, \\
F_{sr}, & \text{if } S_m \leq 0 \leq S_r, \\
F_r, & \text{if } S_r < 0,
\end{cases}
$$

(1.20)

where $S_m$ is given by

$$
S_m = \frac{p_r - p_l + \rho_l u_l (S_l - u_l) - \rho_r u_r (S_r - u_r)}{\rho_l (S_l - u_l) - \rho_r (S_r - u_r)}.
$$

(1.21)

The intermediate fluxes $F_{sl}$ and $F_{sr}$ are given by

$$
F_{sk} = F_k + S_k (U_{sk} - U_k), \quad \text{for } k = l \text{ or } r.
$$

(1.22)

where the intermediate states $U_{sk}$ are given with following form

$$
U_{sk} = \rho_k \left( \frac{S_k - u_k}{S_k - S_m} \right) \begin{bmatrix} 1 \\ S_m \\ \frac{E_k + (S_m - u_k) \left[ S_m + \frac{p_k}{\rho_k (S_k - u_k)} \right]}{\rho_k} \end{bmatrix}.
$$

(1.23)

For $S_l$ and $S_r$ in (1.18) and (1.20), the values are given based on the Roe average [20] as

$$
S_l = \min\{\lambda_1(U_l), \lambda_1(U^{Roe})\}, \\
S_r = \max\{\lambda_m(U^{Roe}), \lambda_m(U_r)\}.
$$

(1.24)
where $\lambda_1(U_l)$ is the minimum characteristic speed of the left wave, $\lambda_m(U_r)$ is the maximum characteristic speed of the right wave, and $\lambda_1(U^{Roe})$ and $\lambda_m(U^{Roe})$ are the minimum and maximum characteristic values of the Roe matrix.

The finite volume method discussed in this section will be used for the algorithm with linear reconstruction. For the quadratic reconstruction, the following residual distribution method will be used.

### 1.4 The Residual Distribution Method

The *Residual Distribution* (RD) schemes, or fluctuation splitting schemes for solving steady state hyperbolic conservation laws was first introduced by Roe, Sidilkover, Deconinck, Struijs, Bourgeois et al.[15, 73, 75] and have been widely discussed [1, 12, 14]. The framework of the residual distribution can be described as follows. For the Euler equations (1.8), steady state means that the residual of the system in every cell $K_i \in T$ should be zero, say,

$$
\sum_{e_{ij} \in \partial K_i} \int_{e_{ij}} F(U) \cdot n_{ij} dS = 0,
\tag{1.25}
$$

where $U$ and $F(U)$ are defined in (1.9) and (1.10) respectively. In practical calculations, the solution of (1.25) is obtained as the limit of the following pseudo unsteady scheme

$$
|K_i| \frac{U_i^{n+1} - U_i^n}{\Delta t} + \sum_{e_{ij} \in \partial K_i} \int_{e_{ij}} F(U_i, U_j) \cdot n_{ij} dS = 0,
\tag{1.26}
$$

where $|K_i|$ is the area of the cell $K_i$.

Similar to the finite volume methods, (1.26) can be solved by traditional techniques, and certain numerical flux such as the HLLC flux which was discussed in Subsection 1.3.4 can be used to approximate the term $F(U_i, U_j)$. However, the reconstruction step implemented in the residual distribution schemes is based on the function values on the centroid of the cell $K_i$, not the cell averages of these variables any more. For every new pseudo time level, the functional values on the centroid of the cell are updated. Finally, with vanishing the cell residuals, the accuracy can be obtained.
1.5 Temporal Discretization

So far, the spatial discretization of the Euler equations have been discussed. For the temporal discretization, there are two main methods: the *Explicit* method and *Implicit* method.

By using the explicit method, the nonlinear term in the Euler equations (1.8) can be evaluated with the information in the current time level. Furthermore, the variables in the domain can be updated cell by cell. That means there is no need to solve the linear system. So the memory storages needed by explicit method are low, and the information updating in each time level can be implemented much more efficiently compared with the implicit methods. However, based on the consideration of the stability of the algorithm, sufficiently small time step should be used, which slows down the convergence of the steady state of the algorithm significantly.

In this thesis, the implicit method is adopted, and the Newton method is used to linearize the nonlinear term in the Euler equations. Although a large size linear system needs to be solved in each Newton iteration step, the stability and the numerical accuracy of the algorithm are enhanced. To solve the linear system efficiently, the multigrid method is used, which significantly accelerates the algorithm. Besides the multigrid method, there are also many other acceleration techniques for the steady problems, which will be discussed in the next subsection.

1.6 Some Acceleration Techniques

In last several years, various techniques have been proposed in order to accelerate the algorithm to obtain the steady state solution of the Euler equations (1.8). These techniques include the local time-stepping, residual smoothing, the multigrid method, low Mach-number preconditioning, and etc. An overview of these acceleration techniques can be found in [42, 60]. In this thesis, the following two methods are adopted.
1.6.1 Local Time-Stepping

Generally, the criterion of selecting the length of the time step is to keep the stability of the algorithm. Consequently, the time step depends on the smallest cell size in the domain which also results in length of the time step a very small value. For solving the stationary problems, the temporally accurate solutions are no longer important, so the largest possible time step for each cell can be used to accelerate the convergence. This is the idea of the local time-stepping method.

Based on the understanding of local time-stepping method, a new method which depends on the local residual in each cell is used instead. It has been proven by a large amount of numerical simulations that the new method works very well. The description of such method will be given in Subsection 2.3.2 in Chapter 2.

1.6.2 The Multigrid Method

The multigrid methodology is a very powerful acceleration technique [8]. It was first proposed by Brandt [9] for solving elliptic partial differential equations. Now, it has been successfully used for solving Euler equations by Jameson [35, 34, 37], and the Navier-Stokes equations [57, 82]. The basic idea of the multigrid method is to use the coarse meshes to drive the solution on the fine meshes to the steady state faster. There are two main ingredients for implementing the multigrid method. One is the Projection Operator which is used to generate the smaller system on the coarse mesh from the system on the fine mesh. The other one is the Smoother which is used to cancel the residual on each coarse meshes.

There are two different types of multigrid methods. If the projection operator is constructed just using the system matrix itself, the method is called Algebraic Multigrid (AMG). In this thesis, the other one, say, the Geometrical Multigrid (GMG) method is used, which the projection operator is constructed based on the topological structure of the meshes. According to the agglomeration multigrid method proposed in [8] which is implemented for the cell-vertex schemes, a similar method is proposed in this thesis to generate the coarse meshes for the cell-centred meshes.
The details of implementation of the multigrid method to solve the final linear system can be found in Section 2.4 in Chapter 2.

1.7 Adaptive Methods

For obtaining the reliable numerical results of the simulations in this thesis, a very big outer boundary is selected. If the uniform meshes are used, a large amount of mesh grids will be wasted since for region nearby the outer boundary, there is almost no variation of the solutions. At the meantime for the transonic flows, there may be shocks around the inner boundary. So sufficiently small size meshes are needed to resolve the shock structure. The adaptive methods become necessary to satisfy the requirement of the meshes.

There are mainly three types of adaptive methods. The $p$-adaptive method which locally enriches the approximate polynomial; the $h$-adaptive method which locally refines and coarsens meshes; and the $r$-adaptive method which relocates the mesh grids in a mesh having a fix number of nodes. The basic idea of all these adaptive methods is to concentrate mesh grids in the region with large variation of the numerical solution. In this thesis, both $r$- and $h$-adaptive methods are introduced in the last chapter.

1.8 Outline of the Thesis

In Chapter 2, an effective and robust linear finite volume solver is proposed. The algorithm contains two main ingredients: Newton iterative scheme which is used to linearize Euler equations, and the multigrid iterative scheme which is used to solve the final linear system. The block Lower-Upper Symmetric Gauss-Seidel (LU-SGS) iterative method is used as the smoother of the multigrid method. The piecewise linear reconstruction is implemented for each cell in the domain. Two kinds of limiting strategies are adopted to adjust the gradient in each cell. One is Venkatakrishnan limiter function which is one kind of the slope limiters, while the other one is the
WENO type reconstruction. The comparison of the quality of numerical results and the implementation efficiency between two methods are demonstrated. Since the steady Euler equations are considered, the linearized system by Newton method is actually singular. To regularize this system, one kind of local time-stepping method is adopted. Different from the classical way which the local time step is used for each cell, a regularization term which is based on the local residual is presented. From the numerical simulations, we can see that the proposed regularization term works very well.

To obtain high order numerical accuracy, we introduce the quadratic reconstruction in Chapter 3. Besides the quadratic reconstruction, there are still many details we should take care to get the desired numerical accuracy such as the integral over the curved boundary and numerical quadrature formula. Different from the linear WENO reconstruction used in Chapter 2, the quadratic reconstruction is sensitive to the non-differentiability which is introduced by WENO procedure. Such non-differentiability takes difficulties to the convergence of the steady state of the algorithm. To avoid this problem, the so-called hierarchical limiting strategy is used which constrains the information level by level from the highest order terms to the lowest order terms. From the numerical results, we can see that the proposed high order numerical scheme works well: the residual of the system of all numerical simulations achieved the machine accuracy within a few Newton-iteration steps; the high order numerical accuracy is kept in the smooth solution region and at the meantime, the non-physical oscillations are removed or reduced significantly.

In fact, the outer boundaries of domains which are used in numerical simulations in Chapters 2 and 3 were selected artificially. To obtain the reliable numerical results, sufficiently large outer boundaries are expected, and the larger the domain used the better the accuracy will be. On the other hand, we also want a small domain which contains less mesh grids, which can save the CPU time required. Though the far field vortex correction technique may be used in the algorithm which uses a relatively small outer boundary, the domain is still very large. In the simulations, the grids
always extend about 20 diameters away from the airfoils. In Chapter 4, to obtain high quality of numerical results and at the same time use less CPU time, the $h$-adaptive technique is introduced in the algorithm. With the $h$-adaptive technique, the mesh around the shock or/and discontinuity region is locally refined, while the mesh in the smooth solution region is coarsened. In other words, the $h$-adaptive technique gives a much more reasonable distribution of mesh grids with the same amount of grid points. In the numerical simulations, it is found that the algorithms proposed in the previous chapters can be improved significantly with the $h$-adaptive methods.

Finally, we give the conclusion of this thesis in Chapter 5, where a brief description about the future work is also presented.
Chapter 2

The Linear Finite Volume Solver

Recently, Li et al. [50] proposed a finite volume solver for 2D steady Euler equations. In the algorithm, the Newton-iteration method is adopted to linearize the Euler equation, and in each Newton-iteration step the multigrid method with block lower-upper symmetric Gauss-Seidel (LU-SGS) iteration as its smoother is used to solve the linearized system. In the reconstruction step, the linear reconstruction is employed to describe the variation of solutions in each cell. To avoid the non-physical oscillations, the Venkatakrishnan limiter is adopted to constrain gradients during the reconstruction process.

The limiting strategy is very important for simulations with the finite volume method. A useful limiter function should be able to remove the non-physical oscillations nearby the shock profiles and can also preserve the numerical accuracy in the smooth regions. Moreover, the limiter function should not affect the convergence to the steady state. So far, many useful limiting strategies for the structured mesh have been proposed, including the total variation diminishing (TVD) limiter [26, 27], the slope limiters like minmod limiter, the superbee limiter, the MC limiter and van Leer limiter (all these limiters can be found in [45] and references therein). However, since the fixed stencil is used to approximate the variation of solutions, the numerical accuracy is always degraded when the above limiter functions are used. To preserve the numerical accuracy, the essentially non-oscillatory (ENO) method is introduced [28, 29]. In order to reconstruct the approximate polynomial
of the solutions on each cell, the ENO methods test different neighboring stencils so that the locally smoothest stencils can be selected eventually. By selecting a convex combination of the results obtained from all possible stencils, the weighted essentially non-oscillatory (WENO) methods were proposed, see, e.g., [40, 51, 74]. These limiter functions yield satisfactory numerical results on structured meshes.

On the unstructured meshes, one of the classical ways to obtain high resolution results is to use the $k$-exact reconstruction [2, 67] together with a slope limiter. For example, for the linear case, it is assumed that the solution is piecewise linearly distributed over the cell. Such linear approximation is determined by solving a least square system based on cell averages of the cell and its neighbours. After that, certain slope limiter is used to guarantee the monotonicity of the solutions. On the unstructured meshes, the first implementation of a limiter function was presented by Barth and Jespersen [4]. The Barth and Jespersen limiter is used to enforce a monotone solution. However, their method is rather dissipative which leads to smear discontinuities. Furthermore, the limiter may be active in smooth flow regions due to the numerical noise, which causes difficulties of the convergence of the steady state [8]. To improve the differentiability of the limiter in [4], the Venkatakrishnan limiter was proposed in [83] and has been widely used. Similar to the structured mesh case, the theoretically predicted accuracy also can not be guaranteed with the fixed stencil when these limiters are used. Moreover, since the Venkatakrishnan limiter does not preserve strict monotonicity, slight oscillations can be observed near shock discontinuities. To further improve the quality of numerical solutions for the unstructured meshes, the ENO/WENO type reconstructions may be considered due to their good performance on the structured meshes.

In [31], Hu and Shu implemented the WENO type finite volume schemes on the unstructured meshes. In [55, 56, 97], the WENO method was adopted as the limiting strategy to the discontinuous Galerkin method. With the help of the WENO method, not only desired numerical accuracy in the smooth region is preserved but also the non-oscillatory sharp shock profiles are obtained. However, as the differen-
iability of the numerical schemes was affected with the use of the WENO methods, the convergence to steady state solutions is found unsatisfactory, see, e.g., [56, 97]. In [97], the WENO type limiting procedure was only implemented in the ”trouble cells”; this may also make the numerical scheme less smooth (i.e., the differentiability becomes worse). As a result, it was observed in [97] that the residual of the system can not be reduced to the machine accuracy. Moreover, the situation of the third-order case was worse than that of the second-order case. Recently, Liu et al. [53, 54] and Xu et al. [89] proposed the hierarchical WENO reconstruction methods, which are adopted by Hu et al. [32] to design high-order residual distribution (RD) schemes for solving the steady Euler equations. It is seen in [32] that the hierarchical WENO reconstruction can improve the differentiability of the numerical schemes so that the system residual resulting from the discretization of the steady Euler equations can reach machine accuracy in a few Newton-iteration steps. However, the hierarchical WENO reconstruction can not guarantee the monotonicity of numerical solutions. Consequently, slight oscillations were observed nearby the shock discontinuities.

In this paper, we will present a robust and effective finite volume solver based on the linear WENO reconstruction to solve the steady Euler equations. The algorithm is based on the solver proposed in [50], where as the Venkatakrishnan limiter was used to constrain gradients on each cell the algorithm suffers from the problems mentioned above. To obtain the high quality of numerical solutions, the WENO type reconstruction will be used in this paper. In [50], a very large reconstruction patch for linear case is adopted to ensure the stability. The principle for choosing patch \( P(\mathcal{K}) \) is as following. For a cell \( \mathcal{K} \), the cell which has one common vertex with \( \mathcal{K} \) is chosen as one component of the reconstruction patch of \( \mathcal{K} \). In this paper, based on the consideration of the algorithm efficiency, a much smaller patch is used. More precisely, for the cell \( \mathcal{K} \), the cell has one common edge with \( \mathcal{K} \) is chosen to be one component of the reconstruction patch of \( \mathcal{K} \). Then with cell averages of those cells in the patch, the linear reconstruction is implemented for each cell \( \mathcal{K} \in P(\mathcal{K}) \).
looping all reconstruction patches in the meshes, each cell has 3 or 4 approximate polynomials. The final approximate polynomial in each cell is given by the convex combination of these candidates according to certain smoothness indicator. It is observed from the numerical experiments that the proposed linear finite volume solver is not sensitive to the influence on the differentiability of numerical scheme which is introduced by the WENO method. The system residuals of all numerical experiments can achieve machine accuracy in a few Newton-iteration steps. Besides the convergence, the proposed solver can keep desired numerical accuracy in the smooth region, and at the same time remove spurious oscillations nearby the shock and/or discontinuity region. It is shown in the final section that the quality of numerical results of the linear case is improved significantly in comparison with that in [50]. Since the smaller patch is used and the behavior of Newton iteration is improved, the increment of CPU time is not significant compared with the algorithm with the Venkatakrishnan limiter.

In the rest of this chapter, the numerical discretization for Euler equations will be described in the next section. In Section 2.2, the linear reconstruction with the Venkatakrishnan limiter and the WENO reconstruction will be introduced respectively. Then the Newton-iteration and the multigrid method for solving the corresponding nonlinear system will be discussed in Sections 2.3 and 2.4. The boundary conditions which are used in the simulations will be simply discussed in Section 2.5. Numerical experiments will be carried out in Section 2.6. Finally, the conclusions of this chapter will be given in Section 2.7.

2.1 The Finite Volume Discretization for the Steady Euler Equations

In the numerical simulations, the system (1.8) is solved in domain \( \Omega := \mathbb{R}^2 - \Omega_c \), where \( \Omega_c \) denotes the domain occupied by the airfoil, which is the body of the aircraft in the two-dimensional case. Since the physical domain is unbounded, the commonly
used strategy is to solve the problem in the domain $\bar{\Omega} := \Omega \cap \{ x < R \}$ and then adopt the far field vortex correction technique to remedy the error introduced by the abrupt domain truncation.

The Cell-centered scheme is adopted to discretize the continuous equation. Let $\mathcal{T}$ be a triangular partition of $\bar{\Omega}$, and $K \in \mathcal{T}$ be one cell in the partition. We assume the intersection of two different cells can only be either an edge or a vertex. Let $n_{ij}$ denote the outer unit normal on the edge $e_{ij}$, pointing from $K_i$ to $K_j$. Since we just focus on the steady state of the Euler equations, the Euler equations (1.8) becomes

$$\oint_{\partial K_i} F(U) \cdot n dS = 0,$$

(2.1)

where $n$ is the unit out normal of $\partial K_i$. By introducing the numerical flux, (2.1) can be approximated as

$$\oint_{\partial K_i} F(U) \cdot n dS \approx \sum_{e_{ij} \in \partial K_i} \int_{e_{ij}} \bar{F}(U_i, U_j) \cdot n_{ij} dS = 0,$$

(2.2)

where $\bar{F}(U_i, U_j)$ denotes the numerical flux, and certain Riemann solver can be used to give its value. Many excellent Riemann solvers have been proposed such as LF solver, Roe solver, Osher solver, HLL solver and its improvement version HLLC [78] solver. For details of these Riemann solvers, we refer to [80] and references therein. In our algorithm, HLLC solver is adopted.

### 2.2 The Reconstruction and Limiting Strategies

The classical Godunov scheme assumed that in each cell, the conserved variable is a constant, which of course is equal to the cell average. This assumption makes Godunov scheme be one of the first order schemes which are too dissipative. To get the second order accuracy, the linear distribution of solutions on each cell should be determined. In this section, the linear reconstruction used in [50] will be summarized first, and then two different limiting strategies are presented respectively later.
2.2.1 The Linear Reconstruction

There are two classical ways to find out the approximate gradient in each cell. One is Green-Gauss approach which approximates the gradient of function $U$ as the surface integral of the product of $U$ with an outward-pointing unit normal vector over control volume $K$, say,

$$\nabla U \approx \frac{1}{|K|} \oint_K U n dS. \quad (2.3)$$

The other one is the least square method which the gradient is reconstructed using the information of all available nearest neighbours. In contrast to Green-Gauss approach, the least square method is more suitable and reliable even on highly stretched meshes. In the rest of this subsection, we describe the linear reconstruction with the least square method.

In fact, the linear reconstruction used in this subsection belongs to category of the $k$-exact reconstruction. For the detailed description of the $k$-exact reconstruction, we refer to [2] and references therein. In this section, we just give the implementation of the $k$-exact reconstruction for the linear case.

First, we use the following principle to choose the reconstruction patch for the cell $K$: the cell which has one common edge with the cell $K$ is selected to be one component of the patch $P_K$. The demonstration of linear reconstruction patch $P(K) = \{K, K_0, K_1, K_2\}$ is shown in Fig. 2.1 (left).

Expanding the approximate function $h(x, y)$ at the centroid $(x_K, y_K)$ of the cell $K$ by using Taylor expansion, and truncating after the linear terms, we get the following approximate polynomial:

$$h(x, y) \approx a_0 + a_1(x - x_K) + a_2(y - y_K). \quad (2.4)$$

The classical $k$-exact reconstruction actually minimize the difference of the cell average of the reconstructed function $h(x, y)$ and the original function $U$ over each cell in $P_K$. However, there is the following relation for the linear function $h(x, y)$ in the cell $K$: the cell average of the function $h(x, y)$ is equal to its functional value on the centroid of the cell. That means if the cell average is conserved during the
reconstruction, $a_0$ in (2.4) will be known in advance, and equal to the cell average over the cell $K$. So the problem becomes the minimization of the difference between the functional values on the centroid of neighbours $K_0$, $K_1$ and $K_2$ and the original values $U$ on the centroid of neighbours. The problem is written as

$$\min_{a_1,a_2} \sum_{\forall K_i \in P(K)} \left\| \frac{a_{0,l} - a_0 - a_1 x_{K,l} - a_2 y_{K,l}}{d_{K,l}} \right\|^2,$$

(2.5)

where $a_{0,l}$ is the cell average over the cell $K_i$, $l = 0, 1, 2$, and $x_{K,l} = (x_l - x_K)$, $y_{K,l} = (y_l - y_K)$. $d_{K,l}$ means the length between the centroids of the cell $K_i$ and the cell $K$.

By solving (2.5), we can get the linear distribution of the numerical solutions in the cell $K$.

For all reconstruction methods which are higher than first order, a powerful limiting strategy is necessary for preventing the non-physical oscillations. Limiting procedure may also enhances the robustness of the algorithm with improving the poor gradient reconstruction accuracy caused by the lack of sufficient grid quality. In the next two subsections, two different limiting methods are introduced respectively.

### 2.2.2 The Venkatakrishnan Limiter

In [4], Barth and Jespersen gave the first implementation of a limiter function on the unstructured mesh. The sufficient condition to avoid non-physical oscillations is
that no new local extrema are formed during reconstruction. Barth and Jespersen limiter achieves this purpose by introducing a value $\Phi$ in each cell to limit the gradient which is obtained from reconstruction. More precisely,

$$h^*(x, y) = a_0 + \Phi(a_1(x - x_K) + a_2(y - y_K)), \quad \Phi \in [0, 1],$$

will be used as the final approximate polynomial in the cell $K$. Barth and Jespersen used the following function to determine the value $\Phi$ to prevent the formation of the local extrema at the Gauss quadrature points $(x_l, y_l)$ of the flux integration on the edge,

$$\Phi_l = \begin{cases} 
\min(1, \Delta_{1,max} / \Delta_2), & \text{if } \Delta_2 > 0 \\
\min(1, \Delta_{1,min} / \Delta_2), & \text{if } \Delta_2 < 0 \\
1, & \text{if } \Delta_2 = 0,
\end{cases}$$

where $\Delta_{1,max} = \max(a_{0,j} - a_0)$ and $\Delta_{1,min} = \min(a_{0,j} - a_0)$ are respectively the largest positive and negative difference of the solution values on the centroids of the neighbours in $K_j \in \mathcal{P}_K$ and the current cell $K$. $\Delta_2 = a_1(x_l - x_K) + a_2(y_l - y_K)$ is the difference between the unconstrained function value at Gauss quadrature point $(x_l, y_l)$ and the centroid of the cell $K$. Finally, $\Phi$ is determined after looping all the Gauss quadrature points in the cell $K$ with

$$\Phi = \min(\Phi_l).$$

In practice, Barth and Jespersen limiter removed the non-physical oscillations effectively. However, since the non-differentiability is introduced to the numerical schemes with Barth limiter, the convergence of numerical schemes to the steady state is adversely affected.

To enhance the differentiability of the limiting method, Venkatakrishnan introduced a smooth function as

$$f(x) = \frac{x^2 + 2x}{x^2 + x + 2}$$

(2.9)

to replace the function $\min(\cdot, \cdot)$ in (2.7). The difference between two functions can be seen from Fig. 2.2.2.
Figure 2.2: The $\min(1, x)$ function used in Barth limiter, and its smooth Venkatakrishnan approximation.

With this modification, the differentiability of the limiting method is improved. However, it is found that the modified limiter is enacted even in certain region with smooth solution. Consequently, the numerical accuracy would be degraded. Finally, Venkatakrishnan gave the further modification of the limiter to eliminate the effect of the limiter when $h_j - a_0 \leq (K\Delta h)^{1.5}$, where $K$ is a parameter and $\Delta h$ is the diameter of the cell $\mathcal{K}$. The modified $\Phi_j$ is given as

$$\Phi_j = \begin{cases} 
\frac{\Delta_{1,max} + \epsilon^2 + 2\Delta_{1,max}\Delta_2}{\Delta_{1,max}^2 + 2\Delta_2^2 + \Delta_{1,max}\Delta_2 + \epsilon^2}, & \text{if} \quad \Delta_2 > 0 \\
\frac{\Delta_{1,min} + \epsilon^2 + 2\Delta_{1,min}\Delta_2}{\Delta_{1,min}^2 + 2\Delta_2^2 + \Delta_{1,min}\Delta_2 + \epsilon^2}, & \text{if} \quad \Delta_2 < 0 \\
1, & \text{if} \quad \Delta_2 = 0 
\end{cases}$$

where $\epsilon^2 = (K\Delta h)^3$.

Venkatakrishnan limiter demonstrated the superior convergence properties and has been widely used. However, Venkatakrishnan limiter can not guarantee the strict monotonicity of the results with $K > 0$. In other word, the slight oscillations may still be observed around the shocks. Furthermore, the limiting strategy proposed in this subsection actually reduces the gradient obtained with the fixed stencil $\{\mathcal{K}, \mathcal{K}_0, \mathcal{K}_1, \mathcal{K}_2\}$, it can be predicted that the numerical accuracy will be degraded by using Venkatakrishnan limiter. To fix these problems, the WENO type reconstruction will be introduced in the next subsection.
2.2.3 The WENO Reconstruction

To implement the WENO reconstruction for the cell $\mathcal{K}$, first we need to select different reconstruction patches. In this paper, the patches $\{\mathcal{K}, \mathcal{K}_0, \mathcal{K}_1, \mathcal{K}_2\}$, $\{\mathcal{K}, \mathcal{K}_0, \mathcal{K}_{0,0}, \mathcal{K}_{0,1}\}$, $\{\mathcal{K}, \mathcal{K}_1, \mathcal{K}_{1,0}, \mathcal{K}_{1,1}\}$, and $\{\mathcal{K}, \mathcal{K}_2, \mathcal{K}_{2,0}, \mathcal{K}_{2,1}\}$ are selected for the cell $\mathcal{K}$, which are shown in Fig. 2.1 (right).

For each patch, the reconstruction procedure described in the last subsection is implemented to generate the approximate linear function $h(x, y)$ in the cell $\mathcal{K}$. After that, there are four linear functions $h_i(x, y), i = 0, 1, 2, 3$ as candidates for the cell $\mathcal{K}$.

For the ENO schemes, the smoothest one of four candidates $h_i(x, y), i = 0, 1, 2, 3$ is selected to be the approximate polynomial in the cell $\mathcal{K}$. Such operation gives sharp shock profiles, and uniform numerical accuracy in the smooth region is kept very well. However, its inherent non-differentiability prevent the convergence to the steady state. So the ENO schemes are not suitable for simulating the steady problems. Fortunately, the WENO schemes improve the differentiability of the numerical schemes, in which the final approximate polynomial is given by certain convex combination of all the candidates. To give the implementation of the WENO reconstruction, first we follow [31] to introduce the smoothness indicator for the polynomial $h(x, y)$ on the cell $\mathcal{K}$ as

$$S = \sum_{1 \leq |\alpha| \leq k} |_{\mathcal{K}} |^{|\alpha|-1} (D^\alpha h(x, y))^2 d\Omega,$$  \hspace{1cm} (2.11)

where $\alpha$ is a multi-index and $D$ is the derivative operator.

Now we need to give the weight of each candidate of the polynomials based on the smoothness indicator. For the candidate $h_j(x, y)$, the weight is

$$\omega_j = \frac{\tilde{\omega}_j}{\sum_{i=0}^{3} \tilde{\omega}_i}, \quad \tilde{\omega}_i = \frac{1}{(\epsilon + S_i)^\beta},$$ \hspace{1cm} (2.12)

where $\epsilon$ is a positive parameter. We follow [31] to choose $\epsilon = 10^{-4}$ and $\beta = 2$ in the implementation.
Figure 2.3: Results of the mesh-refinement study for the linear reconstruction without limiting procedure, the linear WENO reconstruction and the linear reconstruction with Venkatakrishnan limiter. The initial mesh contains 64 cells, and the reconstruction is implemented on six successively refined meshes.

The final approximate polynomial for the cell $K$ is

$$h^*(x, y) = \sum_{i=0}^{3} \omega_i h_i(x, y).$$

(2.13)

Since the cell average is conserved by each candidate in the cell $K$, and the final approximate polynomial is actually the convex combination of these candidates with the summation of the weights to be 1, the final polynomial also conserves the cell average in the cell $K$.

The convergence of the linear WENO reconstruction is studied by using the smooth function $f = \sin(\pi x) \cos(2\pi y)$ on the domain $[0, 1] \times [0, 1]$. The mesh is generated by EasyMesh [68]. The reconstruction is implemented on six successively refined meshes, and the results of $L_2$ norm of the error between the reconstructed values and the exact values in each cell are shown in Fig. 2.3. From results, we can see that the linear reconstruction with Venkatakrishnan limiter degrades the convergence order, while the linear WENO reconstruction worked very well: even higher convergence order than the theoretical one is obtained.

Note that if the criterion to determine the reconstruction patch of the linear reconstruction is carried out strictly, the patch of the cell adjacent to the boundary will be smaller than that of the cell inside the domain. To keep the numerical accu-
racy and enhance the stability of the algorithm, the patch \( \mathcal{P}(\mathcal{K}) \) for the boundary cell will be enlarged by using the cell which has the common vertex with the cell \( \mathcal{K} \).

### 2.3 Newton Iteration

If the system (2.2) is solved implicitly, the evaluation of the numerical flux \( \overline{F}(U_i, U_j) \) needs to be given based on the unknown variables \( U_i \) and \( U_j \). Obviously it is impossible. We can use Newton method to linearize the nonlinear system (2.2) as:

\[
\sum_{e_{ij} \in \partial \mathcal{K}_i} \int_{e_{ij}} \overline{F}^{(n)} \cdot n_{ij} dS + \sum_{e_{ij} \in \partial \mathcal{K}_i} \int_{e_{ij}} \left( \frac{\partial \overline{F}^{(n)}}{\partial U_i} \delta U_i^{(n)} \right) \cdot n_{ij} dS + \sum_{e_{ij} \in \partial \mathcal{K}_i} \int_{e_{ij}} \left( \frac{\partial \overline{F}^{(n)}}{\partial U_j} \delta U_j^{(n)} \right) \cdot n_{ij} dS = 0,
\]

where \( \overline{F}^{(n)} = \overline{F}(U_i^{(n)}, U_j^{(n)}) \), and the terms \( \frac{\partial \overline{F}^{(n)}}{\partial U_i} \) and \( \frac{\partial \overline{F}^{(n)}}{\partial U_j} \) are refered as the Flux Jacobians. After each iteration, the \( U_i \) is updated by

\[
U_i^{(n+1)} = U_i^{(n)} + \tau_i \delta U_i^{(n)},
\]

where \( \tau_i \) is a relaxation parameter on \( \mathcal{K}_i \).

#### 2.3.1 The Jacobian of the Numerical Flux

The calculation of the flux Jacobian \( \frac{\partial \overline{F}}{\partial U_i} \) has been discussed extensively in the references. Often the derivatives are obtained by using the chain rule, which may lead to long and tedious formulas. It is in general difficult to get the exact expression of these derivatives, especially when the boundary condition is evolved. Even these approximated derivatives are often so complex that they give rise to serious difficulties in the implementation. As an alternative, we approximate the derivatives using numerical differentiation, which makes the algorithm converge successfully. The Jacobians are based on the data at the Gauss points on the boundary of each cell. First, the piecewise polynomial is reconstructed on each cell, then we can obtain the left and right states of the conserved variables on each edge, as well as the numerical flux of this edge. Finally, the numerical differentiation \( \frac{\partial \overline{F}}{\partial U} \) can be
obtained by perturbing the left or right state of the conserved variables a little bit.
More precisely, we use the following forms to approximate $\frac{\partial \bar{F}}{\partial U_i}$ and $\frac{\partial \bar{F}}{\partial U_j}$,

$$
\frac{\partial \bar{F}_l}{\partial U_{i,m}} \approx \frac{\bar{F}_l(U_{i,m}^{(n)} + \varepsilon \delta U_{i,m,j}^{(n)} - \bar{F}_l(U_{i,m,j}^{(n)})}{|\varepsilon \delta U_{i,m}|},
$$

$$
\frac{\partial \bar{F}_l}{\partial U_{j,m}} \approx \frac{\bar{F}_l(U_{i,m,j}^{(n)} + \varepsilon \delta U_{i,m,j}^{(n)} - \bar{F}_l(U_{i,m,j}^{(n)})}{|\varepsilon \delta U_{i,m}|},
$$

where $\frac{\partial \bar{F}_l}{\partial U_{i,m}}$ means the the derivative of the $l$-th element of vector $\bar{F}$ respect to the $m$-th element of vector $U_i$, and $\varepsilon$ is chosen as $10^{-6}$ in our implementation which is about half-word length of the machine epsilon.

### 2.3.2 Regularization

Without the time evolving term, (2.14) is in fact a singular system. The popular way to regularize (2.14) is to add an artificial time derivative term to the Jacobian matrix. The term can be written as

$$
\int_{\Omega_i} \frac{\delta U_i^{(n)}}{\Delta t_i} d\Omega,
$$

(2.16)

where $\Delta t_i = CFL \Delta h_i / \lambda^+$, and $CFL$ is the local CFL number, $\Delta h_i$ is the diameter of the cell $\Omega_i$, $\lambda^+$ is the maximal local propagation speed of the system on $\Omega_i$.

For time-dependent problems, $CFL$ number should be chosen as about $O(1)$ to make sure the length of the time step is small enough. Then the reliable intermediate state of solutions in each time level could be obtained. But for the steady problems, the intermediate state of solutions is no longer important. All we care about is the final steady state of the flows. In other word, the state of solutions when $t \to \infty$ is our aim. So only if the iteration method can converge, the $CFL$ number can be chosen as large as possible to accelerate the convergence rate. Generally, the $CFL$ should be chosen as about $O(1)$ at the beginning time of implementation of the whole algorithm. When the residual of the system begin to reduce, the $CFL$ number should be increased dynamically to improve the efficiency.
Based on the understanding of the local time step choosing strategy, it is found in [50] that the local residual

\[ R_i^{(n)} = \sum_{e_{jk} \in K_i} \int_{e_{jk}} \bar{F}^{(n)} \cdot n_{jk} dS, \]  

(2.17)

of the system in the cell \( K_i \) could be used to quantify locally how close to the steady state the flow field is. That is, at the beginning time, since the initial guess of the solutions may be far away from the exact solutions, the local residual in each cell is big. When the algorithm starts converging, the state of solutions is close to the steady state, so the local residual in each cell becomes smaller and smaller. Note that the same scaling of a norm of the local residual and local mesh size, the regularization term can be simply set as a constant times of the \( L_1 \) norm of the local residual. Then the regularized system (2.14) becomes

\[
\alpha \| R_i^{(n)} \|_{l_1} \delta U_i^{(n)} + \sum_{e_{ij} \in K_i} \int_{e_{ij}} \left( \frac{\partial \bar{F}^{(n)}}{\partial U_i} \delta U_i^{(n)} \right) \cdot n_{ij} dS \\
+ \sum_{e_{ij} \in K_i} \int_{e_{ij}} \left( \frac{\partial \bar{F}^{(n)}}{\partial U_j} \delta U_j^{(n)} \right) \cdot n_{ij} dS = -R_i^{(n)},
\]  

(2.18)

where \( \alpha \) is a positive constant.

In the numerical experiments, we can see that the algorithm with such regularization term is very robust. For all the numerical simulations with different free-stream configuration and different geometrical configurations, the algorithm can converge successfully just with one fixed \( \alpha \).

The flow chart of the Newton-iteration algorithm is given below.

**Algorithm 1: [Newton-iteration]**

1. Let \( U^{(0)} \) be initial value, and let \( n = 0 \);
2. Solve linear system (2.18) to get \( \delta U^{(n)} \);
3. Update \( U_i^{(n+1)} \) by using \( U_i^{(n+1)} = U_i^{(n)} + \tau_i \delta U_i^{(n)} \) in every cell;
4. Reconstruct \( U^{(n+1)} \) on each cell;
5. Check if the residual \( R^{(n+1)} \) is small enough: if yes, stop; otherwise, set \( n = n + 1 \) and go to step 2.
2.4 Multigrid Method

The implicit method will result in a linear system with a large and sparse matrix. The multigrid method will be used to solve this system, which can dramatically improve the efficiency of the algorithm. Since Brandt [9] originally developed the multigrid method for the elliptic partial differential equations, a lot of work has been done around this topic. Jameson [34, 35, 36, 37] applied the multigrid method to solve Euler equations. Then it was employed to solve Navier-Stokes equations in [62, 64].

The multigrid method mainly uses the coarse mesh in order to drive the solution on the finest mesh faster to the steady state. This idea came from the discovery that common iterative methods such as Jacobian iterative method and Gauss-Seidel iterative method reduce the high-frequency components of solutions error efficiently, while the low-frequency parts are usually damped out very slow. Consequently, this resulted in very low efficiency for the algorithm to converge to the steady state. The multigrid method uses this property to accelerate the convergence with the following strategy. First, certain iterative scheme is implemented a few times in the fine mesh to cancel the high-frequency components of solutions error. Then certain projection operator is used to generate solutions on coarse mesh from solutions on the fine mesh, and the iterative method is implemented a few times. Since on the coarse mesh, the low-frequency parts of solutions error becomes the high-frequency parts, the error will be reduced efficiently. Finally, the corrections obtained on the coarse mesh are added back to the solutions on the fine mesh. In the practices, several levels of coarse meshes are used to reduce different frequency components of solutions error. With such operation, the whole error is reduced very quickly, and convergence is accelerated significantly.

Based on the above description, we can see that there are two main ingredients in the multigrid method: The Projection Operator and the Smoother. The projection operator is used to project the linear system from the fine mesh to the coarse mesh, and the smoother is certain iterative method which is used to damp out the solu-
tions error. The quality of these two operations determines the quality of multigrid method.

2.4.1 The Projection Operator

For constructing the high quality of projection operator, the key is to generate the high quality of the coarse mesh from the fine mesh. It is straightforward for the structured meshes since the coarse meshes can be obtained easily by deleting every second grid line in the respective coordinate direction. So the diameters of each cell in each level of the coarse meshes are $2h$, $4h$, etc., where $h$ is the diameter of the cell in the finest mesh. Fig. 2.4 shows the fine mesh and the coarse meshes for the structured meshes case.

![Figure 2.4: Representation of a fine mesh with meshsize $h$, and of two coarse meshes with meshsizes $2h$ and $4h$ respectively for the structured meshes case.](image)

Compared with the concise strategy for generating coarse meshes for structured case, the strategy for the unstructured meshes is much more complicated. To implement the multigrid method effectively for the unstructured mesh case, the coarse meshes generated from the fine mesh should be uniform. Furthermore, the ratio of the cell volumes of the coarse to the fine mesh should be kept around certain constant (4 for 2D case, and 8 for 3D case). In spite of the difficulty of generating coarse mesh for unstructured mesh case, many excellent methods have been proposed such as the nonested-grids methods [59, 63], the topological methods [69, 3]
and the agglomeration of control volumes method [43, 41]. Among these methods, the agglomeration method is a very popular one because of its high efficiency. The agglomeration method generates a coarse mesh by fusing the control volume of the fine mesh with their neighbours. There are many implementations of the agglomeration methods on the cell-vertex schemes. In this thesis, we follow the method for cell-vertex scheme which is proposed in [8] to give a similar implementation for the cell-centred scheme:

Algorithm 2: [Agglomeration method for cell-centred scheme]

1. Loop over all control volumes in the current mesh, and build a list of the so-called seed control volumes. Only control volume which has at least 3 (for 2D case) unagglomerated neighbours can be selected as a seed control volume;
2. Loop over all seed control volumes, agglomerate the current seed control volume and its three “nearest” neighbours as one big control volume of the coarse mesh. The “nearest” neighbour means the adjacent control volume which has the most common edges with the seed control volume;
3. Loop all control volumes in the current mesh, and eliminate the unagglomerated control volumes. After implementing Step 1 and 2, there may be some unagglomerated control volumes left. If the control volume is adjacent to some big agglomerated control volumes, merge it to the "nearest" one. Otherwise, agglomerate this control volume and all its unagglomerated neighbours as a new big control volume of the coarse mesh;
4. Repeat Step 3 until there is no unagglomerated control volume in the current mesh;

Algorithm 2 is repeated until all the coarse meshes are generated. Fig. 2.5 shows the element patches generated on four continuous levels with Algorithm 2. The projected linear system on each level of the coarse mesh can be obtained based on the relation between the current mesh and finer mesh. The mathematical description of the projection procedure is given as below.
Figure 2.5: Element patches near the airfoil body of NACA 0012 generated by the aggregation from a quasi-uniform mesh. It shows the element patches on four continuous levels.

We denote the original partition on $\Omega$ as $T_0 = T$. The implementation of Algorithm 2 will give a sequence of meshes denoted as $T_m$, $m = 1, 2, \ldots$. For every cell $K_{i,m+1} \in T_{m+1}$, it is a union of some cells in mesh $T_m$:

$$K_{i,m+1} = \bigcap_{j \in I(i,m+1)} K_{j,m},$$  \hspace{1cm} (2.19)

where $I(i, m+1)$ is the set of indices for cells in $T_m$ as a part of cell $K_{i,m+1}$.

For the linear system (2.18), it can be reformulated as the following expression:

$$\sum_j A_{ij,0} \delta U_{i,0} = -R_{i,0},$$  \hspace{1cm} (2.20)
where

\[ A_{ii,0} = \alpha \| R_i^{(n)} \|_1 + \sum_{e_{ij} \in K_i} \int_{e_{ij}} \frac{\partial \bar{F}^{(n)}}{\partial U_i} \cdot \mathbf{n}_{ij} dS, \]

\[ A_{ij,0} = \sum_{e_{ij} \in K_i} \int_{e_{ij}} \frac{\partial \bar{F}^{(n)}}{\partial U_j} \cdot \mathbf{n}_{ij} dS, \quad i \neq j, \]

\[ R_{i,0} = R_i^{(n)}, \]

\[ \delta U_{i,0} = \delta U_i^{(n)}. \]

Then the projected linear system on mesh level \( m + 1 \) from level \( m \) is given as

\[ \sum_j A_{ij,m+1} \delta U_{i,m+1} = -R_{i,m+1}, \quad (2.21) \]

where

\[ A_{ij,m+1} = \sum_{\xi \in I(i,m+1)} \sum_{\eta \in I(j,m+1)} A_{\xi\eta,m}, \quad (2.22) \]

\[ R_{i,m+1} = \sum_{j \in I(i,m+1)} (R_{j,m} + \sum_\xi A_{j\xi,m} \delta U_{\xi,m}). \quad (2.23) \]

As the corrections obtained from the mesh level \( m + 1 \), \( \delta U_{i,m+1} \) is added back to the solution on the mesh level \( m \) as

\[ \delta U_{j,m} \rightarrow \delta U_{j,m} + \delta U_{i,m+1}, \quad \text{if} \ j \in I(i,m+1). \quad (2.24) \]

### 2.4.2 The Smoother

The other important component of the multigrid method is the smoother. For the unstructured mesh case, the implicit *Lower-Upper Symmetric Gauss-Seidel* (LU-SGS) method has been widely used [7, 71, 95] since it was proposed by Yoon and Jameson [92, 93, 94]. Compared with other iterative scheme, LU-SGS scheme has lower numerical complexity, and its memory requirements are modest. Furthermore, it can be used on both structured and unstructured meshes, and can be implemented easily on parallel computers. Based on the LU-SGS scheme, Chen and Wang [11] presented the *Block LU-SGS* (BLU-SGS) scheme to solve the linearized Jacobian matrix block by block, and showed impressive results. In our algorithm, the BLU-SGS scheme is adopted as the smoother of the multigrid method. The procedure
of implementation of BLU-SGS scheme on mesh level $T_m$ can be formulated as two symmetric loops:

1. For $K_{i,m} \in T_m$, loop for $i$ increasing
   \[
   \delta U_{i,m} := A^{-1}_{ii,m}(R_{i,m} - \sum_{i \neq j} A_{ij,m} \delta U_{j,m});
   \] (2.25)

2. For $K_{i,m} \in T_m$, loop for $i$ decreasing
   \[
   \delta U_{i,m} := A^{-1}_{ii,m}(R_{i,m} - \sum_{i \neq j} A_{ij,m} \delta U_{j,m}).
   \] (2.26)

After updating the information for the cell $K_i$, the system (2.18) is solved by a direct LU decomposition solver.

Finally, the $V$-cycle type iteration is adopted in the implementation of our linear multigrid solver. For getting better efficiency, the BLU-SGS method is implemented symmetrically before the projection and after the coarse mesh corrections.

The algorithm of the $V$-cycle agglomeration multigrid method for solving the linear system is given as

<table>
<thead>
<tr>
<th>Algorithm 3: [Multigrid method]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Let $m = 0$;</td>
</tr>
<tr>
<td>2. For $\sum_j A_{ij,m} \delta U_{i,m} = -R_{i,m}$, iterate for 2 or 3 times by using BLU-SGS method;</td>
</tr>
<tr>
<td>3. Check if the mesh $T_m$ is the coarsest mesh, if yes, go to Step 4. Otherwise, get the linear system (2.21) on the mesh $T_{m+1}$ and let $m = m + 1$, then go to Step 2;</td>
</tr>
<tr>
<td>4. Implement (2.24) to add corrections $\delta U_{i,m}$ obtained from the mesh level $m$ back to the solution on the mesh level $m - 1$, and iterate system $\sum_j A_{ij,m-1} \delta U_{i,m-1} = -R_{i,m-1}$ for 2 or 3 times by using block LU-SGS method;</td>
</tr>
<tr>
<td>5. Check if the mesh $T_{m-1}$ is the finest one, if yes, stop. Otherwise, let $m = m - 1$, and go to Step 4;</td>
</tr>
</tbody>
</table>
The multigrid algorithm, as the inner iterations of the complete Newton-iteration in Algorithm 1, is used for only a few steps in each Newton-iteration. According to the numerical experiences, too many iterations in the multigrid method will make the solver inefficient, especially when the Mach number is bigger than 0.9. For the low Mach number case, increasing number of iterations appropriately in the multigrid method can decrease the number of iterations of the outer Newton method. In the numerical simulations, we just set the number of iterations in the multigrid to be 2, and it works very well for all numerical simulations in this thesis.

2.5 Boundary Condition

As we mentioned in Section 2.1, the computational domain we used is $\Omega := \mathbb{R}^2 - \Omega_c$, where $\Omega_c$ denotes the domain occupied by the airfoil, which is the body of the aircraft in the two-dimensional case.

In practice, the popular way to cope with the unbounded domain $\Omega$ is to use the bounded domain $\bar{\Omega} := \Omega \cap \{|x| < R\}$ instead, then remedy the truncation error by using the far field vortex correction technique. For the detail of this technique, we refer to [10] and references therein.

In the numerical experiments, there are two main boundaries we need to take care: The farfield boundary and the solid wall boundary. A typical demonstration of the computational domain is shown in Fig. 2.5.

For a detailed description for the boundary conditions, we refer to [8] and references therein. In this section, we just list the boundary conditions which are used in our simulations.

2.5.1 Solid Wall Boundary Condition

For the inviscid flows, the fluid just slip over the boundary. So the following formula is adopted to serve as the boundary condition for the velocity:

$$v \cdot n = 0,$$  \hspace{1cm} (2.27)
where $\mathbf{v}$ is the velocity of the flow on the boundary, and $\mathbf{n}$ is the unit outer normal vector on the boundary.

In the implementation, the triangular mesh are used. So one layer of dummy cells are employed outside the boundary followed the suggestion in [22, 21]. The velocity in the dummy cell is set to be

$$\mathbf{v}_d = \mathbf{v}_b - 2(\mathbf{v}_b \cdot \mathbf{n})\mathbf{n},$$

(2.28)

where $\mathbf{v}_b$ means the velocity in the boundary cell, and $\mathbf{v}_d$ is the velocity in the corresponding dummy cell. For the other variables such as the density, the total energy and the pressure on the dummy cell, we just set them equal to the values in the corresponding boundary cell respectively.

### 2.5.2 Farfield Boundary Condition

The derivation of farfield boundary condition is based on the concept of characteristic variables as it was described in [88].
For well-posedness it is necessary to impose additional boundary conditions on the artificial boundary to reduce the negative influence of the reflection of any outgoing disturbances back into the computational region [8]. For all numerical experiments in this thesis, a subsonic free-stream configuration is used as the initial guess for Newton-iteration. So we just give the inflow and outflow boundary conditions for the subsonic case. For the other cases, we refer to [8] and references therein.

Let the subscript $\infty$ denotes the free-stream values, and subscript $e$ denotes the values from the interior cells adjacent to the boundary. $u_n$ is the velocity component normal to the boundary, and $c = \sqrt{\gamma p/\rho}$ is the speed of sound. The one dimensional Riemann invariants along the outer normal vector of the boundary are

$$
R_1 = u_n + \frac{2c}{\gamma - 1}, \quad R_2 = u_n - \frac{2c}{\gamma - 1},
(2.29)
$$

which are correspond to the eigenvalues $\lambda_1 = u_n - c$ and $\lambda_2 = u_n + c$ respectively. The farfield boundary condition can be given as

**Subsonic inflow**

$$
\lambda_1 < 0, \quad \lambda_2 > 0, \quad R_1 = R_{1\infty}, \quad R_2 = R_{2e}, \quad S = S_{\infty}, u_t = u_{t\infty},
(2.30)
$$

where $S = p/\rho^\gamma$ is the entropy and $u_t$ is the tangential velocity along the boundary.

**Subsonic outflow**

$$
\lambda_1 > 0, \quad \lambda_2 < 0, \quad R_1 = R_{1\infty}, \quad R_2 = R_{2e}, \quad S = S_{e}, u_t = u_{te}.
(2.31)
$$

Now we have stated main details of our algorithm. In the next section, plenty of numerical results will be given to demonstrate the convergence and robustness of the solver proposed in this chapter.

### 2.6 Numerical Results

In this section, the significant improvement of the quality of numerical results with the WENO reconstruction compared with that in [50] is demonstrated. First, a
convergence test of the subsonic flows is implemented to show that the theoretical convergence order could be obtained. Then we take a convergence test of the transonic flows which shows that the non-physical oscillations are removed completely. Results of several testes with different free-stream configurations and different geometrical configurations are demonstrated at the end of this section to show the robustness of our algorithm.

In the numerical simulations, we always use \( \alpha = 2 \) (see (2.18)), \( \tau_i = \tau = 1 \) (see (2.15)), \( \beta = 2 \) (see (2.12)), and the smoothing steps in the multigrid solver is 2.

The codes are based on the Adaptive Finite Element Package (AFEPack) developed by Li and Liu [47]. It will be observed that the residual in all experiments are reduced to \( 10^{-12} \).

In all numerical experiments in this section, we used following quantities as the initial guess of the Newton iteration: the density \( \rho = 1 \), the velocity \( V = (u, v) = (\cos \theta, \sin \theta) \), where \( \theta \) is the attack angle. The other quantities such as pressure \( p \) and energy \( e \) are determined by \( \rho, V \) and the Mach number.

2.6.1 Numerical Convergence Tests

**Example 2.6.1** The problem is the two-dimensional steady-state, subsonic flow around a disk at Mach number \( M_\infty = 0.38 \) [5].

This test is implemented using linear reconstruction, linear WENO reconstruction and linear reconstruction with Venkatakrishnan limiter respectively. The computations are performed on four successively refined grids, i.e. 16 × 12, 32 × 24, 64 × 48, 128 × 96 points respectively. Fig. 2.7 shows the initial mesh of the whole domain and the detail around the inner circle. The grids extend about 20 diameters away from the circle.

Figs. 2.8-2.11 show the Mach isolines obtained with linear WENO reconstruction. The isolines are plotted for values of the Mach numbers \( M_i = M_0 + i \Delta M \), \( i = 0, 1, \cdots, M_0 = 0 \) and \( \Delta M = 0.038 \). For the exact solutions of this problem, the mach isolines should be symmetric. As we can see from the results, the quality of
Figure 2.7: Mesh used in the simulation of subsonic flow around a disk. The left one is entire mesh, the right one is the detail of mesh around the disk.

numerical results becomes better and better with the refinement of the mesh.

Note that there are no analytical solutions for this simulation. Since the inviscid subsonic flow is isentropic, the entropy production in the flow fields can be used to measure the numerical error introduced by the methods. The numerical convergence is studied for different limiting methods, and results are shown in Fig. 2.12. From the results, it is observed that all three methods obtained convergent results; the convergence order of the algorithm with Venkatakrishnan limiter is lower than that without limiting procedure, while with the WENO reconstruction, higher convergence order than that without limiting procedure is obtained.

Note that in the test in Section 3 which the results are shown in Fig. 2.3, and in this test, the convergence orders of the algorithm with linear WENO reconstruction are higher than that without limiting procedure. This could be explained that with WENO method, the stencil which is used to generate the approximate polynomial in each cell is actually wider than that without limiting strategy. So with the refinement of the mesh, the much more reasonable distribution of solutions is obtained with the WENO method in each cell.

Example 2.6.2 The problem is the two-dimensional steady-state transonic flow around the NACA 0012 airfoil at Mach number $M_\infty = 0.8$ and attack angle $1.25^\circ$. 

48
Figure 2.8: The Mach isolines (bottom) obtained with mesh size $16 \times 12$ (top).
Figure 2.9: The Mach isolines (bottom) obtained with mesh size $32 \times 24$ (top).
Figure 2.10: The Mach isolines (bottom) obtained with mesh size 64 × 48 (top).
Figure 2.11: The Mach isolines (bottom) obtained with mesh size $128 \times 96$ (top).
Figure 2.12: Results of mesh-refinement study for linear reconstruction without limiting procedure, linear WENO reconstruction and linear reconstruction with Venkatakrishnan limiter. The error is given as the $L_2$ norm of the entropy production.

In this simulation, two shocks exist around the airfoil. The strong shock is located at the upper boundary of the airfoil, while the weak one is located at the lower boundary. To study the numerical convergence of the algorithm, the experiment is implemented on three successively refined meshes. There are 2662 cells, 10648 cells and 42592 cells in three meshes respectively. Fig. 2.13 (left) shows the mesh around the NACA 0012 airfoil.

Figs. 2.14-2.16 show the Mach isolines obtained with the linear WENO reconstruction. The numerical convergence can be read easily from the figure. That is, with the mesh becomes denser, the shock also becomes sharper. Especially, with the coarse mesh (2662 cells), there is no shock along the lower boundary of the airfoil. It becomes evident when dense mesh is used.

Apart from keeping the numerical accuracy which is shown in Example 2.6.1, the results shown in Figs. 2.14-2.16 demonstrate that the algorithm with the linear WENO reconstruction also works very well for the cases with shocks in the flow fields. To demonstrate the ability of preventing non-physical oscillations of
the WENO reconstruction methods, two comparisons between the Venkatakrishnan limiter and the WENO reconstruction method are given. From the Figs. 2.17 - 2.19 where the Mach isolines around the airfoil are shown, we can see that although the Venkatakrishnan limiter is used, the numerical oscillations are still observed (top figures), while the oscillations are removed completely by using the WENO reconstruction method (bottom figures). The superior of the WENO reconstruction method can be further demonstrated in Fig. 2.20 where the pressure distributions along the airfoil are given. As we can see from the figure, the undershoot and overshoot phenomena are observed clearly for the results obtained with Venkatakrishnan limiter (left column of the figure), while non-oscillatory distributions are given by the WENO reconstruction method.

Example 2.6.1 and Example 2.6.2 show that the algorithm proposed in this paper is stable and convergent. With linear WENO reconstruction, the algorithm keeps the theoretical convergence order, and at the meantime generates the non-oscillatory shock profiles. The following several examples mainly show the robustness of our algorithm.
Figure 2.14: Free-stream configuration: Mach number 0.8, attack angle 1.25°. The airfoil is NACA 0012, and the mesh contains 2662 cells. Top: Mesh details around the airfoil. Bottom: The Mach isolines.
Figure 2.15: Same as Fig. 2.14, except that the mesh contains 10648 cells.
Figure 2.16: Same as Fig. 2.14, except that the mesh contains 42592 cells.
Figure 2.17: The comparison between the Mach isolines obtained with the linear WENO reconstruction (top) and the reconstruction with Venkatakrishnan limiter (bottom). There are 2662 cells in the domain.
Figure 2.18: Same as Fig. 2.17, except that there are 10648 cells in the domain.
Figure 2.19: Same as Fig. 2.17, except that there are 42592 cells in the domain.
Figure 2.20: Comparison of pressure distribution along the surface of the airfoil between the results obtained with linear WENO reconstruction (left) and reconstruction with Venkatakrishnan limiter (right). The mesh contains 2662 cells (top), 10648 cells (middle), and 42592 cells (bottom).
2.6.2 The Robustness of the Algorithm

To demonstrate the robustness of our algorithm, plenty of numerical experiments are executed in this subsection, with different free-stream and geometrical configurations. The meshes used in the following simulations are demonstrated in Figs. 2.13 and 2.21.

Example 2.6.3 The airfoil is NACA 0012. The free-stream configuration: Mach number 0.3, and attack angle 3.0°. The mesh contains 2662 cells.

Example 2.6.4 The airfoil is RAE 2822. The free-stream configuration: Mach number 0.75, and attack angle 1.0°. The mesh contains 3444 cells.

Example 2.6.5 The airfoil is NACA 0012. The free-stream configuration: Mach number 0.85, and attack angle 1.0°. The mesh contains 2662 cells.

Example 2.6.6 The airfoil is NACA 0012. The free-stream configuration: Mach number 0.99, and attack angle 0.0°. The mesh contains 2662 cells.

Example 2.6.7 Two airfoils in the flow fields: NACA 0012 and RAE 2822. The free-stream configuration: Mach number 0.75, and attack angle 1.0°. The mesh contains 7870 cells.

Example 2.6.8 Two NACA 0012 airfoils in the flow fields. The free-stream configuration: Mach number 0.85, and attack angle 1.0°. The mesh contains 2662 cells.

The mesh contains 4360 cells.

The results of Examples 2.6.3-2.6.8 are shown in Figs. 2.22-2.24 respectively. In these results, the isolines are plotted for values of the Mach number $M_i = M_0 + i\Delta M$, $i = 0, 1, \cdots, M_0 = 0$ and $\Delta M = 0.04$. From these results, three observations can be made as followings

1. The algorithm with the linear WENO reconstruction can handle problems with a large range of free-stream configurations, say, from subsonic case with
Figure 2.21: The meshes for two airfoils in the flow fields. Left: the mesh around a NACA 0012 airfoil and a RAE 2822 airfoil, and the whole mesh contains 7870 cells. Right: the mesh around two NACA 0012 airfoils, and the whole mesh contains 4360 cells.

Mach number 0.3 or even lower to transonic case with Mach number 0.99, and different attack angles. For the cases with multi-airfoil in the flow fields, it also works very well. Furthermore, for all numerical experiments, it is no need to adjust the parameters in (2.18), (2.15), and (2.12);

2. From the results of pressure distribution along the surface of the airfoils, we can see that our algorithm can remove the non-physical oscillations very effectively, which is a significant improvement compared with that given in [50] where the Venkatakrishnan limiter is used.

3. The steady state of the numerical solutions is achieved successfully for every numerical experiment. The residual of the system in every simulation is reduced to the machine accuracy with just a few Newton-iteration steps.

2.6.3 Remarks on the Efficiency of the Algorithm

Examples 2.6.1-2.6.8 show the convergence and the robustness of our algorithm, and the significant improvement of quality of numerical results with the help of the
Figure 2.22: Results of Mach isolines (top), pressure distribution along the surface of the airfoil (middle), and convergence history (bottom) obtained using the algorithm with the linear WENO reconstruction. Left: NACA 0012 airfoil with Mach number 0.3 and attack angle 3.0°. Right: RAE 2822 airfoil with Mach number 0.75 and attack angle 1.0°
Figure 2.23: Same as Fig. 2.22, except for the free-stream configurations: NACA 0012 airfoil with Mach number 0.85 and attack angle 1.0° (left), and Mach number 0.99 and attack angle 0.0° (right).
Figure 2.24: Same as Fig. 2.22, except for the free-stream configurations: one NACA 0012 airfoil and one RAE 2822 airfoil in the flow fields with Mach number 0.75 and attack angle 1.0° (left), and two NACA 0012 airfoils in the flow fields with Mach number 0.99 and attack angle 0.0° (right).
WENO reconstruction. However, since the WENO method is used, the efficiency of the algorithm is degraded. When the Venkatakrishnan limiter is used, just one $2 \times 2$ system needs to be solved for every cell in each Newton iteration, while it is four by using the WENO method. There is a large increment of CPU time in the reconstruction step when the WENO method is used. But it should be pointed out that the change of the overall CPU time of the whole algorithm is not significant, mainly because the linear WENO reconstruction improved the convergence of Newton-iteration method. For instance, in Example 2.6.4 the convergence history of the algorithm is shown in Fig. 2.22 (right column, bottom). From the result we can see that around 70 Newton-iteration steps are needed to solve the problem with residual convergence up to machine accuracy, while for the same example which is shown in [50] where Venkatakrishnan limiter is used, around 250 Newton iterations are needed. In other word, to solve the problem shown in Example 2.6.4, the algorithm with the linear WENO reconstruction saved 70% Newton-iteration steps compared with that when Venkatakrishnan limiter is used.

The other comparison is given by Fig. 2.23 (left column, bottom) where the convergence histroy for Example 2.6.5 is shown. In [50], almost 90 Newton-iterations are needed to make the residual convergent to the machine accuracy, while the number becomes 60 when the WENO reconstruction is used. For this case, around 30% iteration steps are saved when the WENO reconstruction is used. The speed up of the Newton-iteration method is observed for all other simulations in Examples 2.6.2-2.6.8.

To enhance the efficiency of the algorithm with the WENO reconstruction, we also used the idea that the WENO method is implemented only in the ”trouble cells”, say, in the cells which the discontinuities are detected. With this strategy, quite a lot of CPU time is saved. However, the quality of numerical results is degraded, and the convergence to the steady state may not be achieved if there exist strong shocks in the flow fields.

We emphasize again that one of the advantages of our proposed scheme is insen-
sitive to the parameters used in the computations: $\alpha = 2$ (see (2.18)), $\tau_i = \tau = 1$ (see (2.15)), $\mu = 2$ (see (2.12)), and the smoothing steps in the multigrid solver is 2.

## 2.7 Conclusion Remarks

In this chapter, we present a robust linear finite volume solver for inviscid steady Euler equations. The algorithm uses Newton-iteration to linearize the Euler equations, then the multigrid method is used to solve the linearized Jacobian matrix with block LU-SGS iteration as its smoother. Both the linear reconstruction with the Venkatakrishnan limiter and the linear WENO reconstruction are used to generate the approximate polynomial in each cell. The local Jacobian matrix of the numerical fluxes is computed using the numerical differentiation, which can simplify the implementation significantly. With the WENO reconstruction, the quality of numerical results is improved significantly compared with that obtained with Venkatakrishnan limiter. The numerical accuracy is kept in the smooth regions, and the non-physical oscillations are removed effectively at the same time. The numerical simulations also demonstrated the robustness of the algorithm: only one set of the three parameters (i.e., the proportional constant $\alpha$ for the local residual, the relaxation parameter $\tau$ in the Newton-iteration, the weight $\beta$ in the WENO method and the number of smoothing steps in the multigrid solver) is employed for problems with various geometrical and free-stream configurations.

It seems that the algorithm proposed in this paper can not be extended to the higher order WENO reconstruction directly. In fact, the WENO reconstruction affected the differentiability of the numerical schemes, and it may also affect the convergence of the solution to the steady state. From results given in this paper, we can see that the algorithm with linear WENO reconstruction is not sensitive to the influence. But it does give the negative effect to the algorithm with the quadratic WENO reconstruction. In next chapter, a new hierarchical WENO reconstruction method will be introduced to help the simulations achieve the steady state successfully.
Chapter 3

The Quadratic Solver Based on the Residual Distribution Schemes

In the last chapter, a robust and effective linear finite volume solver is proposed to solve 2D inviscid steady Euler equations. However, since linear reconstruction is used, only second-order convergence is obtained. To obtain higher order accuracy, the higher order reconstruction is needed. In this chapter, we extend the linear solver proposed in the last chapter to the higher order solver, and discuss several critical topics about keeping high order numerical accuracy.

On unstructured grids, many high-order methods have been presented on unstructured grids. Barth [2] develops the concept of $k$-exact reconstruction scheme for the median-dual finite volume scheme. Such technique is extended to the cell-centered finite volume scheme by Mitchell and Walters [67]. Based on the idea of selecting the locally smoothest stencil, ENO/WENO schemes [40, 52, 74] are proposed for the hyperbolic system of conservation laws. Recently, Discontinuous Galerkin (DG) schemes [13] and Spectral Volume (SV) methods [76, 87] have been developed; both methods employ the information inside the control volume. All these numerical schemes demonstrate excellent high-order behaviors.

To extend the linear solver proposed in Chapter 2 to the higher order case, there are two classical ways:
• The $k$-exact schemes[2]. The reconstruction is based on the cell averages of each cell. The corresponding least square system minimizes the difference between the original cell average and the cell average obtained from the reconstructed polynomial in each cell. Note that the constant term needs to be adjusted to conserve the cell average finally.

• The approximate polynomial in each cell is reconstructed based on the functional value on the centroid of each cell. The corresponding least square system minimizes the difference between the original function value and the reconstructed function value on the centroid of each cell.

In fact, the above two methods are consistent with each other in the linear reconstruction case because for the linear function, its function value on the centroid of the cell is equal to its cell average over the cell. However, they are different with each other for higher order reconstruction case. More precisely, the $k$-exact reconstruction method results in a high order finite volume scheme, while the second reconstruction method results in a high order Residual Distribution (RD) scheme.

The RD schemes have been extensively discussed in the literatures [1, 12, 14]. Traditional RD schemes start with point values of the solution, then an interpolation to obtain a polynomial approximation of the solution, then an approximation of the residual resulting from an integrated version of the steady state PDE (which, by the divergence theorem, involves only integrals of the flux along the cell boundaries), then a pseudo-time or another method to compute the resulting nonlinear system. The scheme in this chapter differs from traditional RD schemes only in the last step, namely the nonlinear system is solved directly by the Newton type method rather than through residual distribution and pseudo-time marching. We point out that our algorithm has almost the same efficiency as finite volume schemes on the unstructured mesh case, since for both schemes, a least square system ($5 \times 5$ for our scheme and $6 \times 6$ for finite volume schemes) is solved in every cell in each Newton iteration step.

To implement the high order reconstruction, a larger reconstruction patch is
needed compared with linear case. For the quadratic reconstruction case, a reasonable choice is to select the cell which has the common vertex with the current cell as the component of the patch. By using such criterion, even the patch of the boundary cell can supply sufficient cells for the quadratic reconstruction.

For the high order reconstruction, we always need certain limiting procedure to guarantee the monotonicity of the numerical solutions. Since the excellent performance of the linear WENO reconstruction in the last chapter, a nature method for the high order reconstruction is to use the WENO method directly. Unfortunately, this strategy works not well according to our numerical experiments. Because the WENO procedure affect the differentiability of the numerical scheme, the convergence of the steady state of the algorithm may also be prevented. Such phenomena are also observed in [56] and [97] where the WENO methods are used as the limiting strategy. In their results, the residual of the system can not achieve the machine accuracy under certain circumstance.

Recently, a so-called Hierarchical Reconstruction method is proposed by Liu et al.[53, 54] for the central and finite volume schemes. Then such limiting strategy is extended to the Discontinuous Galerkin method by Xu et al.[90] and the Spectral Volume method by Xu et al.[91]. For the traditional limiting methods, all the coefficients of the approximate polynomial are constrained at the same time according to the monotonicity criterion of the numerical solutions. Different from those methods, the hierarchical strategy limits these coefficients level by level from the highest order terms to the lowest order terms. With the hierarchical reconstruction technique, the non-physical oscillations are reduced significantly around the discontinuity region, and desired numerical accuracy is also kept. However, results shown in [53, 54, 90, 91] are mainly about the unsteady problems. For the steady problems, the solver works not well if the hierarchical reconstruction technique proposed in these papers is used directly. The possible reason may be that the operators introduced in the hierarchical reconstruction procedure such as the ENO schemes, the minmod functions affected the differentiability of the limiting procedure. So in
the implementation of the algorithm in this chapter, the hierarchical reconstruction is used with some modification.

In this chapter, we first give the quadratic reconstruction based on the RD schemes. After the initial reconstruction, the information on the centroid of each cell such as the function value, the gradients and the second-order derivatives is known. What the hierarchical reconstruction does is to recompute all the information level by level from the highest order terms to the lowest order terms by using the WENO method. The quadratic reconstruction gives a quadratic polynomial in each cell, for example, first the coefficients of second-order terms are recomputed using constant term and gradients in different reconstruction patch, and several candidates for these coefficients are generated. Then the WENO schemes are used to determine the limited coefficients for the second-order terms. For the linear terms, a similar way is used to generate the limited coefficients with constant term and the updated second-order terms of the polynomial. The numerical results in the final section show that such limiting strategy works very well: the high order numerical accuracy is kept in the smooth region, at the meantime, the spurious oscillations nearby the shock region are removed or reduced significantly.

The other ingredients of the algorithm are same as the linear case. The nonlinear Euler equations are linearized by Newton-iteration methods, and the singular linear system is regularized by adding the factor which depends on the local residual. Finally, the regularized system is solved by using the geometrical multigrid methods, and the block LU-SGS iterative scheme is adopted as the smoother.

The rest of this chapter is organized as follows. In the next two section, the quadratic reconstruction technique and the hierarchical limiting strategy will be introduced. Then the curved boundary which is one of the critical topics about keeping high order numerical accuracy will be discussed in Section 3.3. Finally, Numerical experiments will be carried out in Section 3.4.
3.1 High order reconstruction

The discretized steady Euler equations can be written as

$$\sum_{e_{ij} \in \partial K_i} \int_{e_{ij}} \bar{F}(U_i, U_j) \cdot n_{ij} dS = 0,$$

where $\bar{F}(U_i, U_j)$ denotes the numerical flux, and the left and right states of each edge are calculated by using the approximate polynomials which are reconstructed with functional value on the centroid of each cell.

In order to achieve the high order accuracy, we need to reconstruct a high order polynomial on each cell to describe the variation of the solution. In this section, we use the quadratic reconstruction as an example to describe the procedure of reconstruction.

Expanding the function $U$ at the centroid of control volume $K$ by using Taylor expansion, and truncating after the quadratic term, we get the following approximate polynomial solution:

$$h_q(x, y) = a_0 + a_1(x - x_0) + a_2(y - y_0) + \frac{1}{2}a_3(x - x_0)^2 + a_4(x - x_0)(y - y_0) + \frac{1}{2}a_5(y - y_0)^2.$$

where $a_0 = U(x_0, y_0)$ is the function value at the centroid of cell. Our goal is to find out $a_1 = U_x(x_0, y_0), a_2 = U_y(x_0, y_0), a_3 = U_{xx}(x_0, y_0), a_4 = U_{xy}(x_0, y_0)$ and $a_5 = U_{yy}(x_0, y_0)$.

The method to obtain above unknowns is the least square method. The detail is as the follows. First, we choose the reconstruction patch for each cell. For the linear reconstruction which is described in Chapter 2, the reconstruction patch contains the current cell $K_0$ and its direct neighbours $K_1, K_4,$ and $K_7$. Such patch is too small for the quadratic case. So for the quadratic reconstruction, the enlarged patch which contains the current cell $K_0$ and the cell which has at least one common vertex with $K_0$ is adopted. The demonstration of patch $P_q(K_0) = \{K_i\}, i = 0, 1, \cdots, 12$ is shown in Fig. 3.1. Generally speaking, there are around 13 cells in the reconstruction patch for the cell inside the computational domain. This is sufficient for the quadratic
We obtain the unknowns $a_1, a_2, a_3, a_4$ and $a_5$ by solving the following minimization problem:

$$\min_{a_1, a_2, a_3, a_4, a_5} \sum_{\forall K_l \in \mathcal{P}(K_0), l \neq 0} \left| \left| a_{0,l} - a_0 - a_1 x_{0,l} - a_2 y_{0,l} - \frac{1}{2} a_3 x_{0,l}^2 - a_4 x_{0,l} y_{0,l} - \frac{1}{2} a_5 y_{0,l}^2 \right| \right|_2^2, \quad (3.3)$$

where $a_{0,l}$ means the solution value of at the centroid $(x_l, y_l) \in K_l$, and $x_{0,l} = x_l - x_0$, $y_{0,l} = y_l - y_0$.

Before we use the obtained approximate polynomial $h_q(x, y)$ in the cell $K_0$ to calculate the numerical flux, certain limiting procedure should be implemented for preventing the numerical oscillations. In the linear case, the linear WENO reconstruction works very well. But for higher order schemes, it seems not easy to give a limiting strategy which can remove the spurious oscillations and also keep the high order accuracy. In [16], two different limiter functions are designed for the linear and quadratic terms respectively, and satisfactory numerical results are obtained. In [66], the classical Venkatakrishnan limiter function for the second order schemes is extend to the higher order (third- and fourth-order) cases with some modifications. The non-oscillatory numerical results are observed in [66], but the ability
of limiter functions to keep the numerical accuracy is not clear since there are no results on the study of the convergence order. Recently, Liu et al. present a series of papers [53, 54] to describe a new strategy, called hierarchical reconstruction, to limit high order polynomials. The hierarchical reconstruction limits each term of the polynomial in a unified way, which makes the implementation easier. In the next section, we will introduce a method to limit (3.2) following the idea of the hierarchical reconstruction.

### 3.2 The WENO hierarchical limiting strategy

Before introducing the limiting strategy used in the algorithm of this chapter, we briefly summarize the hierarchical reconstruction proposed in [53, 54, 90] first.

#### 3.2.1 Summarization of the Hierarchical Reconstruction

After implementing the quadratic reconstruction for all cells in the computational domain, there is a quadratic polynomial in each cell. For example, in the cell $K_i$, the approximate polynomial is

$$h_i(x, y) = a_{0,i} + a_{1,i}(x - x_0) + a_{2,i}(y - y_0) + \frac{1}{2}a_{3,i}(x - x_0)^2$$

$$+ a_{4,i}(x - x_0)(y - y_0) + \frac{1}{2}a_{5,i}(y - y_0)^2.$$  \hfill (3.4)

To constrain these polynomials, we first limit the coefficients of the second order terms, $a_{3,i}, a_{4,i}$ and $a_{5,i}$. Take the first partial derivative with respect to $x$ for every $h_i(x, y)$, we obtain

$$h_{i,x} = a_{1,i} + a_{3,i}(x - x_i) + a_{4,i}(y - y_i).$$ \hfill (3.5)

Then we calculate the cell average of $h_{i,x}$ on the cell $K_i$ to obtain $\bar{h}_{i,x}$, for all $K_i$ in the computational domain. Now we can reconstruct the linear polynomial based on these cell averages $\bar{h}_{i,x}$ with certain non-oscillatory method, and get the new approximate linear polynomial

$$\tilde{h}_{i,x} = \tilde{a}_{1,i} + \tilde{a}_{3,i}(x - x_i) + \tilde{a}_{4,i}(y - y_i),$$ \hfill (3.6)
where $\tilde{a}_{3,i}$ and $\tilde{a}_{4,i}$ are the candidates for the final limited coefficients $a_{3,i}$ and $a_{4,i}$ respectively.

Similarly, take the first derivative with respect to $y$ for every $h_i(x, y)$, we obtain

$$h_{i,y} = a_{2,i} + a_{4,i}(x - x_i) + a_{5,i}(y - y_i).$$

After calculating the cell average in each cell for $h_{i,y}$ and implementing the non-oscillatory reconstruction, we obtain the following new approximate linear polynomial in each cell

$$\tilde{h}_{i,y} = \tilde{a}_{2,i} + \tilde{a}_{4,i}(x - x_i) + \tilde{a}_{5,i}(y - y_i),$$

where $\tilde{a}_{4,i}$ and $\tilde{a}_{5,i}$ are the candidates for the final limited coefficients $a_{4,i}$ and $a_{5,i}$ respectively.

Note that there is only one candidate $\tilde{a}_{3,i}$ or $\tilde{a}_{5,i}$ for the final limited $a_{3,i}$ or $a_{5,i}$ respectively, so these two candidates are used as the new values. However, there are two candidates for the final $a_{i,4}$ which are given in (3.6) and (3.7) respectively. Certain limiter function is introduced to determine the final value of $a_{4,i}$.

So far, the second order information such as $a_{3,i}$, $a_{4,i}$ and $a_{5,i}$ has been updated. The next step is to constrain the linear information, say, to update $a_{1,i}$ and $a_{2,i}$ in (3.4). Equation (3.4) can be reformulated as

$$h_i(x, y) = L_i(x, y) + R_i(x, y),$$

where $L_i(x, y) = a_{0,i} + a_{1,i}(x - x_i) + a_{2,i}(y - y_i)$ is the linear part of $h_i(x, y)$, and $R_i(x, y) = \frac{1}{2}a_{3,i}(x - x_i)^2 + a_{4,i}(x - x_i)(y - y_i) + \frac{1}{2}a_{5,i}(y - y_i)^2$ is the remainder. Note that $a_{3,i}$, $a_{4,i}$ and $a_{5,i}$ in $R_i(x, y)$ are the updated version. In each cell $\mathcal{K}_i$, let $\bar{U}_i$ and $\bar{R}_i(x, y)$ denote the cell average of the numerical solution and $R_i(x, y)$ in (3.8) respectively, where $R_i(x, y)$ is calculated from the updated $a_{3,i}$, $a_{4,i}$ and $a_{5,i}$. Then the cell average for the linear part of the numerical solution $\bar{L}_i(x, y)$ in each cell $\mathcal{K}_i$ can be obtained by

$$\bar{L}_i(x, y) = \bar{U}_i(x, y) - \bar{R}_i(x, y).$$

After the cell average $\bar{L}_i(x, y)$ of the linear part of the numerical solution in each cell is obtained, the same non-oscillatory reconstruction which is used to generate
the candidates for quadratic information could be implemented for each cell. We
obtain the new approximate linear part of the numerical solution
\[ \tilde{L}_i = \tilde{a}_{0,i} + \tilde{a}_{1,i}(x - x_i) + \tilde{a}_{2,i}(y - y_i), \]
(3.10)
where \( \tilde{a}_{1,i} \) and \( \tilde{a}_{2,i} \) are the candidates for the final limited coefficients \( a_{1,i} \) and \( a_{2,i} \), respectively. Since there is only one candidate for \( a_{1,i} \) and \( a_{2,i} \), \( \tilde{a}_{1,i} \) and \( \tilde{a}_{2,i} \) are used as the final limited values directly.

In the last step, the constant term \( a_{0,i} \) in (3.4) is adjusted in each cell to conserve the cell average of the numerical solution.

In [54], the non-oscillatory MUSCL or the second-order ENO strategy is adopted to implement the linear reconstruction on each level, and the minmod function is used to determine the final value of the term which has several candidates.

However, the linear reconstruction procedure using minmod or ENO limiter function often fails to give the desired order of accuracy. By implementing the reconstruction procedure with a center biased minmod limiter or ENO limiter function can keep the desired order of accuracy, but the significant overshoots and undershoots may be introduced [89]. To fix this problem, Xu et al.[89] used WENO-type linear reconstruction procedure as the non-oscillatory method in each level. To further improve the quality of the numerical solution, the linear WENO reconstruction using partial neighbouring cells strategy is proposed. With the WENO-type reconstruction procedure, the desired numerical accuracy is obtained in the numerical simulations in [89], and the non-physical oscillations are also reduced significantly.

Note that the hierarchical reconstruction strategy presented in [53, 54, 90, 91] is mainly used for solving unsteady problems. Based on our numerical experience, such strategy works not well for simulating the steady problems if it is used directly. For example, since the operators such as the minmod and the ENO functions are adopted, the differentiability of the numerical schemes is affected. Consequently, the convergence of the steady state of the algorithm is also affected.

In this thesis, we only consider the steady state of the Euler equations. So the hierarchical reconstruction is adopted to prevent the non-physical oscillations, with
some modifications.

3.2.2 Implementation of The Hierarchical Reconstruction for Steady Problems

In this subsection, we use the hierarchical reconstruction for the quadratic case as the example to describe the limiting strategy adopted in our algorithm.

To enhance the differentiability of the numerical schemes by using the hierarchical reconstruction method, the ENO schemes are abandoned, and the WENO schemes are adopted for the non-oscillatory reconstruction in each level. Note that for the quadratic or higher order reconstruction, there are always several candidates generated during the implementation of the hierarchical reconstruction for such terms \((x - x_i)^i(y - y_i)^j, i, j \neq 0\). To avoid using the minmod or similar limiter functions to determine the final values (since the minmod limiter function also affects the differentiability of the numerical schemes), and to make the implementation of the limiting procedure much simpler, we reconstruct the whole information in certain level at the same time. For example, for the quadratic reconstruction, we use the constant term \(a_{0,i}\) and the gradients information \(a_{1,i}\) and \(a_{2,i}\) to constrain the second-order information \(a_{3,i}, a_{4,i}\) and \(a_{5,i}\). The problem becomes minimizing the difference of the second-order coefficients between the original ones in the cell \(K_i\) and the reconstructed ones from the reconstruction patch \(P_q\) of \(K_i\). Then \(a_{3,i}, a_{4,i}\) and \(a_{5,i}\) can be obtained by solving the minimization problem, and we do not need the minmod limiter any more.

As mentioned before, our numerical scheme belongs to the residual distribution schemes. The reconstruction is based on the functional value on the centroid of each cell, then the reconstructed polynomial is used directly to describe the steady state of the Euler equations in the integral form. So during the limiting polynomial, we just keep the functional value on the centroid of each cell unchanged.

Since the WENO method is used for the non-oscillatory reconstruction, the stencils for each cell should be determined first. The stencils used in our algorithm for
the quadratic case are much complicated than that for the linear case which is shown
in Fig. 2.1. It seems a little difficult to give a clear figure to demonstrate it. But
the principle to determine the stencils for quadratic case is same as for the linear
case. For example, for the reconstruction patch which is shown in Fig. 3.1, we know
that it is the patch for the cell $K_0$. Furthermore, such reconstruction patch is also
selected to reconstruct the polynomial for the other cells $K_i, i = 1, 2, \cdots, 12$ in the
patch. With such principle, each cell in the computational domain has around 13
stencils. The linear and quadratic information of the approximate polynomial will
be reconstructed with these stencils.

Now we are ready to present the hierarchical limiting procedure used in our
algorithm. Suppose in the cell $K_i$, there are a quadratic polynomial $h_i$ (3.4), and
stencils $S_j, j = 0, 1, 2 \cdots$. According to the idea of hierarchical reconstruction, we
first recompute the quadratic information $a_{3,i}, a_{4,i}$ and $a_{5,i}$. Since the constant term
and the gradients are known in every cell, we can obtain the quadratic information
by solving the following minimization problem

\[
\min_{\tilde{a}_{3,i,j}, \tilde{a}_{4,i,j}, \tilde{a}_{5,i,j}} \sum_{S_j, l \neq i} || a_{l} - a_{0,i} - a_{1,i}x_{0l} - a_{2,i}y_{0l} - \frac{1}{2} \tilde{a}_{3,i,j}x_{0l}^2 - \tilde{a}_{4,i,j}x_{0l}y_{0l} - \frac{1}{2} \tilde{a}_{5,i,j}y_{0l}^2 ||^2_2,
\]

in every stencil. Here $\tilde{a}_{3,i,j}, \tilde{a}_{4,i,j}$ and $\tilde{a}_{5,i,j}$ are the candidates of coefficients of the
second order terms on the cell $K_i$ obtained with stencil $S_j$.

After looping all stencil $S_j$, we obtain several candidates for second-order information in the cell $K_i$, say, $\tilde{a}_{3,i,j}, \tilde{a}_{4,i,j}$ and $\tilde{a}_{5,i,j}, j = 0, 1, 2, \cdots, m$. The final limited quadratic information is given by using the WENO method. First, the smoothness indicator is introduced [31] for each candidate

\[
S_j = \int_{K_i} \left| (\tilde{a}_{3,i,j}^2 + \tilde{a}_{4,i,j}^2 + \tilde{a}_{5,i,j}^2) d\Omega, \right.
\]

where $| K_i |$ is the area of the cell $K_i$. Then the weight of each candidate is given as

\[
\omega_j = \frac{\tilde{\omega}_j}{\sum_k \tilde{\omega}_k}, \quad \tilde{\omega}_k = \frac{1}{(\epsilon + S_k)^\mu},
\]

where $\epsilon = 10^{-4}$ and $\mu = 1$. Finally, the second order information in the cell $K$ is
\[ \tilde{a}_{k,i} = \sum_{j=1}^{m} \omega_j \tilde{a}_{k,i}, \quad k = 3, 4, 5. \] (3.14)

The linear information in each cell \( K_i \) is also reconstructed in a similar way. For obtaining candidates of coefficients of the linear terms \( \tilde{a}_{1,i,j} \) and \( \tilde{a}_{2,i,j} \), the following minimization problem is solved in every stencil \( S_j, j = 0, 1, \cdots, m \),

\[
\min \sum_{a_{1,i,j}, a_{2,i,j} \forall K_l \in S_j, l \neq i} || a_{0,t} - a_{0,i} - \tilde{a}_{1,i,j} x_0 t - \tilde{a}_{2,i,j} y_0 t - \frac{1}{2} \tilde{a}_{3,i} x_0^2 - \tilde{a}_{4,i} x_0 y_0 t - \frac{1}{2} \tilde{a}_{5,i} y_0^2 ||^2_2.
\] (3.15)

For the linear case, the following smoothness indicator is used

\[
S_j = \int_{K_i} (\tilde{a}_{1,j}^2 + \tilde{a}_{2,j}^2) d\Omega.
\] (3.16)

The coefficients of the linear term in each cell is determined in the same way to the quadratic case

\[
\tilde{a}_{k,i} = \sum_{j=1}^{m} \omega_j \tilde{a}_{k,i}, \quad k = 1, 2,
\] (3.17)

where \( \omega_j \) is similarly given by (3.13) with the smoothness indicator (3.16).

Finally, the approximate polynomial in the cell \( K_i \) after the limiting strategy is

\[
h_q(x, y) = a_0 + \tilde{a}_{1,i}(x - x_0) + \tilde{a}_{2,i}(y - y_0) + \frac{1}{2} \tilde{a}_{3,i}(x - x_0)^2
\]
\[
+ \tilde{a}_{4,i}(x - x_0)(y - y_0) + \frac{1}{2} \tilde{a}_{5,i}(y - y_0)^2.
\] (3.18)

Note that the parameter \( \mu = 1 \) used in the quadratic reconstruction is different from \( \mu = 2 \) used in the linear case. The reason is that when \( \mu \to \infty \), the WENO schemes become the ENO schemes. That is, the smoothest results are selected as the final values. We know that the ENO schemes have the inherent non-differentiability which affects the convergence of the steady state of the numerical schemes. So we use the smaller \( \mu = 1 \) compared with that of linear case to make the high order solver converge smoothly. Based on the numerical experiments shown in the last section, \( \mu = 1 \) works very well.

The convergence of the WENO hierarchical limiting strategy is studied by using the smooth function \( f = \sin(\pi x) \cos(2\pi y) \) on the domain \([0, 1] \times [0, 1]\). The mesh
Table 3.1: $L_2$ error and convergence order of quadratic reconstruction with hierarchical limiting strategy.

<table>
<thead>
<tr>
<th>No. of cells</th>
<th>$L_2$ error</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>4.99e-02</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>9.36e-03</td>
<td>2.4</td>
</tr>
<tr>
<td>1024</td>
<td>1.34e-03</td>
<td>2.8</td>
</tr>
<tr>
<td>4096</td>
<td>1.69e-04</td>
<td>3.0</td>
</tr>
<tr>
<td>16384</td>
<td>2.11e-05</td>
<td>3.0</td>
</tr>
<tr>
<td>65536</td>
<td>2.65e-06</td>
<td>3.0</td>
</tr>
</tbody>
</table>

used is generated by EasyMesh [68]. Table 3.1 gives $L_2$ errors on six successively refined meshes, and the expected convergence order is observed.

### 3.3 Remarks on the Curved Boundary

For the simulations in the last chapter where the linear reconstruction is employed, we use the polygon to approximate the computational domain, see 3.2 (left). It works well for keeping second-order numerical accuracy as we can see in the results from the last chapter. However, when the quadratic or even higher order reconstruction is used in each cell, such crude approximation fails to hold the desired numerical accuracy for the computational domain with the curved boundary. In Fig. 3.2 (right), for the Gauss quadrature formula with two points, the exact points should be point A and B. But with the polygon approximation, the quadrature points C and D are actually used. The difference of the position of the points and their normal directions introduces the second-order error which is not suitable for the quadratic reconstruction case. Thus it becomes important to describe the boundary much more precisely.

In [5], Bassi and Rebay carefully discuss the impact on the accuracy when the discontinuous Galerkin method is used with the curved boundary. From their results,
we can see that even for the linear approximation of the solutions in each cell, a high order approximation for the curved boundary cell is necessary to obtain the desired numerical accuracy. This is very different from the results which are given in the last chapter. In the last chapter, though the linear reconstruction is adopted, and the polygon is used to approximate the circle in Example 2.6.1, we still obtain the correct and convergent results with desired convergence order. This confirms the phenomenon given in [5] that the discontinuous finite element method suffers from a piecewise linear approximation of the geometry of solid boundaries much more than standard finite volume schemes. For obtaining desired accuracy, the standard second-order geometrical transformation is adopted in [5], and the demonstration for the quadratic isoparametric element is shown in Fig. 3.3 (left).

Besides the high order approximation for the curved boundary element, Bassi and Rebay also test the case that the algorithm with the domain approximated by the polygon, but the normal direction is adjusted locally along the boundary according to element obtained with quadratic geometrical transformation. Though the accuracy of the numerical solutions is still not good, the quality of the solutions is improved. In [56], Luo et al. further improve the numerical accuracy by adjusting the normal direction in the quadrature points with local true surface normal based
Figure 3.3: Quadratic isoparametric element (left) and simplified quadratic element (right).

on the analytically defined boundary geometries, and the desired convergence order is observed in their results.

In most cases, there is only one curved boundary in each cell in the computational domain, as shown in Fig. 3.3 (right). Based on this observation, Wang and Liu [86] propose a much simpler second-order geometrical transformation which is uniquely determined by four nodes. Such technique works very well for the spectral volume methods. For details about the transformation, we refer to [86].

In our simulation with the high-order reconstruction, we used the method proposed in [56]. Note that we implemented the numerical quadrature with Simpson integral formula instead of Gauss formula. For the Gauss integral formula, only two quadrature points are needed to hold the third-order numerical accuracy. But these two points are inside the interval of the curved edge. By using Simpson formula, the quadrature points are two vertices and the midpoint of the curved edge, see Fig. 3.4. The vertices are known in advanced, so we only need to find out the midpoint of the curved edge (point C in Fig. 3.4). It make the implementation much simpler. From the numerical results demonstrated in the next section, we can see that desired convergence order is obtained with such approximation of the curved boundary. For the edges which are not on the boundary, we use the two-point Gauss integral formula.
The other details of the algorithm with the quadratic reconstruction are same as that with the linear reconstruction case. The nonlinear system 3.1 is linearized by Newton method, and then the linear system is regularized with the $L_1$ norm of the local residual in each cell. Finally, the multigrid method is adopted to solve the system, and the block LU-SGS method is used as the smoother in every mesh level.

Now we take the implementation of the algorithm with the quadratic reconstruction.

### 3.4 Numerical Results

In the following numerical simulations, we first show the convergence of the algorithm with the quadratic reconstruction when the WENO hierarchical limiting procedure is used, and the comparison of the numerical accuracy with results obtained with linear WENO reconstruction. The ability to remove or reduce the non-physical oscillations of the algorithm with the quadratic WENO hierarchical reconstruction is also demonstrated in the second numerical experiment which has shocks in the domain. For the rest of the numerical experiments, different free-stream and geometrical configurations are tested which show the robustness of the algorithm with the quadratic hierarchical WENO reconstruction.

In the simulations, we always use $\alpha = 2$ (see (2.18)), $\tau_i = \tau = 0.2$ (see (2.15)), $\mu = 1.0$ (see (3.13)), and the smoothing steps in the multigrid solver is 2.

The initial guess for the Newton-iteration method is same as the linear case: the density $\rho = 1.0$, the velocity $V = (u, v) = (\cos \theta, \sin \theta)$, where $\theta$ is the attack angle. The other quantities such as the pressure $p$ and the energy $e$ are determined by $\rho$, ...
$V$ together with the Mach number.

The residual of each variable in every experiment reduced to the machine accuracy if there is no special statement.

### 3.4.1 Numerical Convergence Tests

**Example 3.4.1** The test problem is the two-dimensional steady-state, subsonic flow around a disk at Mach number $M_\infty = 0.38$ [5].

The computations have been performed on four successively refined grids, i.e. $16 \times 12, 32 \times 24, 64 \times 48, 128 \times 96$ points. The initial mesh is shown in Fig. 2.7.

Figs. 3.5-3.8 show the comparisons of the Mach isolines obtained with the linear WENO reconstruction (top) and the quadratic hierarchical WENO reconstruction (bottom) on four successively refined meshes. It is obviously observed that the symmetry of the Mach isolines around the inner circle is improved with the refinement of the meshes for both case, and the quality of the quadratic results is better than that of the linear results. In fact, from Fig. 3.8 we can see that, the Mach isolines for the quadratic case (bottom) are almost perfectly symmetric, while there is a little flaw for the linear case which can be observed from the outflow side of the circle.

The numerical convergence and the superiority of the algorithm with quadratic reconstruction are also demonstrated in Table 3.2 where the $L_2$ entropy errors of the algorithm with the linear WENO reconstruction, the quadratic hierarchical WENO reconstruction, and the quadratic reconstruction without limiting procedure are listed respectively. First, with the refinement of the mesh, the error of each case becomes small, and the error obtained with quadratic reconstruction (whatever the limiting procedure is used or not) is always smaller than that obtained with the linear WENO reconstruction. Second, the convergence order of the algorithm with quadratic hierarchical WENO reconstruction is always bigger than that with the linear WENO reconstruction. For the quadratic case, although the convergence order is degraded when limiting procedure is used, the difference of the convergence order obtained between with and without limiting procedure becomes small when
Figure 3.5: The Mach isolines obtained with linear WENO reconstruction (top) and with quadratic reconstruction using WENO hierarchical limiting procedure (bottom). The mesh size is $16 \times 12$. 
Figure 3.6: Same as Fig. 3.5, except that the mesh size is $32 \times 24$. 
Figure 3.7: Same as Fig. 3.5, except that the mesh size is $64 \times 48$. 
Figure 3.8: Same as Fig. 3.5, except that the mesh size is $128 \times 96$. 
mesh is globally refined. This means that the algorithm with quadratic hierarchical WENO reconstruction successfully kept the desired numerical accuracy.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Err. (L-lim)</th>
<th>Order</th>
<th>Err. (Q-lim)</th>
<th>Order</th>
<th>Err. (Q.)</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x12</td>
<td>2.50e-01</td>
<td></td>
<td>2.30e-01</td>
<td></td>
<td>1.61e-01</td>
<td></td>
</tr>
<tr>
<td>32x24</td>
<td>6.22e-02</td>
<td>2.0</td>
<td>5.05e-02</td>
<td>2.2</td>
<td>2.26e-02</td>
<td>2.8</td>
</tr>
<tr>
<td>64x48</td>
<td>1.03e-02</td>
<td>2.6</td>
<td>5.99e-03</td>
<td>3.1</td>
<td>2.52e-03</td>
<td>3.2</td>
</tr>
<tr>
<td>128x96</td>
<td>1.74e-03</td>
<td>2.6</td>
<td>6.33e-04</td>
<td>3.2</td>
<td>2.56e-04</td>
<td>3.3</td>
</tr>
</tbody>
</table>

Table 3.2: The $L_2$ entropy errors and orders of convergence for the linear reconstruction and quadratic reconstruction computations for the flow around a circle on four successive grids; errors are measured in regions $0.5 \leq r \leq 2.0$. "L-lim" and "Q-lim" stand for linear and quadratic reconstruction with limiting procedure respectively. "Q." stands for quadratic reconstruction without limiting procedure.

It is pointed out that the region used to measure the errors is the cirque with radius from 0.5 to 2.0, which contains the inner boundary. Results in Table 3.2 show that the method we propose to take care of the curved boundary works very well. As we discuss in Section 3.3 for the quadratic reconstruction, if the polygon is used to approximate the curved boundary of computational domain, the second-order error will be introduced from the curved boundary, which will pollute the numerical accuracy in the whole computational domain. Table 3.3 shows the $L_2$ entropy error obtained with quadratic reconstruction when polygon is used as the approximation of the curved boundary. From results we can see that, the convergence order is degraded significantly, which confirms our discussion in Section 3.3. Finally, we give the demonstration of the Mach isolines obtained with quadratic reconstruction when polygon boundary is used, the mesh size is $64 \times 48$, see Fig. 3.9. This figure shows that even the polygon boundary is used, the approximate solutions can still be obtained, which confirms the conclusion proposed in [5] that standard finite volume methods suffer from the linear approximation of the curved boundary less than that of DG method.
Figure 3.9: The Mach isolines obtained with quadratic reconstruction using WENO hierarchical limiting procedure, the mesh size is $64 \times 48$. The curved boundary is approximated with the straight line.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Err. (Q.)</th>
<th>Ord.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16x12</td>
<td>1.44e-01</td>
<td></td>
</tr>
<tr>
<td>32x24</td>
<td>2.10e-02</td>
<td>2.8</td>
</tr>
<tr>
<td>64x48</td>
<td>3.09e-03</td>
<td>2.8</td>
</tr>
<tr>
<td>128x96</td>
<td>7.75e-04</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Table 3.3: Same as the last columns in Table 3.2, except that the curved boundary is approximated by the linear segments.
Example 3.4.2  The test problem is a NACA 0012 airfoil in the flow field with the free-stream condition: Mach number 0.8 and the attack angle 1.25°.

As the same example in the last chapter, the computations have been performed on three successively refined grids (2662 cells, 10648 cells and 42592 cells respectively). The numerical results in Figs. 3.10-3.12 show the Mach contour lines around the whole airfoil (top) and nearby the shock region (bottom) with three different meshes respectively. The results are obtained by using the quadratic hierarchical WENO limiting procedure. As expected, the numerical convergence can be read obviously from these results. With the refinement of the mesh, the Mach isolines become smoother, and the shock profiles become sharper.

Besides the numerical convergence, the results presented in this example also demonstrate the ability of the limiting methods to prevent the non-physical oscillations. It is obvious that there are very big overshoot and undershoot around the shock profile as shown in Fig. 3.13 (top). The results are obtained without limiting procedure. As a comparison, the oscillations are reduced significantly with the help of the hierarchical WENO reconstruction, which is shown in Fig. 3.13 (bottom). The improvement of the quality of the numerical solutions by using the hierarchical WENO reconstruction can also be observed from the comparison between results shown in Figs. 3.14 and 3.11. With the hierarchical WENO reconstruction, the Mach isolines shown in Fig. 3.11 are much more ordered than that shown in Fig. 3.14 where the limiting procedure is not used.

It should be pointed out that the numerical oscillations are not removed thoroughly with the hierarchical WENO reconstruction, as the slight overshoot/undershoot phenomenon are still observed in Figs. 3.10-3.12. This seems not avoidable with the present hierarchical reconstruction strategy; see Liu et al. [53, 54, 89] where overshoot/undershoot are also observed. More efforts have to be made to improve the limiting strategy.

In fact, if we use the quadratic WENO reconstruction method in the algorithm, the quality of the numerical solutions can be further improved. Here the quadratic
Figure 3.10: Mach isolines around the whole airfoil (top) and around the shock profiles (bottom) obtained with the quadratic hierarchical WENO reconstruction. The free-stream condition is Mach number 0.8 and attack angle 1.25°. The mesh contains 2662 cells.
Figure 3.11: Same as Fig. 3.10, except that the mesh contains 10648 cells.
Figure 3.12: Same as Fig. 3.10, except that the mesh contains 42592 cells.
Figure 3.13: Comparison of the pressure distribution along the airfoil surface obtained with two methods: the algorithm with quadratic reconstruction where no limiting procedure is used (top), and with quadratic hierarchical WENO reconstruction (bottom).
Figure 3.14: Same as Fig. 3.11, expect that the limiting strategy is not used.
Figure 3.15: Convergence history of the algorithm with quadratic WENO reconstruction (top) and with the hierarchical WENO reconstruction. The mesh contains 42592 cells.
WENO reconstruction means the direct extension of the linear WENO reconstruction which is presented in the last chapter. That is, the whole quadratic polynomial is reconstructed in every stencil of each cell, then the final approximate polynomial is given by certain convex combination of these candidates. As our numerical experience and the results shown in [97], when the WENO reconstruction is used, the convergence order is superior than that obtained without limiting strategy, and the non-physical oscillations are removed effectively. However, the convergence of the steady state of the algorithm is affected by the quadratic WENO reconstruction. As shown in Fig. 3.15, the results on the top is the convergence history of the algorithm with the quadratic WENO reconstruction, and the residual of the system oscillates around $10^{-3}$. For the algorithm with the quadratic hierarchical WENO reconstruction, the system residual achieves the machine accuracy smoothly.

One possible method to improve the convergence of the algorithm by using the quadratic WENO reconstruction is maybe the usage of the artificial viscous term in the governing equation. When the artificial viscous term is used, the differentiability of the numerical schemes will be improved, and hopefully the convergence of the steady state of the algorithm will also be improved. Some preliminary results confirm it, and more effort to improve the convergence of the high order schemes will be taken in the future work.

### 3.4.2 Robustness

We apply our algorithm to problems with different free-stream and geometrical configurations.

**On different free-stream configurations**

Three free-stream configurations are tested for the airfoil NACA 0012 or RAE 2822:

**Example 3.4.3** Low free-stream Mach number of 0.3 and a big attack angle of $3^\circ$, the airfoil is NACA 0012. Results are shown in Fig. 3.16.
Figure 3.16: The contours of Mach number (top left), pressure (top right), surface pressure profiles of the airfoil (bottom left), and convergence history (bottom right) for NACA 0012 airfoil with free-stream configuration: the Mach number is 0.3 and the attack angle is $3^\circ$.

**Example 3.4.4** Moderate free-stream Mach number of 0.73 and attack angle of $1^\circ$, the airfoil is RAE 2822. Results are shown in Fig. 3.17.

**Example 3.4.5** High free-stream Mach number of 0.99 and attack angle of $0.0^\circ$, the airfoil is NACA0012. Results are shown in Fig. 3.18.

The meshes used in all the numerical tests are generated by EasyMesh, and are demonstrated in Figs. 2.13. The mesh for NACA 0012 airfoil contains 2662 cells, and the mesh for RAE 2822 airfoil contains 3444 cells. From results shown in Figs. 3.16-3.18, we can see that our algorithm works very well with the same set of parameters.
Figure 3.17: Same as Fig. 3.16 except with Mach number 0.73 and attack angle $1.0^\circ$, and the airfoil is RAE 2822.

indicated at the beginning of this section. For all the free-stream configurations, the residual of the system reduces to $10^{-10}$ with around 500 Newton iteration steps. Although there are still slight numerical oscillations which can be observed from the plot of surface pressure coefficients, the WENO hierarchical limiting strategy improves the quality of the numerical solutions nearby the shock region significantly.

**On multi-airfoil in the flow fields**

To test the robustness of the algorithm, we also present the following example which has two airfoils in the flow field.

**Example 3.4.6** One NACA 0012 airfoil and one RAE 2822 airfoil in the flow field with Mach number 0.73 and the attack angle $1.0^\circ$. 

101
Fig. 3.18: Same as Fig. 3.16 except with Mach number 0.99 and attack angle $0.0^\circ$.

Fig. 3.19 shows the numerical solutions of Example 3.4.6. It can be seen from the figure that our algorithm works successfully. It is also observed from the convergence history that the residual for this case can achieve the machine accuracy with almost the same number of iterations as that of the single airfoil case (see Fig. 3.17).

We emphasize again that one of the advantages of our proposed scheme is insensitive to the parameters used in the computations: $\alpha = 2$ (see (2.18)), $\tau_i = \tau = 0.2$ (see (2.15)), $\mu = 0.2$ in the WENO procedure, and the smoothing steps in the multigrid solver is 2. This is in contrast with many existing algorithms where quite a number of parameters have to be fixed case by case.

Of course, by adjusting the parameters used, the efficiency can be further improved. For example, for simulations shown in Figs. 3.16 and 3.18, if the value
τ = τ_i = 0.6 is used, the iteration steps can be reduced significantly compared with that by using τ = τ_i = 0.2, see Fig. 3.20.

3.5 Conclusion Remarks

In this chapter, the linear solver proposed in the last chapter is extended to the quadratic case successfully. A new hierarchical WENO limiting strategy is presented, which can reduce the numerical oscillations significantly and can also keep the high numerical accuracy generated by the quadratic reconstruction. Apart from the high-order accuracy, the numerical examples also demonstrate the robustness of our algorithm: one set of the four parameters (i.e., the proportional constant α for the local residual, the relaxation parameter τ in the Newton-iteration, the weight μ
in the WENO scheme, and the number of smoothing steps in the multigrid solver) is sufficient for various geometrical configurations and free-stream configurations. This is regarded as a strong indicator of robustness. Of course, the efficiency can be optimized by adjusting these parameters case by case.

The hierarchical WENO limiting strategy used in this paper works smoothly for all the numerical experiments. However, the slight undershoot/overshoot phenomenon can still be observed near the shocks. This will be further studied in future by improving the limiting strategy.

It is important to point out that the scheme in this paper is not cheaper than the traditional finite volume schemes, since the least square interpolation used in this paper is not less expensive than the traditional least square reconstruction from cell averages in a finite volume setting. This is different from the residual distribution scheme in Chou and Shu [12] in which a non-smooth tensor product mesh is assumed and then the interpolation is avoided. A numerical integration for the flux along the cell boundary is directly computed based on the point values of the solution along the line of the cell boundary, thus resulting in significant cost saving over finite volume schemes.

In Chapters 2 and 3, the fixed meshes are used in the simulations. Compared
with the whole physical domain, the domain occupied by the airfoils is relatively much smaller. Furthermore, generally we only concentrate on the solutions around the airfoils. For the regions far away from the airfoils, there are almost no variations of the solutions. For the fixed mesh case, if a very dense mesh is used to obtain the high quality numerical accuracy around the airfoil, a large amount of mesh grids are also distributed in those regions far away from the airfoils. Consequently, very much CPU time and large memory are needed to solve this large system. In the next chapter, the adaptive techniques will be employed to optimize the efficiency of the algorithm proposed in Chapters 2 and 3.
Chapter 4

Combination with the Adaptive Techniques

In Chapters 2 and 3, to obtain high resolution results, the mesh was globally refined, see Figs. 2.8-2.11, and 2.14-2.16 in Chapter 2, and corresponding results shown in Chapter 3.

For some problems, they have the following common properties. In most parts of the computational domain, the variation of the numerical is very small. Just in a very small region, the numerical solutions vary dramatically. If the global refinement of the mesh was used to obtain high resolution results, a large amount of degree of freedoms will be wasted. In fact, we just need a large amount of degrees of freedoms in that small region where the solutions vary dramatically.

So far, there are three classical methods to optimize the distribution of the degree of freedoms [77]:

- $h$-method. The $h$-method involves automatic refinement or coarsening of the spatial mesh based on a posteriori error estimates or error indicators. The overall method contains two independent parts, i.e. a solution algorithm and a mesh selection algorithm.

- $p$-method. The $p$-method involves the adaptive enrichment of the polynomial order.
• r-method. The r-method is also known as moving mesh method (MMM). It relocates grid points in a mesh having a fixed number of nodes in such a way that the nodes remain concentrated in regions of rapid variation of the solution.

In the following two sections, we briefly introduce the moving mesh method with some relevant applications. The moving mesh method used in the numerical simulations is based on the harmonic mapping approach, which was first introduced by Dvinsky [18] and further developed by Li et al. [48, 49]. Based on the consideration of the existence and uniqueness of the harmonic mapping, the algorithm proposed by Li et al. [48, 49] can only be used for the simply connected domains. As the domains in Chapters 2 and 3 are multiply connected domains, the h-adaptive method will be considered in Section 4.2 to improve the algorithms proposed in Chapters 2 and 3.

4.1 Moving Finite Element Method for the Simulation of Gravity Fingers in Porous Media

In [33], the moving finite element method was adopted to simulate the finger phenomenon in the porous media. The governing equation for modeling the finger phenomenon is the Non-equilibrium Richards equation (NERE), which can be read as

$$\frac{\partial S}{\partial t} = \nabla \cdot (D(S)\nabla S) + \frac{\partial}{\partial z} K(S) + \tau \nabla \cdot \left( K(S)\nabla \frac{\partial S}{\partial t} \right),$$

(4.1)

where $S$ is the effective saturation ($0 \leq S \leq 1$), $K(S)$ is the hydraulic conductivity, $D(S)$ is the diffusivity function, and $\tau$ is a constant. When $\tau = 0$, NERE becomes the classical Richards equation, which has been proved unconditional stable [19] and so not suitable for modeling finger phenomenon.
4.1.1 Finite Element Discretization

Suppose the physical domain is $\Omega \in \mathbb{R}^2$, and the triangulation of $\Omega$ is $T$ with $T_i$ as its elements and $X_i$ as its nodes. The piecewise linear finite element space is denoted by $V_h$ corresponding to $T$. Then the finite element approximation of unknown $S$ in (4.1) can be written as

$$S_h = \sum_{i=1}^{N_T} S_i N_i(T),$$

where $\{S_i\}_{i=1}^{N_T}$ is the coefficient for variable $S$ in (4.1) with $N_T$ the dimension of $V_h$, and $\{N_i(T)\}_{i=1}^{N_T}$ the finite element basis for $V_h$.

For convenience, $S$ is also used to denote the coefficient. Based on the above assumptions, the discretization formulation for Eq. (4.1) takes the form

$$M_S \frac{\partial S}{\partial t} = -\mathcal{K}_S S + F_S - \tau \tilde{\mathcal{K}}_S \frac{\partial S}{\partial t},$$

(4.2)

where $M_S$, $\mathcal{K}_S$ and $\tilde{\mathcal{K}}_S$ are matrices. whose entries are

$$M_S(i, j) = \int_{\Omega} N_i(T) N_j(T) + \tau K(S) \nabla N_i(T) \nabla N_j(T) dx dz,$$

$$\mathcal{K}_S(i, j) = \int_{\Omega} D(S_h) \nabla N_i(T) \nabla N_j(T) dx dz,$$

$$\tilde{\mathcal{K}}_S(i, j) = \int_{\Omega} K(S_h) \nabla N_i(T) \nabla N_j(T) dx dz,$$

and $F_S$ is a vector whose $i$-th entry is

$$(F_S)_i = \int_{\Omega} \frac{\partial}{\partial z} K(S) N_i(T) dx dz, \quad 1 \leq i \leq N_T.$$ 

For the time discretization, the Euler scheme is used. Let $\Delta t$ be the length of time step, the final full-discretized scheme can be formulated as

$$\left( \frac{1}{\Delta t} (M_S + \tau \tilde{\mathcal{K}}_S(S_h^{(n)}) + \mathcal{K}_S(S_h^{(n)})) \right) S^{(n+1)} = \frac{1}{\Delta t} \left( M_S + \tau \tilde{\mathcal{K}}_S(S_h^{(n)}) \right) S^{(n)} + F_S(S_h^{(n)}).$$

(4.3)

In the above equation, the first and third term on the right hand side of (4.2) are handled implicitly for the consideration of stability.
4.1.2 Moving Mesh Strategy

Our mesh redistribution strategy is based on the harmonic mappings [48, 49]. Suppose the physical domain $\Omega$ and the computational domain $\Omega_c$ are two compact Riemannian manifolds of 2D with metric tensors $G_{ij}$ and $g_{\alpha\beta}$ in some local coordinates $\vec{x}$ and $\vec{\xi}$ respectively. The energy for a map $\vec{\xi} = \vec{\xi}(\vec{x})$ can be defined as

$$E(\vec{\xi}) = \sum_k \int_{\Omega} G_{ij} \frac{\partial \xi^k}{\partial x^i} \frac{\partial \xi^k}{\partial x^j} d\vec{x},$$

where $(G^{ij})^{-1}$ is called the monitor functions. By solving the following Euler-Lagrange equation

$$\frac{\partial}{\partial x^i} \left( G_{ij} \frac{\partial \xi^k}{\partial x^j} \right) = 0,$$

the minimizer of 4.4 can be obtained.

In the implementation, we use the following procedure to move the mesh and to redistribute the numerical solutions. First, we get the initial mesh $T_c$ with $A$ as its nodes in the logical domain by solving the Poisson equation

$$\begin{align*}
\Delta_\xi \xi &= 0, \quad \vec{x} \in \Omega, \\
\xi |_{\partial\Omega} &= \xi_b.
\end{align*}$$

(4.6)

This initial logical mesh is used as the reference to denote the movement of the mesh grid in the physical domain. Now suppose the solution $U_i = \vec{u}(X_i)$ on the current nodes $X_i$ has been given at the time step $t = t_n$, then the new location of nodes $X_i^*$ and the new solution $U^*$ on the new nodes can be obtained by using the following three steps.

1. Obtain the error of mesh points in the logical domain.

    First, we solve the following generalized Poisson equation

$$\frac{\partial}{\partial x^i} \left( G_{ij} \frac{\partial \xi^k}{\partial x^j} \right) = 0$$

(4.7)

together with the same boundary condition to (4.6), and the new logical mesh $T^{new}$ with $A^{new}$ as its nodes can be obtained. Then the quantity

$$\delta A = A - A^{new}$$

(4.8)

can be used as the movement of nodes in the physical domain in the next step;
2. Obtain the movement of physical nodes.

By solving the following system on each element $E$ with $X_{E_i}$ as its nodes in the physical domain (we use triangular mesh in the implementation, so it is a $2 \times 2$ order system)

\[
\begin{bmatrix}
A_{E_1}^{new,1} - A_{E_0}^{new,1} & A_{E_2}^{new,1} - A_{E_0}^{new,1} \\
A_{E_1}^{new,2} - A_{E_0}^{new,2} & A_{E_2}^{new,2} - A_{E_0}^{new,2}
\end{bmatrix}
\begin{bmatrix}
\frac{\partial x^1}{\partial \xi^1} & \frac{\partial x^1}{\partial \xi^2} \\
\frac{\partial x^2}{\partial \xi^1} & \frac{\partial x^2}{\partial \xi^2}
\end{bmatrix}
= 
\begin{bmatrix}
X_{E_1}^1 - X_{E_0}^1 & X_{E_2}^1 - X_{E_0}^1 \\
X_{E_1}^2 - X_{E_0}^2 & X_{E_2}^2 - X_{E_0}^2
\end{bmatrix}
\]

we can get the $\frac{\partial \vec{x}}{\partial \xi}$ in the element $E$. Then the weighted average error of mesh points in the physical domain is given by

\[
\delta X_i = \frac{\sum_{\vec{x} \in E} |E| \left| \frac{\partial \vec{x}}{\partial \xi} \right| E \delta A_i}{\sum_{\vec{x} \in E} |E|}
\]

(4.10)

where $|E|$ is the volume of element $E$. Finally, we get the new mesh on the physical domain with $X^{new}$ as its nodes

\[
X^{new} = X + \tau \delta X
\]

where $\tau$ is a positive parameter and is used to avoid the intersection of new grid points.

3. Update the solution on the new mesh

The method we used to update the solution based on the assumption that the surface of solution on each time step will be kept unchanged. For the detail of the method, we refer to [48].

**Monitor Function**

Generally, people use the standard gradient based monitor function

\[
m = \sqrt{1 + \beta |\nabla S|^2}
\]

(4.11)

where $\beta$ is a positive constant. When $\beta = 0$, it means one identity map between the computational domain and physical domain is defined and there is totally no
adaptivity. With $\beta > 0$, the adaptive phenomenon of solution can be observed, and the higher value of $\beta$, the more adaptivity. Unfortunately, $\beta$ is problem-dependent, how to choose this parameter is mostly depend on your comprehension about the problem.

In the simulation of this paper, we used the BM type monitor function$[23, 24, 96]$ which can be read as

$$m = M(t) + \delta |\nabla S|^\alpha$$

where $\alpha$ and $\delta$ are positive constant, and followed $[6]$ to choose the time dependent function $M(t) = \frac{1}{|\Omega|} \int_{\Omega} |\nabla S|^\beta d\Omega$, where constant $\beta$ is also a positive parameter, and $\Omega$ is the physical domain and $|\Omega|$ is its area. In $[6]$, $\alpha = \beta = 0.5$ and $\delta = 1.0$ are suggested (actually, there is no parameter $\delta$ in that paper). With these choices, the floor on the BM monitor matrix is adjusted automatically in proportion to the measure of the (smoothed) solution gradient$[6]$. In this paper, we used parameters $\alpha = 1$, $\beta = 2$, and for different computational domain, we use different $\delta \in (0, 1]$. For some simulation, we used smaller $\delta$ to enhance the robustness of the algorithm. In certain simulation, the diffusive strategy for monitor function proposed by $[85]$ is adopted to cluster much more grid points around the wetting front.

So far we have stated all details about the algorithm, the flow chart of the algorithm is as following

<table>
<thead>
<tr>
<th>Algorithm 4: Moving FEM for NERE model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Generate initial mesh and set the initial status for $S$;</td>
</tr>
<tr>
<td>2. Solve (4.3);</td>
</tr>
<tr>
<td>3. Move the mesh using the strategy proposed in this section;</td>
</tr>
<tr>
<td>4. Let $t = t + \Delta t$, if $t &lt; T$, goto step 2; else stop.</td>
</tr>
</tbody>
</table>

4.1.3 Numerical Results

We just present one numerical example in this subsection, and refer to $[33]$ for fruitful and detailed numerical results about this model.
Example 4.1.1 Simulating one single finger which created through introducing water via a continuous constant region source in a dry soil.

The initial condition is taken as

\[
S_{\text{init}} = \begin{cases} 
7.0, & 7.5 \leq x \leq 12.5 \text{ } \& \text{ } z \geq 18.0 \\
1.0, & \text{elsewhere}
\end{cases}
\] (4.13)

For the boundary \( y = 20 \) (See Fig. 4.1), the Dirichlet boundary condition is used, and homogeneous Neumann boundary condition is used for the other boundaries. The initial mesh size is around \( \Delta h = 0.5 \). We follow [19] to choose \( K(S) = S^2 \) and \( D(S) = S^2 \) in (4.1).

![Figure 4.1: Computational region for single finger](image)

From results shown in Fig. 4.2, we can see that the grid points are clustered around the edge of the "finger" successfully with moving mesh strategy proposed in Section 4.1.2. With sufficient mesh dense, the hold-back-pile-up phenomenon was obviously observed from the saturation isolines shown in Fig. 4.2 (left).

Note that with the time development, the area of solutions with the large gradients in the computational domain becomes bigger. So a hidden trouble is that when such area is sufficiently big, there may be not enough grid points to describe
Figure 4.2: Numerical solutions of Nonequilibrium Richard equation with initial condition: $K(S) = S^2$, $D(S) = S^2$, $\tau = 0.5$, $S_1 = 0.7$ and $S_0 = 0.06$. 
the variation of the solutions precisely in the domain. By using $h$-adaptive method, that hidden trouble could be avoided effectively.

As we mentioned at the beginning of this chapter, the moving mesh method based on the harmonic mapping used in this section can not be used directly for numerical simulations in Chapter 2 and 3, since the existence and uniqueness of the harmonic mapping can not be guaranteed for the multiply connected domain. So in the next section, the $h$-adaptive method will be introduced to enhance the algorithm proposed in this thesis.

### 4.2 $h$-Adaptive Method

Different from the $r$-adaptive method, which moves the grid points to where they are most needed while keeping the total number of grid points unchanged, the $h$-adaptive method adjusts the mesh density locally based on certain indicator by refining or coarsening the meshes.

Generally speaking, the algorithm for solving certain partial differential equation with adaptive technique has similar framework to Algorithm 4. For $h$-adaptive method, Step 3 in Algorithm 4 becomes locally refining or coarsening the mesh, and then updating solutions from the original mesh to the updated mesh. The idea of $h$-adaptive method is concise. For region with large indicator, the element is refined, while for region with sufficient small indicator, the element is coarsen, say, several small elements is combined together to be a big element. However, it is a great challenge to develop one efficient algorithm to implement such procedure. In [46], Li proposed an algorithm which can locally refine or coarsen mesh efficiently, based on a hierarchical data structure which is defined by himself. In the numerical simulations, the algorithm proposed by Li [46] will be used. In the following section, such algorithm will be briefly summarized.
4.2.1 Mesh Refinement and Coarsening

For keeping the flexibility and the efficiency of the implementation of $h$-adaptive method, Li adopted a hierarchical way to record and manage the mesh, and the refining procedure.

In the implementation, everything in the mesh is considered as a Geometry. For example, for a closed domain $\Omega$ in $\mathbb{R}^2$, there is a given regular simplex triangulation $\mathcal{T}$. For every element $K \in \mathcal{T}$, $K$ itself is a geometry, while its edges are also geometries, and so are its nodes. For these geometries, they all have vertices and edges. The current geometry is defined by describing its vertices and edges. For clearly demonstrating such hierarchical way, the mesh data file for the mesh shown in Fig. 4.3 is given in Table 4.1. From Table 4.1, we can see that everything in the mesh including nodes, edges, and elements has a record. For a high-dimension geometry, it is defined by describing its low-dimension geometries such as its vertices and edges which belong to itself. By using this hierarchical structure, a mesh for a domain can be described clearly, and from these belong-to relations, the vertices and the edges for a geometry could be find out easily.

Figure 4.3: A mesh with four elements.
Table 4.1: The data file of the mesh shown in Fig. 4.3.
Besides the hierarchical definition of the geometry, the implementation of the mesh refinement is also managed by using a hierarchical way. For example, for the element $K \in \Omega$ which is a triangle, when it is refined, there will be four small triangles in the mesh; when these triangles are refined, each triangle will also generate four small triangles, and so on. This procedure can be clearly demonstrated in Fig. 4.4. From the figure we can see that, by refining the element $K$, four smaller elements $K_0$, $K_1$, $K_2$, and $K_3$ are generated. We can call the element $K$ the parent, and the four smaller elements is its children. Similarly, $K_2$ is the parent, and $K_{20}$, $K_{21}$, $K_{22}$, and $K_{23}$ are its children, and so on. The structure was called *Hierarchy Geometry Tree* in [46].

With the hierarchical description of the geometry and the refinement of the mesh, the locally refining and coarsening the mesh can be implemented efficiently. First, we need certain strategy to tell which element needs to be refined or which patch needs to be coarsened. The typical way is to use the fixed threshold rule and the equi-distribution principle [84], and the popular indicator for the finite element method is an posteriori error estimator. For simulating steady Euler equations, we follow [50] to use the gradient of the pressure as the indicator of the mesh adaption. The indicator in each cell $K_i$ is given as

$$I_i = \sum_{e_{i,j} \in \partial K_i} \frac{|p_i - p_j|}{|d_{i,j}|} |e_{i,j}|,$$  \hspace{1cm} (4.14)

where $d_{i,j}$ is the distance between the barycenters of the cell $K_i$ and $K_j$, $|e_{i,j}|$ is the length of the edge $e_{i,j}$.

The adaption procedure now can be implemented toward equi-distributing each element indicator $I_i$

$$\theta \cdot tol \leq I_i \leq \bar{\theta} \cdot tol,$$  \hspace{1cm} (4.15)

where $tol$ is the tolerance and $0 < \theta, \bar{\theta} < 1$. In the simulations, $tol = 5.0 e - 03$ is used according the numerical experience.

For the element $K_i$ with the indicator $I_i > \bar{\theta} \cdot tol$, it is refined in the implementation, while for a patch with all the indicators of its children satisfying $I_i < \theta \cdot tol$, the patch is coarsened.
Figure 4.4: A uniform and successive refinement of the element $\mathcal{K}$. 
the patch is used as an element in the mesh. Fig. 4.5 shows the effect of the mesh refinement and coarsening. Since the indicator of the element $K_3$ satisfies $I_3 > \theta \cdot tol$, it is refined which gives four new elements $K_{30}$, $K_{31}$, $K_{32}$, and $K_{33}$ in the mesh. Since for all $K_{2,i}$, $i = 0, 1, 2, 3$, the indicators $I_{2,i} < \theta \cdot tol$, they are then dropped, and $K_2$ is used as the new element of the mesh.

After looping all Leaf Nodes in the hierarchy geometry tree, the adapted mesh based on the current solutions is obtained. The leaf node stands for the element used in the current mesh. The leaf nodes do not have children in the geometry tree.

Generally, the operation of updating solutions from the current mesh to the adapted mesh is important to keep the numerical accuracy for the time-dependent problems. This thesis is concerned with the simulations of the steady problems, so the solution updating becomes much simpler. In the implementation, we just let the solutions in the new element equal to solutions in its parent if the new element is obtained by refinement. On the other hand, the solution will be chosen as one of its children if the element is obtained by coarsening. Though it is not very accurate, the updated solutions are already a very good initial guess for the implementation of the Newton-iteration method.

Note that there is a midpoint in the element $K_2$ in Fig. 4.5 (right). In the implementation, $K_2$ is considered as a polygon with four vertices, which gives four
edges. This approach is different with the finite element method discussed in [46] where a basis function is designed for this point. Theoretically, arbitrary closed region can be used as a control volume in the finite volume method.

4.2.2 Remarks on the Weight in the Reconstruction

Before demonstrating the numerical results, let us remark on the reconstruction step which has been discussed in Sections 2.2 and 3.1.

Note that the weight $1/d_{K,l}$ in (2.5) is used when solving the minimization problem. Such weight is not used for the quadratic reconstruction in (3.3). In fact, for the uniform mesh used in the simulations in Chapters 2 and 3, whether using the weight in the reconstruction step does not affect the numerical accuracy very much. However, it is pointed out in [61] that serious drawbacks of the unweighted approach are exposed, particularly for highly-stretched meshes in the presence of surface curvatures. Several comparisons of the results obtained with weighted method and unweighted method are still listed in that paper. Their results demonstrate that, if the stretched mesh is used the weighted reconstruction method gives more reliable approximate polynomials.

Although the semiregularization technique in [46] is used in the implementation of mesh adaptation, which can make the variation of the mesh size much smoother, the difference of the cell sizes in the two adjacent cells may be big. For example, it is observed from Fig. 4.5 (right) that the difference of the size of $K_{32}$ and $K_2$ is big. It can become bigger if $K_{32}$ is refined. So based on the consideration of the numerical accuracy and the stability of the algorithm, it becomes necessary to use the weight in the reconstruction step.

4.2.3 Numerical Results

Example 4.2.1 The problem is the two-dimensional steady-state, transonic flow around the NACA 0012 airfoil at Mach number $M_\infty = 0.8$ and attack angle $1.25^\circ$.

The computation is implemented on four successively locally refined meshes. The
initial mesh contains 2,662 cells, which is demonstrated in Fig. 2.13 (left). The three locally refined meshes contain 4,555 cells, 8,746 cells, and 15,031 cells respectively. The parameters used in the implementation are the same to those used in Chapter 2. First, the element patches generated by Algorithm 2 in Chapter 2 are demonstrated in Fig. 4.6. It is demonstrated that with $h$-adaptive technique Algorithm 2 can generate high quality coarse meshes which are used for the multigrid method.

The results of the Mach isolines (top) and details of the mesh around the airfoil (bottom) are presented in Figs. 4.7-4.10. It is obvious that the shock profiles become sharper with the locally refined mesh, which shows the numerical convergence of the algorithm. At the same time, the meshes become denser around the shock profiles. For example, it is observed from Fig. 4.10 (bottom) that there are two shock profiles around the airfoil just from the mesh information. To see the figure clearly, the results in the box which shown in 4.10 (top) are zoomed out in Fig. 4.11. The results confirm that the locally refined strategy proposed in Subsection 4.2.1 works very well.

The sharp shock profiles obtained by using the locally refined technique are almost the same as those obtained with globally refined mesh. However, a large amount of mesh grids are saved. For example, compared the results shown in Fig. 2.15 where globally refined meshes are used with the results shown in Fig. 4.8, the shock profiles are almost the same since the mesh densities around the shock region for both approaches are almost the same. With the globally refined technique, there are 10,648 cells in the domain, while it is 4,555 cells for the locally refined technique. In other words, by using locally refined technique, up to 50% mesh grids are saved to obtain almost the same solution reduction. The percentage of the saved mesh grids becomes bigger when the mesh is further refined.

The convergence histories of these computations with respect to the residual of the system are listed in Fig. 4.12. Residuals of all four computations achieved the machine accuracy successfully within around 60 Newton-iteration steps. Fig. 4.12 also suggests that the rough solution updating technique discussed in the end of
Subsection 4.2.1 works very well for this example.

Finally, results in Fig. 4.13 show that the pressure distributions along the surface of the airfoil. With the linear WENO reconstruction, the non-physical oscillations are removed effectively. The grid points around the shock region increased with the locally refined technique, which makes the shock profiles shaper.

**Example 4.2.2** Moderate free-stream Mach number of 0.75 and attack angle of $2^\circ$. There are two NACA 0012 airfoils in the flow fields.

For the final numerical example of this chapter, it simulates the steady state of two NACA 0012 airfoils in the flow fields, and the algorithm with quadratic hierarchical WENO reconstruction is used. The initial mesh is shown in Fig. 2.21 (right), and there are 4360 cells in the domain. The parameters used in the implementation are same to that used in Chapter 3.

The simulations are implemented on three successively locally refined meshes, with 4360 cells, 7240 cells, and 13339 cells respectively. All three simulations converge to steady state successfully with residual of the system achieving the machine accuracy. The numerical results are shown in Figs. 4.14 - 4.16. The numerical convergence can be read easily from the figures of Mach isolines (top). With the locally refined technique, the region nearby the shock profiles is refined effectively which can be seen from the figures (bottom). This indicates that the $h$-adaptive technique proposed in this chapter also works well for the algorithm with quadratic hierarchical WENO reconstruction which was proposed in Chapter 3.

### 4.3 Conclusion Remarks

In this chapter, the adaptive method is employed to enhance the algorithm proposed in previous chapters. First, the work which is finished under the guidance of Prof. Zegeling of Utrecht university is demonstrated. In that work, the gravity finger phenomenon in the porous media is simulated by using the moving finite element method. With the appropriate monitor function, the mesh grids are clustered around
Figure 4.6: Element Patches near the airfoil body of NACA 0012 generated by Algorithm 2. It shows the element patches on four successive levels when $h$-adaptive technique was adopted.
Figure 4.7: The Mach isolines (top) and mesh details (bottom) around the NACA 0012 airfoil which obtained with linear WENO reconstruction, and $h$-adaptive method is used. There are 2662 cells in the domain.
Figure 4.8: Same as Fig. 4.7, except that there are 4555 cells in the domain.
Figure 4.9: Same as Fig. 4.7, except that there are 8746 cells in the domain.
Figure 4.10: Same as Fig. 4.7, except that there are 15031 cells in the domain.
Figure 4.11: Same as Fig. 4.10, except that just the Mach isolines and the meshes shown in the box in Fig. 4.10 (top) are demonstrated.
Figure 4.12: The convergence history of the algorithm which is implemented on four successively refined mesh with $h$-adaptive method.

the region with large variation of the solution successfully, and the characteristic of the finger phenomenon is depicted very well by the numerical results.

Since the moving mesh method proposed in this chapter can not be used in multi-connected domain, the $h$-adaptive method is adopted. The details of the implementation of the $h$-adaptive method is described in Section 4.2. Numerical results indicate that the algorithm proposed in the previous chapters worked very well combined with the $h$-adaptive method. The advantages of the algorithm are successfully inherited, say, all simulations achieved the steady state smoothly and the non-physical oscillations are removed or reduced significantly. Furthermore, the region nearby the shock profiles are locally refined effectively, which saves a large amount of mesh grids.
Figure 4.13: The pressure distribution along the NACA 0012 airfoil obtained from algorithm which is implemented by using four successively refined mesh with $h$-adaptive method. Top left: 2662 cells; Top right: 4555 cells; Bottom left: 8746 cells; Bottom right: 15031 cells.
Figure 4.14: The Mach isolines (top) and mesh details (bottom) around two NACA 0012 airfoils which obtained with quadratic hierarchical WENO reconstruction, and $h$-adaptive method is used. There are 4360 cells in the domain.
Figure 4.15: Same as Fig. 4.14, except that there are 7240 cells in the domain.
Figure 4.16: Same as Fig. 4.14, except that there are 13339 cells in the domain.
Chapter 5

Concluding Remarks

In this thesis, an effective and robust framework of the numerical algorithm for solving 2D steady Euler equations is developed. The main ingredients of the algorithm include a standard Newton method as the outer iterative scheme and a linear multigrid method as the inner iterative scheme with the block lower-upper symmetric Gauss-Seidel (LU-SGS) iteration as its smoother. With the proposed algorithm, the nonlinear system is solved with residual up to machine accuracy within a few Newton-iteration steps. It is also found that the algorithm is not sensitive to the parameters. More precisely, for each reconstruction method, one set of the parameters can be used for all simulations with different free-stream configurations and geometrical configurations.

In Chapter 2, the details of the algorithm are carefully described. Since the Euler equations are nonlinear, the Newton method is used to linearize this nonlinear system (Section 2.3). The local Jacobian matrix of the numerical fluxes are computed using the numerical differentiation, which can significantly simplify the implementations by comparing with the manually derived approximate derivatives (Subsection 2.3.1). Without the temporal term, the linearized system is actually singular. A new factor which is based on the local residual in each cell is used to regularize the singular system (Subsection 2.3.2). Finally, the regularized linear system is solved by using the geometrical multigrid method (Section 2.4). The boundary conditions are briefly summarized in Section 2.5.
The linear reconstruction is used in Chapter 2. To prevent the non-physical oscillations, two different limiting procedure are introduced. One is the Venkatakrishnan limiter function (Subsection 2.2.2), and the other one is the WENO reconstruction (Subsection 2.2.3). In the numerical experiments, the comparison of two different limiting methods is presented. The comparison suggests that with the WENO reconstruction, much higher quality of the numerical results is obtained. Although the CPU time is increased in the reconstruction step when the WENO method is adopted, it is observed that the outer Newton-iteration steps used are reduced quite significantly. Consequently, compared with the Venkatakrishnan limiter case, the increment of the CPU time for solving the nonlinear system is not significant when the WENO method is used.

In Chapter 3, the quadratic reconstruction is employed for obtaining high order numerical accuracy. Our numerical experiments suggest that the direct extension of the linear WENO reconstruction proposed in Chapter 2 to the quadratic case does not work not well as it may be difficult to obtain convergence of the steady state. To fix the problem, the hierarchical reconstruction method is adopted. With the hierarchical WENO reconstruction, the algorithm worked very well. Residual of the system for each simulation in Chapter 3 achieved machine accuracy successfully. The desired convergence order is obtained for the smooth solution case. Finally, compared with the results obtained without the limiting procedure, the hierarchical WENO reconstruction reduces the non-physical oscillations significantly.

To further improve the ability of the algorithm, the adaptive technique is employed in Chapter 4. First the moving finite element method is summarized, and results of simulations for the NERE model are demonstrated. Since the moving mesh method used in this chapter can not be extended to the multi-connected domain smoothly, $h$-adaptive technique is adopted to optimize the algorithm. Numerical results showed that the algorithm with $h$-adaptive technique works very well. All advantages of the algorithm proposed in Chapters 2 and 3 are inherited, and at the meantime, the mesh grids around the shock profiles are locally refined successfully,
which saves a large amount of mesh grids.

The quadratic hierarchical WENO reconstruction works smoothly for obtaining the steady solutions. However, the quality of the numerical results are affected. Different from the linear WENO reconstruction, the convergence order obtained with hierarchical WENO reconstruction was degraded compared with that obtained without limiting procedure. The slight oscillations around the shock profile are still observed. Although the quality of the numerical solutions can be improved when the hierarchical strategy was dropped, the quadratic WENO reconstruction method will take the difficulties about the convergence of the steady state to the algorithm. Based on some preliminary results, it is found that the convergence of the algorithm can be improved by introducing certain diffusive mechanism. We will work on this direction to improve the high order algorithm proposed in this thesis in the future.

In this thesis, we mainly focus on the 2D problems. For practical applications, it is necessary to extend the 2D solver proposed in this thesis to the 3D case. Since the algorithm is developed with unstructured meshes, such extension should be straightforward. However, the requirement of the memory for storing the Jacobian matrix is a main challenge when the implicit methods is used. In the future, we will work on the combination of our algorithm with the parallel computation technique to solve 3D problems.

Finally, the research on the aerodynamics design is also expected to do with the robust and effective algorithm proposed in this thesis.
Bibliography


Curriculum Vitae

Personal Information

Name: Guanghui Hu   Sex: Male   Nationality: Chinese
Address: Department of Mathematics, Hong Kong Baptist University, Hong Kong
Email: ghhu@math.hkbu.edu.hk

Education

Ph.D., Computational Mathematics, Hong Kong Baptist University 2009 (Expected)
M.A., Computational Mathematics, Sichuan University 2006
B.S., Computational Mathematics, Sichuan University 2003

Exchange/Working Experiences

Visiting Student at Dept. of Mathematics, Charles Univ., Czech Republic
April 2009 - May 2009
Visiting Student at School of Mathematics, Peking Univ., China
Nov., 2008 - Jan., 2009
Visiting Student at Department of Mathematics, Utrecht Univ., Netherlands
Feb., 2008 - April, 2008
Teaching Assistant at Department of Mathematics, HK Baptist Univ.
2006 - present
Teaching Assistant at School of Mathematics, Sichuan Univ.
2004 - 2006
Honours

- The 1st Prize winner of the 3rd Student Paper Competetion for Young Computational Mathematicians, The China Society for Computational Mathematics 2007

- National Postgarduates Mathematical Contest In Modeling, First-Class Award 2004

Publications


