IMMERSED BOUNDARY METHODS FOR COMPRESSIBLE FLOWS

WANG JUNHONG

NATIONAL UNIVERSITY OF SINGAPORE

2012

IMMERSED BOUNDARY METHODS FOR COMPRESSIBLE FLOWS

WANG JUNHONG

(*B. ENG.*, *M. ENG.*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2012

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

WANG JUNHONG
19 November 2012

Acknowledgements

I would like to express my sincere gratitude to my advisor, Professor Shu Chang, for his invaluable guidance, advice, stimulating suggestions, encouragement and patience throughout the course of this thesis.

I also wish to take this opportunity to thank all the research students and research staff in Prof. Shu's group for their valuable suggestions and kind encouragement. Especially, I thank Dr. Zheng Hongwei and Dr. Liu Ningyu, who gave me valuable suggestions in the early stage of the thesis; Dr. Wu Jie, Ms. Shao Jiangyan, Ms. Ren Weiwei and Mr. Wang Yan, who gave me their selfless, friendly and valuable suggestion and advice on the coding and troubleshooting.

I would like to thank the National University of Singapore (NUS), my employer as well, offering me this opportunity to pursue the Ph.D degree on part-time basis and waiver of the research tuition fees. Moreover, I would like to thank Mr. Tan Chee Chiang, my superior and the Senior Associate Directory of the Computer Centre and Mr. Tommy Ho, the Director of the Computer Centre, for their continuous support and understanding.

Last but not least, I would like to give my special thanks to my wife, daughters and parents. Without their endless and considerate love, support and encouragement, I would not be able to complete this thesis.

Contents

Summary

The major challenge to implement the immersed boundary method (IBM) for compressible inviscid flows is that the velocity on the boundary is unknown and the needs to enforce boundary conditions for other flow variables associated with compressible flows. Moreover, the presence of shock waves adds to the challenges in implementation. In this thesis, a novel flux correction-based immersed boundary method (FC-IBM), a local domain-free discretization (LDFD) method and a local DFD-based immersed boundary method (LDFD-IBM) were proposed and implemented to simulate compressible inviscid flows in a Cartesian grid-based adaptive Euler solver. For the first time, we are able to implement the widely adopted IBM concept for incompressible flows to solve compressible inviscid flows.

The FC-IBM was proposed based on the notion that a wall boundary shall satisfy the conditions of no-penetration, zero mass flux and zero energy flux. The method is implemented through velocity correction to enforce the no-penetration condition and flux correction to enforce the zero flux condition. The advantage of the method is that it avoids the tedious process to compute the boundary curvature and to identify whether the cells are in fluid domain or solid domain. This makes the method unique and simple in implementation.

The LDFD method was proposed with the concept that the wall boundary condition can be enforced by correcting the flow information on the solid cells (DFD cells) next to the boundary in X direction and Y direction directly for 2D flows. Because the DFD cells near the boundary can be identified easily in

X and Y directions, the method is simple to implement. The LDFD-IBM was proposed to make the implementation simpler by avoiding the need to identify the solid DFD cells and fluid DFD cells.

The proposed methods, FC-IBM, LDFD and LDFD-IBM, have been integrated with a 2D adaptive Euler solver developed based on the finite-volume discretization on Cartesian grids. The validation and test results for the adaptive solver demonstrated that it is accurate, efficient and robust in simulating compressible flows with weak or strong shock waves. The ghost-cell method was implemented and benchmarked against the adaptive solver. The three proposed methods have been validated comprehensively by simulating subsonic, transonic, supersonic and hypersonic 2D flows with various irregular boundaries. The results obtained from the validation and numerical tests demonstrated that the new methods to be accurate and efficient. However, numerical viscosity caused by the implementation of FC-IBM and LDFD-IBM is noted and needs to be further studied.

The FC-IBM and LDFD method were integrated into the 3D adaptive Euler solver and have been validated and benchmarked primarily with supersonic flows over bluff body wall boundaries and 3D space vehicle consisting of curved body and wings. The validation results and the benchmark performance demonstrated to certain extent the viability of the solver and the proposed methods for compressible inviscid flows. The present Euler solver was also extended to solve steady-state and unsteady-state laminar viscous flows with minor effort.

List of Tables

List of Figures

Nomenclature

Abbreviations

AMR	Adaptive mesh-refinement
AoA	The angle of attack
CFD	Computational fluid dynamics
CFL	Courant-Friedrichs-Lewy number
DFD	Domain-free discretization
FC-IBM	Flux correction-based immersed boundary method
FVM	Finite volume method
GCM	Ghost-cell method
HLLC	The Harten, Lax and van Leer approximate Riemann solver with the contact wave restored scheme
IBM	Immersed boundary method
LDFD	Local DFD
LDFD-IBM	Local DFD and immersed boundary method
MUSCL	Monotone Upstream-centered Schemes for Conservation Laws

Latin Letters

$\tilde{a}$	The sound speed, the Roe average sound speed
a^L, a^R	The left and right sound speed
A_c	Area of the mesh cell.

c	The sound speed
C_d	The drag coefficient
C_l	The lift coefficient
Cp	The heat capacity
D	Dimension or the diameter of a cylinder
D_{ij} , $D_{i,j}$	Factor of delta function
dS^w	The average arc length of wall segment
dx, dy	The mesh size in X and Y coordinate
dh	The mesh size
dt	Time step
E	Internal energy
F, G, H	Flux vector in conservative form Euler equation
F_v , G_v	The viscous stresses
f_i	Flux term
f^{*L}, f^{*R}	Flux term contributed by left-cell and right-cell
f_d	Drag force
f_l	Lift force
f_v	The viscous flux
i, j, k	The integer index
K	The conductivity
l	The length of 2D cell interface
L	The reference lenght
L_2	The L2 norm

M, Ma	Mach number
M_∞	Free stream Mach number
N	The total number of mesh cells
n	Unit normal vector
n_x, n_y, n_z	Normal vector in X, Y and Z direction
P	The pressure
Pr	The Prandtl number
q	The energy flux
$\tilde{q}$	The average velocity
$q_x,\ q_y$	The heat flux vector
R, R_w	Curvature of wall boundary,
r	The normalized distance in delta function
r, r_i	The ratio of successive gradients on the mesh
Re	The Reynolds number
S_L, S_R	The intermediate signal speeds of HLLC scheme
s_m	The signal speed of the contact wave
St	Strouhal number
t	Time
U	Quantity vector in conservative form Euler equation
$U*^L, U*^R$	The intermediate state for left and right
u, v, w	Velocity components in X, Y, and Z direction
u_B, v_B	The Cartesian velocity at a wall point
V_{NB}, V_{TB}	The normal and tangential velocity at a wall point

V_n	The normal velocity on wall boundary
x, y, z	The Cartesian coordinates

Greek Letters

β	The Cartesian coordinates
ε	The residual threshold
ρ	The density
γ	The adiabatic index for air
λ	The Stokes hypothesis
τ	The stress tensor
μ, υ	The dynamic and kinetic viscosity
ω	The vorticity
Δt	The time step
$\sum$	Summation
Ω	The specific domain of all shadow cells
θ	Angle

Miscellaneous Symbols

∂	Partial derivative
∇	Nabla operator
$(\ )^L$, $(\ )^R$	Superscription denotes the intermediate state at left and right
$(\ )_L$, $(\ )_R$	Subscription denotes the variable at left and right
$(\ )_i$	Subscription denotes the variable at cell (i)

$( )^{n}$	Superscription denotes the variable at time step (n)
$( )_{n}$	Subscription denotes the variable at normal direction
$( )^{w}$	Subscription denotes the variable at wall boundary
$\delta(\square)$	The change of variable $(\square)$

Chapter 1 Introduction

Computational fluid dynamics (CFD) is a numerical analysis tool commonly employed in many industrial areas not only for research & development purposes but also for advanced design and optimization. The wide availability of commercial CFD software tools and highly regarded open-sourced CFD solvers are the evidence for its applications. Body-fitted grids are generally used in almost all such software largely because they are conformed to the boundary surfaces and thus the boundary conditions can be implemented directly. Depending on the type of mesh cell for grids, the structure and the connections among mesh cells for grids, body-fitted grids can be generated either as structured or unstructured grids.

Structured grid-based body-fitted solvers that dominated CFD modeling in the early years are less popular due to their application limitations for irregular and complex domains. The unique advantage of structured grids is that they can be indexed and retrieved directly via (i, j, k), which makes the coding for the solver simple. However, it is difficult, or sometimes not possible, to generate structured grids in irregular and complex domains. To overcome this difficulty, unstructured grid-based body-fitted solvers were introduced and became the most popular solver technique in CFD modeling for most tools/solvers available in the market. The unstructured grid approach demonstrates good flexibility in CFD modeling with irregular boundary geometries in complex domains through using various types of mesh cells and ability to align directly the grid nodes or mesh points on the complex

boundaries. However, the development of CFD solvers based on the unstructured grids is relatively difficult and demanding because it is difficult to directly solve the flow governing equations which are in partial differential equation (PDE) form on such grids. Coordinate transformation of unstructured grids is required for the discretization of the PDE-based governing equations. The data structures of unstructured grids are complex and less computationally efficient than their structured-grid counterparts. Hence, additional computing resources for RAM, storage space and computing time are needed for unstructured grid-based solvers. Despite the flexibility and advantages in modeling irregular and complex domains for CFD analyses, it is difficult to use unstructured body-fitted grids in the CFD analyses for problems involving moving boundaries. Such problems are not limited to the translation, rotation and mixed motion of the boundaries, but also include small deformation or minor change of local boundary and the variation of geometry topology for boundaries such as splitting and merging of solid bodies. As the grid nodes have to be aligned with the boundaries, dynamic re-meshing has to be carried out whenever there is a boundary change. This further increases the complexity and the computational demand in the development and coding for unstructured grid-based solvers for moving boundary CFD analysis.

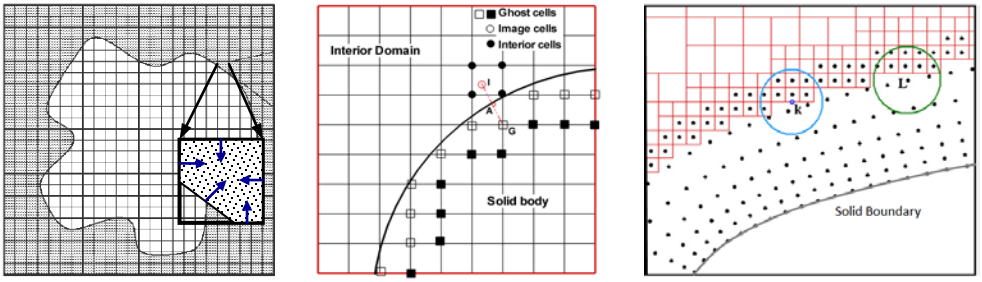
Taking the three-dimensional CFD analysis for high speed compressible flow over an aircraft as an example, tetrahedral unstructured grids can be generated in the domain around the complex aircraft geometries. However enormous efforts have to be spent to re-generate the grid in the entire domain when there is any small geometry change locally in either airfoil profile of the wing, the

slat and flap in expanded position, or the auxiliary fuel tanks' adding or removal.

Cartesian grid-based solver is an alternate for the example described above, besides the body-fitted structured grids and unstructured grids. Cartesian grids have inherently good quality and are easy for grid generation, lower computational storage requirements and significantly less computation per cell as compared to the conventional structured and unstructured grids, as presented by Aftosmis [1]. Cartesian grids use rectangle domain for 2D problems and box domain for 3D problems, and are easy to be adapted and to be extended to higher-order spatial schemes. The convergence performance of the solver is obviously better as there are no skewed or distorted mesh cells. The adaptive mesh-refinement (AMR) technique can be simply implemented on Cartesian grids to give better resolution in resolving the rapid solution changes. However, the major challenge in using the Cartesian grid is to take into account the influence of the arbitrary boundaries to the flow solution. Because Cartesian grids are not aligned with the solid body and its curved boundary surfaces, the grids will intersect with the boundaries in the vicinity of the boundary surfaces. Therefore to ensure the successful use of Cartesian grids, accurate representation of arbitrary boundaries and quick accounting the influence to the solution is essential and inevitable. In the following section, non-body-fitted grid methods in solving compressible flows based on Cartesian grids will be reviewed.

1.1 Non-body-fitted grid methods for compressible flows

Cartesian grids are not fitted to the boundary or solid body. The adoption of Cartesian grid methods to solve compressible flows was initially implemented with cut-cell scheme for compressible Euler equations. The other two common methods proposed to model the arbitrary immersed boundary in compressible flow studies are ghost-cell method and grid-less (or mesh-less as named by some researchers) method. Figure 1.1 briefly illustrates the treatment of the boundary by the three methods for compressible flows.



(a) Cut-cell Method [4] (b) Ghost-cell Method [9] (c) Grid-less Method [13]

Figure 1.1 IBM Approaches for Compressible Flows

1.1.1 Cut-cell method

The concept of cut-cell method on Cartesian grid is to reconstruct the cells that intersect with the solid boundary, as illustrated in Figure 1.1 (a). The cut-cell approach will produce very small cells near the boundary, and this causes stability problem and small time step restriction for the solver [2][3][4]. In addition to the stability problem, special care is required for the calculation of the exact flux for those cut-cells.

A Cartesian cell can be cut into various shapes and sizes by the boundary. By

defining the desired shape for cut-cells and the suitable curvature slope of the neighbor cut-cells, Zeeuw and Powell [4] used geometry-based mesh refinement to construct the cut-cells. The flux calculation on the cut-cells is done through a linear reconstruction approach near the boundary. Coirier and Powell [2] proposed a cell merging method to merge the small cut-cells into larger neighbor parent cells to avoid the stability problems and small time step caused by cut-cell approach. However, this will change the original Cartesian cells not only in the geometries but also on the flux calculation and local accuracy. The small cut-cells to merge have to be identified and the merged cells have to be treated specially. Pember et al. [3] proposed a method that discretizes the fluid-body interface based on a volume-of-fluid approach. A correction is computed for each irregular cut-cell based on the fraction of the area/volume of a cell that is inside the fluid and then it is applied to maintain the conservation items by a variation of the algebraic redistribution algorithm. Nemec and Aftosmis [72], [73] used cut-cell boundaries and refinement interfaces to evaluate the solution accuracy by using linear solution reconstruction functions and trilinear/triquadratic interpolation functions for the solution reconstruction, similar approach to the standard shape functions used in the finite element method. The 3D embedded-boundary problems for the ONERA M6 wing, a missile and a re-entry capsule were studied for the performance analysis.

Numerical experiments obtained using different approaches show that the cut-cell method on Cartesian grid works for various speed of compressible flows. However, the implementation of cut-cell method is rather complex and

inconvenience using either the linear reconstruction approaches, the merging of small cut-cells approach, or the volume-of-fluid approach. A common issue is that the flux calculation for the cut-cells cannot be calculated via the similar simple steps for the Cartesian cells. This degrades the advantages in simulating the flow on Cartesian grids.

1.1.2 Ghost-cell method

To avoid the stability problem and small time step introduced by the cut-cell method, Forrer and Jeltsch [5] derived a new wall treatment method from the concept of one-dimensional wall boundary. The method treats the wall boundary as a symmetrical boundary and hence a mirror point of a fluid point is created symmetrically to the boundary. Because this mirror point is an artificial point, the method is commonly named as ghost-cell method (GCM), as illustrated in Figure 1.1 (b). Since the flow information is symmetrical against the boundary, the scalar quantities such as pressure, density, tangential velocity and energy on the mirror point are set to be the same as they are on the fluid point; the normal velocity vector on the mirror point is set as the same value as it is on the fluid point but in the opposite direction. This treatment satisfies the physical boundary condition, but does not satisfy the curvature condition on the boundary. Because the cells cut through the boundary are still treated as original whole cells, the method becomes simple in implementation.

An improved ghost-cell method, the curvature corrected symmetry technique (CCST), as named by Dadone and Grossmann [6], [7] was proposed to

consider the influence of the boundary curvature on entropy and total enthalpy in the normal direction of the wall boundary. Besides the no-penetration physical condition, the pressure and density at the mirror point are corrected according to the integration of the normal momentum equation and constant entropy relationship, as shown by equation (1.1). The transonic flows computed by the CCST approach show remarkable agreement with the numerical results obtained on body-fitted grids. The CCST approach was further improved by introducing the adaptive mesh-refinement to cluster finer meshes near the boundaries. This has greatly improved the computing efficiency and demonstrates good potential to solve complex flows with arbitrary boundaries such as transonic airfoil [8]. Jiang et al. [9] and Liu et al. [10] adopted the CCST ghost-cell method and successfully studied the shockwave interaction of the supersonic flow over a cylinder and the transonic flow over an airfoil, respectively.

$$\begin{aligned} &\text{Pressure condition:} \quad \frac{\partial P}{\partial n} = -\rho \frac{u_s^2}{R}. \\ &\text{Density condition:} \quad \frac{P}{\rho^{\gamma}} = Const. \end{aligned} \tag{1.1}$$

In general, the ghost-cell method is straightforward and simple in implementation. The method can be more efficient and useful when coupled with the AMR. However, the successful implementation and the accuracy of the method depend on the accurate creation of the ghost-cells, accurate interpolation in determining the values on the ghost-cells, and accurate calculation of the curvature for the boundary. In addition, in the derivation of the pressure condition formula, the curvature or radius (R) is assumed to be positive or the boundary is in convex shape. For concave shape boundary, the

radius (R) in equation (1.1) should be negative. This is because the normal pressure gradient (∂p/∂n) on the wall boundary which follows the normal direction of the wall boundary is now in the opposite direction of the radial direction of the curved boundary, as illustrated in Figure 1.2. Such difference is not mentioned in implementation of those ghost-cell methods.

(a) Convex wall boundary (b) Concave wall boundary

Figure 1.2 Pressure gradient on convex and concave wall boundary

1.1.3 Grid-less method

Grid-less method, or mesh-less method as illustrated in Figure 1.1 (c), is another approach introduced to solve compressible flows on Cartesian grids. The concept is to model the complex boundaries using grid-less points and the rest of domain using simple Cartesian grids. The governing equations are solved by a grid-less solver and a Cartesian grid-based solver in the grid-less zone and the Cartesian grid zone, respectively. So it is a hybrid method consisting of a Cartesian grid-based solver and a grid-less solver. In the interface of grid-less zone and Cartesian grid zone, some Cartesian grids are included as grid-less points, hence the flow information can be exchanged between two zones. Luo et al. [11] proposed and implemented grid-less

method by forming the local cloud of grid-less points from the boundary nodes, cut-off cells and the nearby Cartesian cells on a 3×3×3 stencil for 3D flows. Liang and Yang [12] adopted similar grid-less method and the local cloud of grid-less points were chosen from the nearest eight cells for 2D flows. Zero flow gradients and zero normal velocity conditions on the boundary are used for the interpolation. Jahangirian and Hashemi [13] presented a different implementation of grid-less method. In their work, adaptive Cartesian cells are generated in the domain but keep away from the solid boundary with a pre-defined distance, and within this distance a cluster of grid-less points are generated to surround the solid boundary. So the interface of the grid-less zone and Cartesian grid zone is in some distance away from the boundary, as shown in Figure 1.1 (c). The data exchange between the two zones is thus performed in the interface zone away from the boundary. The benefit of the grid-less method is that the geometry of boundaries can be modeled directly using the grid-less points. However, being a hybrid solver the grid-less method requires extra development efforts and integration of a Cartesian grid solver and a grid-less solver. As a result, the development of the solver becomes complex and the coding demands increase obviously.

In summary, all the non-body-fitted grid methods, cut-cell, ghost-cell, or grid-less, implemented on Cartesian grids can be used to simulate compressible flows with arbitrary wall boundary inside the Cartesian domain. All three methods require the interior grids in the fluid domain and the exterior grids in the solid body to be known. It is noted that the cut-cell method is limited by the stability problem and small time step besides the special needs for flux

calculation on the cut-cells; the ghost-cell method requires the calculation of the curvature of the boundary and the mirror point of the exterior grids against the boundary; the grid-less method needs to integrate a grid-less scheme-based solver with a Cartesian grid-based solver, which requires more development work and demands additional computing resources and more computational effort.

1.2 Local Domain-Free Discretization (DFD) method

An innovative domain-free discretization (DFD) method was proposed by Shu and Fan [14] to solve incompressible Navier-Stokes equations in the Cartesian coordinate system. The method was enhanced by introducing the local DFD method, which is more general and simple in implementation.

The unique feature of the DFD method is that it can be applied to solve irregular domain problems without the need of coordinate transformation technique, which is generally required for body-fitted solvers in irregular domains. The numerical experiments carried out by Shu and co-workers demonstrated high efficiency and accuracy of the DFD method in solving PDEs such as Navier-Stokes equations in arbitrary domain. Their numerical studies also demonstrated that a large numerical error would be introduced into the computation when all the points in horizontal or vertical line in Cartesian coordinate system were used for the discretization process; and accurate numerical results can be obtained by using a few local points.

To eliminate the difficulties and make the DFD method be more general, Shu

and Wu [15] proposed and implemented the local DFD method on cylindrical coordinate system for incompressible flows. The concept of the local DFD method is that the functional values such as pressure, density and velocity at an exterior point can be approximated by spline function using the boundary point and local interior points near the boundary. In Shu and Wu's implementation, two interior points and one node on the boundary were used to approximate the function values on the external point via a quadratic polynomial. The local DFD method has been successfully tested to solve the incompressible natural convection problem in concentric annulus between an inner circular cylinder and an outer square cylinder. Shu and Wu [16] extended the local DFD method with the use of Cartesian mesh for incompressible flows. The functional value at the exterior node can be computed by extrapolation in X direction and Y direction separately using two local interior points and the boundary point. To ensure unique value at the exterior point, the functional value computed in X direction was used for X derivatives calculation only. The same way was applied in Y direction. The adaptive stencil refinement was adopted to enhance the efficiency and accuracy of the results. Wu and Shu [17] further extended the local DFD method to solve an incompressible flow with moving boundary, represented by an oscillating circular cylinder in the domain. The unsteady state results obtained through the method are in good agreement with other numerical results in the literature.

Zhou et al. [18] successfully applied the local DFD method to solve compressible flows using a finite element Galerkin method-based solver. In

their work, the DFD extrapolation is carried out in the normal direction to ensure unique functional values to be obtained at the exterior points. A linear extrapolation is proposed, which involves the wall point inside the fluid domain and one adjacent point along the normal direction. The results obtained for the inviscid and viscous flows around stationary and oscillating airfoils demonstrate that the local DFD method is accurate in solving compressible flows as well. However, the tedious part in the implementation is to construct the fictitious point in the normal direction and ensure that the fictitious point is inside an element fully in the interior fluid zone, despite the adoption of less efficient unstructured grid-based solver. Zhou and Shu [19] extended the similar local DFD method to a 3D finite element solver. The three-dimensional transonic flow over the ONERA M6 wing is solved for the validation of the method. The implementation becomes more complex in 3D solver as the construction of fictitious points in the normal direction and the relationship identification for tetrahedral elements are not straightforward and simple.

1.3 Immersed Boundary Method (IBM) for incompressible flows

The immersed boundary method, or IBM in short, has been adopted, implemented and enhanced in solving various incompressible flows on simple Cartesian grid-based solvers enormously in the past decades since it was first introduced by Peskin in 1970's to solve the incompressible blood flow inside a pumping heat [20], [21]. Since the influence of wall boundary to incompressible flows is either through body force or velocity field, IBM will

work for incompressible flow solver if either one can be modeled.

The concept of IBM is that the deformation or the displacement of the boundary will generate a force that tends to restore the boundary to its original shape or position; this restoring force on the boundary are then re-distributed into the surrounding nodes as body force; after that the flow field with this added body force is solved in the entire domain including both inside and outside of the immersed solid body. This is known as the virtual boundary force IBM. Shu and co-workers [22] proposed a new approach, known as the velocity correction-based IBM, in implementing the immersed boundary by correcting the velocity near the boundary directly to enforce the no-slip physical boundary condition.

1.3.1 Body force correction based IBM

In the virtual boundary force IBM, the wall boundary condition and the governing equation are satisfied through iterative process with the aim to seek zero restoring force, and in theory to enforce the no-slip boundary condition. Based on the IBM proposed by Peskin, various improvements were introduced to enhance and refine the method. Many research works and numerical experiments have been done in solving various incompressible flows involving complex and arbitrary solid boundaries. Goldstein et al. [23] introduced a virtual boundary method which facilitates the simulations to handle complex geometry in space and time as well. Lai and Peskin [24] enhanced the IBM by a proposed second-order immersed boundary method with adoption of a well-chosen Dirac delta function, which was widely adopted and used in the implementation of IBM in the last decade. The

second-order method has less numerical viscosity and is a better choice for the simulation of high Reynolds number flows. Linnick and Fasel [25] proposed a high-order modified IBM for 2D unsteady incompressible Navier-Stokes equations in stream function-vorticity formulation. With explicit fourth-order in temporal domain and fourth-order in spatial domain, the steady and unsteady flows past a circular cylinder and Tollmien-Schlichting waves in a boundary layer were validated extensively.

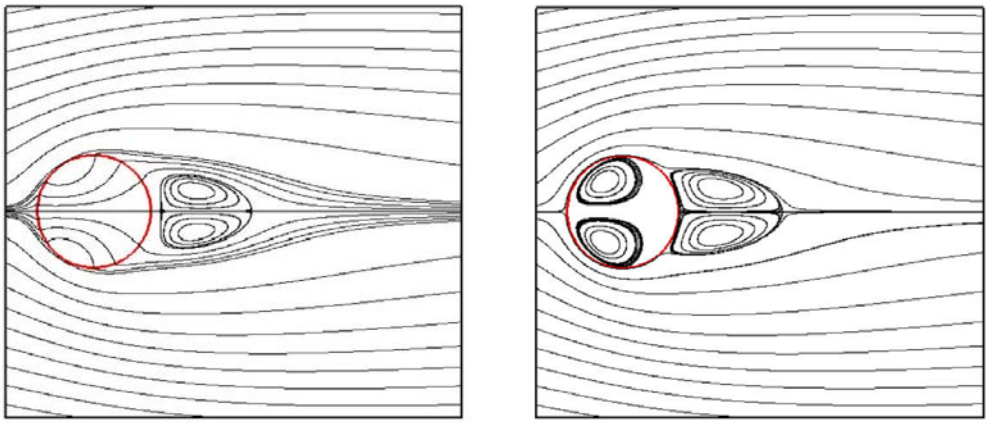
In the past few years, IBM has been implemented in solving many complex 2D and 3D flows with moving boundary or deformation. Deng et al. [26] proposed a method to scale the force on solid interface to the corresponding grids nearby through a linear interpolation. The treatment of the immersed boundary has been successfully implemented in the study of 3D complex flow over a giant danio fish. Sui et al. [27] proposed a hybrid immersed boundary and multi-block lattice Boltzmann method. The incompressible flows and moving boundary interactions were simulated to understand the principal of propulsion force generation for fish swimming. Coupled the structural analysis with IBM, Shi and Lim [28] simulated the fluid-structure interaction modeling for 3D plate and sphere deformation. Borazjani et al. [29] extended the IBM on the curvilinear domain and hence was able to solve the 3D pulsating blood flow through a heart valve. Pan and Shen [30] proposed and implemented an implicit pressure correction based IBM on Cartesian grids with local refinement. Multigrid method was also adopted to improve the convergence and stability, and the unsteady incompressible flow was simulated and validated in a 2D cylinder and 3D sphere. Finite-element method is less favor

for adoption by CFD solver development due to non-conservative discretization sometimes [74]. While embedded-boundary method and direct body forcing approach was used by Vanella et al. [75] in a finite-difference solver for incompressible flows to enforce the boundary conditions on a complex moving body. With that they demonstrated the robustness and accuracy for the proposed formulation.

1.3.2 Velocity correction based IBM

Shu and co-workers studied various virtual boundary force IBM applications and concluded that the no-slip boundary condition on the immersed wall boundary is not directly enforced through the iterative process in seeking the zero restoring force [22]. Due to numerical errors, the no-slip condition is only satisfied approximately when zero restoring force is achieved. This is demonstrated in Figure 1.3 (a) for the plot of streamlines near the boundary as predicted through the conventional virtual boundary force IBM [31]. Since the streamlines may pass through the boundary and enter into the solid body, the actual physics of the solid boundary is not fully satisfied. To overcome this problem, Shu and co-workers proposed an innovative velocity correction based IBM, where the no-slip boundary condition was directly enforced through the velocity correction at the grid points near the boundary [22], [31]-[34]. Using the new velocity correction approach, the no-slip boundary condition can be enforced directly and fully satisfied. The computed streamlines, as shown in Figure 1.3 (b), do not pass through the boundary. This implies that the physics of solid wall boundary is represented accurately and logically. The force introduced by the solid body can be calculated from

the converged flow field.



(a) streamlines obtained by conventional IBM

(b) streamlines obtained by velocity correction IBM

Figure 1.3 Streamlines obtained by IBM ([22], [31])

The velocity correction based IBM named IBVCM was proposed by Shu and co-workers [22]. The concept is that the boundary velocity is corrected directly to satisfy the no-slip boundary condition and the correction of velocity will be transformed into the nearby Cartesian grids linearly. The body force generated by the solid body is calculated by integration of momentum equations over a local domain enclosed the solid body. The advantage of the IBVCM is that the physical no-slip boundary condition is enforced directly. The IBVCM was validated numerically by the incompressible flow over a 2D circular cylinder for Reynolds number of 20, 40, 80, 100 and 150. Good convergence performance and robustness of the method has been demonstrated as compared with the conventional IBM. The method was applied for the study of an unsteady flow past an undulating 2D fish-like body by Shu et al. [33] to understand the mechanics of propulsion force generation for the undulatory swimming fish. The study shows that the IBVCM is an

effective approach in capturing the flow feature for the problem with moving boundary.

Wu and Shu [32] recently proposed an implicit velocity correction-based immersed boundary-lattice Boltzmann method (IB-LBM), of which the velocity correction at the grid points near the boundary is determined from enforcement of no-slip condition directly. The body force such as drag force and lift force can be computed directly from the obtained velocity correction through the relationship between the force and velocity correction. The performed validation work demonstrates that the method can satisfy the no-slip boundary condition accurately and offer a great potential for practical application due to the ease in implementation. Wu and Shu [34] used the method successfully in a study of the laminar flow behind a stationary circular cylinder with a flapping plate. The comprehensive results demonstrate that the velocity correction-based IB-LBM can be easily applied to study incompressible flows with moving boundary easily.

The major advantage introduced by the velocity correction based IBM, either IBVCM or IB-LBM, is that it is simple in implementation and enforces the no-slip condition on boundaries directly and accurately for incompressible flow problems. The body force can be computed from the velocity correction.

To summarize, various IBM approaches have been successfully implemented in the numerical studies of 2D and 3D incompressible viscous flows with stationary boundaries and moving bodies/boundaries in both steady state and

unsteady state. The representation of wall boundary in incompressible flows can be implemented through either body force correction or velocity correction.

Unlike incompressible flows, wall boundary condition for compressible flows is not represented by the body force or velocity field alone. Other flow variables including pressure, density and temperature are affected by the wall boundary as well, and the values of which need to be determined on the boundary. For compressible inviscid flows, the velocity on the wall boundary is unknown and then, there is no target velocity field for the implementation of IBM. Moreover, the presence of shock waves in compressible flows adds to the challenges in the implementation. Hence the use of IBM in modeling immersed boundary for compressible flows on Cartesian grids is not yet found in the literature.

Inspired by the concept of the velocity correction-based IBM that was proposed to represent the physic boundary condition directly in overcoming the limitations of the conventional body force correction-based IBM, the immersed boundary concept can be implemented to model immersed wall boundary for compressible flows if the physical boundary condition can be enforced.

1.4 Motivation in current study

With the achievements of IBM implementation for incompressible flow analyses in mind and the understanding of the advantages and limitations of

the various methods implemented on Cartesian grid-based solvers for compressible flow simulations, the objective of the current work is to develop simple and accurate immersed boundary methods to simulate compressible inviscid flows on a simple and efficient Cartesian grid-based solver. For the first time, we are able to implement the widely adopted IBM concept for incompressible flows to solve compressible inviscid flows. Numerical simulations for compressible inviscid flows play an important role to understand flow patterns, shock wave capturing and lift force prediction for compressible flows at high Reynolds numbers as the viscous effect of the air has very minor contribution. The main benefit is that the development efforts and the computational time can be reduced significantly from the saving of viscous boundary layer capturing and turbulence modeling as the viscous effect is ignored.

In this study, a new flux correction-based immersed boundary method (FC-IBM), a modified local DFD method (LDFD) and a local DFD based immersed boundary method (LDFD-IBM) are introduced and integrated with the Cartesian grid-based adaptive Euler solver for CFD analyses of compressible inviscid flows. The validation and results for various test problems show that FC-IBM and LDFD-IBM are simpler in implementation, while LDFD is more accurate and robust.

The Cartesian grid-based adaptive Euler solver is developed using the finite volume method. The AMR technique is adopted and implemented in the solver to refine the mesh near the boundary and high flow-gradient regions to

improve the resolution for boundary representation and solution accuracy in particular for shock wave capture, respectively. The ghost-cell method in representing the wall boundary is implemented and validated in the current adaptive solver.

1.5 Outline of the Thesis

Chapter 2 describes the governing equations and the development of the finite volume method based Euler solver in 2D. The flux is obtained via the HLLC scheme on the cell interface. Five second-order flux schemes are implemented, validated and compared. In the later part of the chapter, the implementation of tree structure AMR technique is given with the validation results and the analysis of performance enhancement by using AMR is presented.

In Chapter 3, the ghost-cell method is introduced and implemented in the current Cartesian grid-based adaptive solver to model the immersed wall boundary for compressible inviscid flows. Validation and some test cases are presented.

A new flux correction based-immersed boundary method (FC-IBM) is presented in Chapter 4. The methodology, implementation procedures, validation, test cases and discussion are given in details. In Chapter 5, a modified local DFD method (LDFD) and its extension on immersed boundary method (LDFD-IBM) are proposed to represent the wall boundary condition. The concept of LDFD and LDFD-IBM for treating wall boundary in compressible inviscid flows is explained. The methodology, implementation procedures, validation, test cases and discussion are given in details.

Chapter 6 illustrates the development of the 3D adaptive solver. The validation of accuracy and adaptive performance are presented. The FC-IBM and LDFD will be implemented on the 3D solver and the results will be compared.

Chapter 7 explains the extension of the current adaptive compressible inviscid flow solver into an adaptive solver for laminar viscous flow. Test cases of laminar viscous flow past a circular cylinder and a NACA0012 airfoil are presented.

Conclusions and recommendation for future works are given in Chapter 8.

Chapter 2
Adaptive Cartesian Grid Euler Solver

The two-dimensional compressible inviscid solver is developed in this chapter by using finite volume method. In this framework, the adaptive solver is also developed and validated. The details for the development of 2D Euler solver and the implementation of solution adaption capability will be discussed in the following context.

2.1 Governing equations

Two-dimensional compressible Euler equations expressing the conservation laws of mass, momentum and energy are given in the following vector forms:

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}=0, \tag{2.1}$$

with

$$U=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ E\end{bmatrix},\qquad F=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ u(E+p)\end{bmatrix},\qquad G=\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ v(E+p)\end{bmatrix}. \tag{2.2}$$

where U in the equation is the state vector of the conservative variables. F and G are the flux vectors. The physical variables in the above equations are density (ρ), velocity components (u) and (v), internal energy (E) and pressure (p). The pressure p is obtained using the equation of the state for ideal gas.

$$p = (\gamma - 1)\left[E - \frac{1}{2}\rho\left(u^2 + v^2\right) \right]. \tag{2.3}$$

The constant γ in the equation is the adiabatic index. For ideal gas, γ=1.4.

2.2 Finite volume discretization and HLLC scheme

In the present work, finite volume method (FVM) is used for the development of the base Euler solver for compressible flows on Cartesian grid. For a finite control volume, the numerical discretization of the governing equation (2.1) by the first order time evolution scheme will be

$$U_c^{n+1} = U_c^n - \frac{\Delta t}{A_c}\sum_i \left(f_i \vec{l}_i \cdot \vec{n}_i \right), \tag{2.4}$$

here, U_c^{n+1} and U_c^n are the states of the conservative variables at $(n+1)^{th}$ and n^{th} time levels; Δt is the time step; A_c is the area of the control cell; $\vec{l}_i$, f_i and $\vec{n}_i$ stand for the length, the flux and the normal direction for the $(i)^{\text{th}}$-interface of the cell. The solution $\boldsymbol{U}$ is defined on the cell center.

The time step is determined by the minimum time step of all the control cells, and it is calculated by

$$\Delta t = \min\left\{ CFL \times \frac{A_c}{\sum\left(|u_n| + a_c\right) \cdot \left\|\vec{l}\right\|} \right\}, \tag{2.5}$$

where, the denominator in the above equation is to summate the terms on all four interfaces of the cell. CFL is Courant-Friedrichs-Lewy number, and is set as CFL=0.3~0.5 in current work.

From equation (2.4), it is known that calculation of the flux term f_i on cell interface is crucial for the solver. The HLLC scheme, or the Harten, Lax and van Leer approximate Riemann solver with the contact wave restored scheme, is adopted due to its good resolution for shocks, contact waves and the ability in preserving the positivity of density and internal energy [35]. It has been proved to give robust and accurate solutions for most of compressible flows [36], [37].

On the Cartesian grid system, the flux term f_i on a cell interface can be expressed as following using the left and right cells of the interface

$$f_i = f^{HLLC}\left(U^L, \quad U^R, \quad \vec{n}\right) = \begin{cases} f^{*L} & s_m \geq 0 \\ f^{*R} & s_m \leq 0, \end{cases} \tag{2.6}$$

with

$$f^{*L} = \Phi\left(U^L, \quad \vec{n}\right) + s_L\left(U^{*L} - \Phi\left(U^L, \quad \vec{n}\right)\right), \tag{2.7}$$

$$f^{*R} = \Phi\left(U^R, \quad \vec{n}\right) + s_R\left(U^{*R} - \Phi\left(U^R, \quad \vec{n}\right)\right). \tag{2.8}$$

Here $\vec{n}$ is the edge vector $\vec{n} = \langle n_x, n_y \rangle$. For horizontal edges $\vec{n} = \langle 1,0 \rangle$, for vertical edges $\vec{n} = \langle 0, \ 1 \rangle$. The terms S_L and S_R are two intermediate signal speeds of HLLC scheme on the left-side and right-side of the interface, and are given in equations (2.9) and (2.10). The term $\Phi\left(U, \quad \vec{n}\right)$ is the flux vector, given in equation (2.11).

$$s_L = \min\left(u_n^L - a^L, \quad \tilde{q} - \tilde{a}\right), \tag{2.9}$$

$$s_R = \max\left(u_n^R + a^R, \quad \tilde{q} + \tilde{a}\right), \tag{2.10}$$

$$\Phi(U,\quad \vec{n}) = \begin{pmatrix} \rho(un_x + vn_y) \\ \rho u(un_x + vn_y) + pn_x \\ \rho v(un_y + vn_x) + pn_y \\ (E+p)(un_x + vn_y) \end{pmatrix}, \qquad \vec{n} = \langle n_x, n_y \rangle \text{ for } \boldsymbol{F} \text{ and } \boldsymbol{G}. \tag{2.11}$$

In the above equations, a^L and a^R are the left sound speed and the right sound speed; $\tilde{a}$ and $\tilde{q}$ are the Roe average sound speed and average velocity, respectively. They are determined by equation (2.12). The two intermediate states $U*^L$ and $U*^R$ in equations (2.7) and (2.8) and the signal speed of the contact wave s_m in equation (2.6) are given in the following equations:

$$\begin{cases} a^{[L,R]} = \sqrt{\gamma (p/\rho)^{[L,R]}} \\ \tilde{a} = \left(a^L \sqrt{\rho^L} + a^R \sqrt{\rho^R}\right) \Big/ \left(\sqrt{\rho^L} + \sqrt{\rho^R}\right) \\ \tilde{q} = \left(V_n^L \sqrt{\rho^L} + V_n^R \sqrt{\rho^R}\right) \Big/ \left(\sqrt{\rho^L} + \sqrt{\rho^R}\right), \end{cases} \tag{2.12}$$

$$U*^L = \left(\frac{s_L - u_n^L}{s_L - s_m}\right) \begin{pmatrix} \rho^L \\ \rho^L \left[u_x^L - (u_n^L - s_m) n_x\right] \\ \rho^L \left[u_y^L - (u_n^L - s_m) n_y\right] \\ E^L + E_m^L \end{pmatrix}, \quad E_m^L = \frac{p*^L s_m - p^L u_n^L}{s_L - s_m}, \tag{2.13}$$

$$U*^R = \left(\frac{s_R - u_n^R}{s_R - s_m}\right) \begin{pmatrix} \rho^R \\ \rho^R \left[u_x^R - (u_n^R - s_m) n_x\right] \\ \rho^R \left[u_y^R - (u_n^R - s_m) n_y\right] \\ E^R + E_m^R \end{pmatrix}, \quad E_m^R = \frac{p*^R s_m - p^R u_n^R}{s_R - s_m}, \tag{2.14}$$

$$s_m = \frac{\rho^R u_n^R \left(s_R - u_n^R\right) - \rho^L u_n^L \left(s_L - u_n^L\right) + \left(p^L - p^R\right)}{\rho^R (s_R - u_n^R) - \rho^L (s_L - u_n^L)}. \tag{2.15}$$

2.3 Second-Order flux solver

The previous section describes the first-order HLLC scheme. The flux on cell interfaces is calculated simply using the values of flow variables on the centers of two neighbor cells, or assume piecewise constant for all the flow variables inside each cell. To achieve more accurate solutions, in particular for cases when shocks, discontinuities or large gradients exist in the system, high order schemes are usually adopted. For finite volume method, Monotone Upstream-centered Schemes for Conservation Laws, or MUSCL scheme in short, is a popular method that can provide second-order spatial accuracy. The concept is to replace the piecewise constant approximation of Godunov's scheme by the reconstructed states. For each cell, the left and right states are obtained via reconstruction through the slope limiter, and then used to calculate fluxes on the interfaces.

There are tens of common slope limiter functions for the reconstruction of the left and right states. The limiter function is constrained to be greater than or equal to zero, i.e. $r \geq 0$. The parameter $\boldsymbol{r}$ represents the ratio of successive gradients on the solution mesh, as illustrated in Figure 2.1. It is computed via:

$$r_i = \frac{u_i - u_{i-1}}{u_{i+1} - u_i}. \tag{2.16}$$

The following five typical flux/slope limiter functions $\phi(r)$ are adopted and implemented in the current solver to achieve the second-order accuracy. For the Osher and Sweby limiters, β=1.5 is used and tested in the current solver.

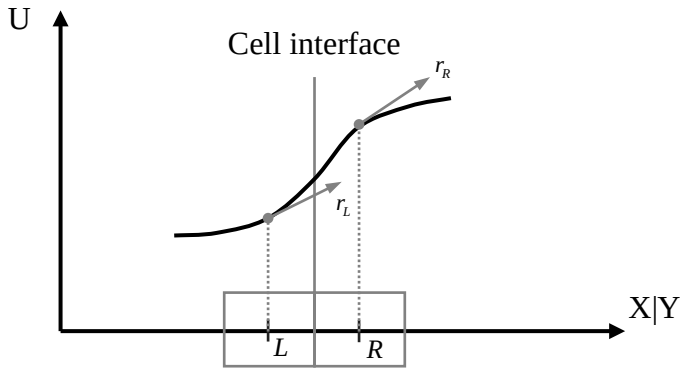


Figure 2.1 Ratio of successive gradients on two neighbor cells

Minmod: $$\phi_{mm}(r)=\max\left[0,\min(1,r)\right]. \tag{2.17}$$

Osher: $$\phi_{os}(r)=\max\left[0,\min(r,\beta)\right],\quad (1\le\beta\le 2). \tag{2.18}$$

Sweby: $$\phi_{sw}(r)=\max\left[0,\min(\beta r,1),\min(r,\beta)\right],\quad (1\le\beta\le 2). \tag{2.19}$$

Van Leer: $$\phi_{vl}(r)=\frac{r+|r|}{1+|r|}. \tag{2.20}$$

Van Albada: $$\phi_{va}(r)=\frac{r^2+r}{r^2+1}. \tag{2.21}$$

With the limiters evaluated via the flux limiter functions in equations (2.17)-(2.21), the second-order scheme can be achieved by the new states $\left(U^{L+},U^{R-}\right)$ interpolated by:

$$\begin{cases} U^{L+}=U^{L}+\phi(r_i)\cdot dh/2 \\ U^{R-}=U^{R}-\phi(r_i)\cdot dh/2. \end{cases} \tag{2.22}$$

The flux calculation will use the new states $\left(U^{L+},U^{R-}\right)$ to replace the current states $\left(U^{L},U^{R}\right)$ in equations (2.6)-(2.8).

2.4 Implementation of boundary conditions

There are three types of boundary in numerical simulations for compressible inviscid flows: inlet, outlet and wall. In the finite volume method, the flux calculation on cell interface needs to use two cells, normally named left cell and right cell. Hence for the implementation of boundary conditions, one additional cell layer is extended away from the boundary surfaces by mirroring the next interior cells of the boundary surfaces. Those cells are virtual cells, or named as ghost cells by many researchers. Although the governing equations are not solved on these virtual cells, the states on these virtual cells are updated according to the appropriate boundary condition every time step.

For inlet boundary, the flow variables of density, velocity components, pressure and internal energy are set as given inlet conditions if the inflow is in supersonic state. When the inflow is in subsonic state, the pressure is extrapolated from the neighbor interior cell and the rest variables are set according to the given inlet condition.

Next, for outlet boundary, when it is under supersonic condition, all the flow information can be extrapolated from the interior neighbor cells simply. When it is under subsonic condition, pressure will be set as the background pressure and the rest of variables can still be extrapolated from the interior neighbor cells.

For inviscid flow, wall boundary condition becomes the same as reflecting

wall condition or same as a “symmetrical” condition. The density, pressure, tangent velocity and internal energy in the virtual cells are the same as those of the neighbor interior cell. The normal velocity is set to be negative of that on the neighbor interior cell. This will ensure the no-penetration wall boundary condition. The boundary condition for curved wall boundary is special and needs to be treated in different ways. The treatment of curved boundary will be discussed in the following Chapters 3, 4 and 5.

2.5 Solution adaptive method

The quad-tree data structure for storing of mesh cells is adopted in the current work for the development of the adaptive mesh refinement (AMR) solver on Cartesian grids. To make the adaptive process efficiently during the solution evolution, all the connectivity and hierarchy information of the Cartesian cells and cell interfaces are explicitly stored for quick retrieval and use. The current solver is in unstructured Cartesian grid family. In this category, the mesh adaption is only performed at regions where high resolution is necessary such as the place where shock wave is located. In this manner, high resolution of the problems can be achieved without exaggerative computational load. This advantage of adaptive solver is remarkable compared with the solver based on uniform mesh, in which the whole domain must be refined when high resolution is required.

The current solver and the solution adaption process are developed by the object-oriented programming language C/C^{++}. For the 2D solver, three major objects are defined in individual data structure and they are node, edge and

cell. A cell object consists of four edge objects that enclose the cell; an edge object consists of two node objects that are the two ends of the edge. The object trees of cell, edge and node are listed in Figure 2.2. The objects **Tcell**, **Tedge** and **Tnode** define the control cell, the cell interface or edge and the grid node, respectively. The four edges of a cell are stored in ***Edges[4]**; the neighbor cells of an edge are stored in ***neighborCells[2]**. Similarly, the **childCells** and **parentCell** are defined accordingly. The variables defined under the objects can be very easily indexed and accessed when needing to use them. New data information can be added into the object easily when needed.

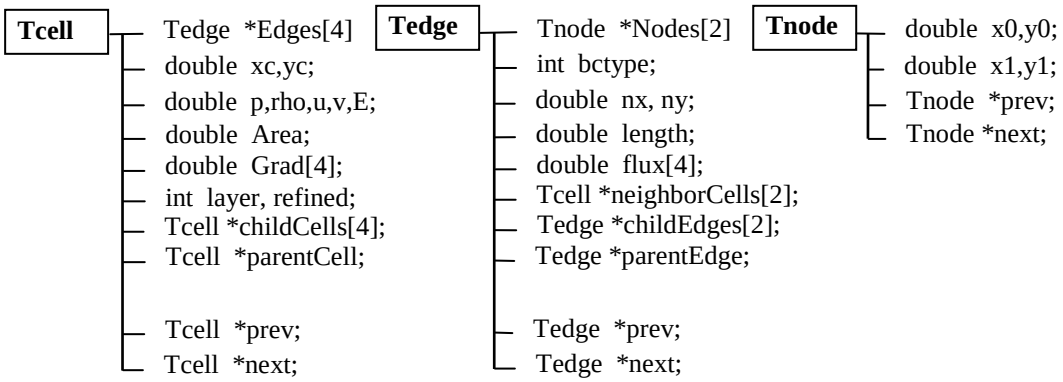


Figure 2.2 Data structure of the objects (cell, edge and node) for 2D solver

For the quad-tree mesh adaption on Cartesian grids, a parent cell will be refined with four child cells. In this process, a maximum 12 new edges will be created and a maximum of 5 new nodes will be added, as illustrated in Figure 2.3. New nodes are appended to the existing node object list, while new edges and new cells are inserted before their parents in the corresponding object list. To decide whether a cell can be refined, its adaption level and status of the neighbor cells must be considered to avoid duplicate creation of new edges and nodes. For example, if all the four neighbor cells of a cell are already

refined to a finer level, then only one new node at the cell center and four new edges forming the cross frame (+) inside the cell will be created.

In order to ensure the smooth transition from coarse mesh region to fine mesh region and the accuracy of the solution, the level difference of two neighbor cells for an edge is restricted to one. So the shadowed refined cell marked as dotted-line in Figure 2.3 (c) is not allowed. As illustrated in the diagram, checking of refinement dependency is needed and the refinement of the dependent cells must be performed first before the marked cells are refined. In the current work, the check of refinement dependency is limited to one cell away from the cell marked for refinement to avoid recursion need of adaption for too many cells. This makes the code implementation easier and keeps the adaption solver efficient. The validation of the accuracy benefit and efficient of the solver will be presented in the following sections.

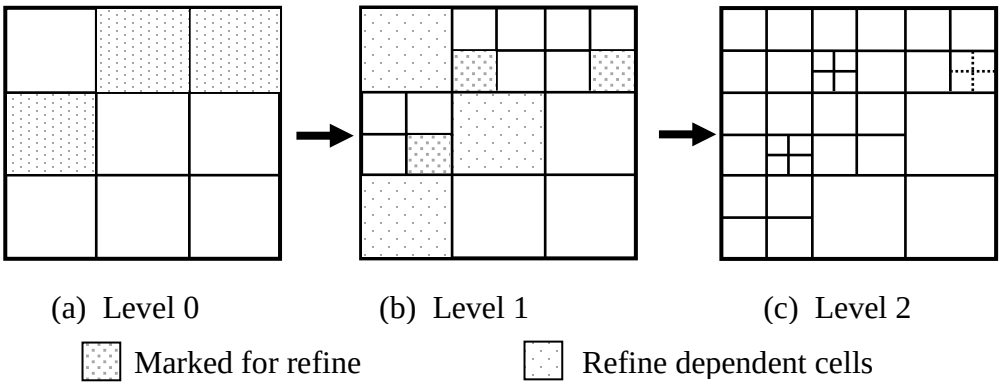


Figure 2.3 Illustration of mesh refinement process

Beside the refinement process discussed above, coarsening process is also necessary in AMR technique to reduce the number of grid used. In the

coarsening process, only parent cells can be chosen for coarsening but the grand-parent cells will be ignored until their children cells are not marked as refined. In addition, it is necessary to check the adaptive status of the neighbor cells before the confirmation of the coarsening. In specific, if any neighbor cells are grand-parent cells, then the current cell will not be coarsened to ensure that the level difference is limited to one. Particularly, to perform coarsening, the 4 edges in the cross (+) position and the central node will be deleted from the object list. The other 8 sub-edges and 4 nodes can be deleted if they are not associated with the neighbor cells. All the deleted edges, nodes and the coarsening cells will be deleted from the corresponding object list and then freed from the memory to save the memory usage.

As introduced previously, on an adaptive refinement grid, accurate calculation of edge flux is of utmost important for the solver. Great attention must be paid on the flux calculation especially on the adapted edges. When a cell is refined, the cell will become a parent cell and has its child cells; the four edges formed the cell will have their child edges. Flux should be calculated on the children edges only and then their contribution to the left and right cells are directly updated. It is not necessary to perform the flux calculation for the refined edges. As illustrated in Figure 2.4, flux calculation will be performed on edges AB and BC as f_1 and f_2, but it is no need for edge AC. The flux f_1 will be contributed to cell[n] and cell[n+2]; the flux f_2 will be contributed to cell[n+1] and cell[n+2]. So to cell[n+2], though the flux through its edge AC (or A′C′) is not directly computed, the total flux through this edge is still counted by edges AB and BC.

In the current work, flux is calculated on all the valid children edges and the non-refined edges as described above. After the flux on all edges is evaluated, solution at the next time step in the term of the state U_c^{n+1} for all the non-parent cells can be obtained via equation (2.4). For the parent cells, their new state will be determined by summation of the state from all four child cells. Having the new state U_c^{n+1}, the physical variables of density, pressure, velocity component and internal energy at the cell centers can be obtained.

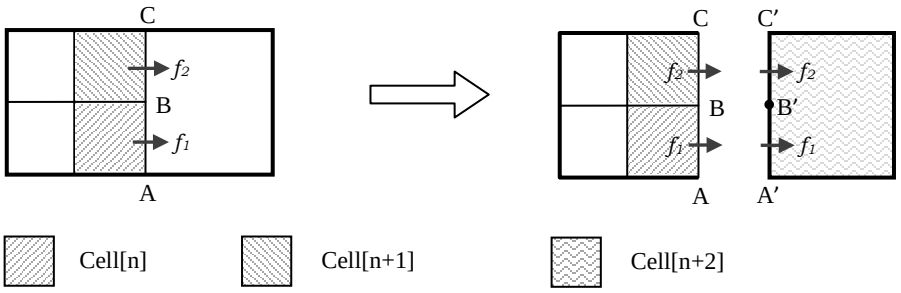


Figure 2.4 Flux calculation for the edges of a refined mesh cell

AMR technique is useful to cluster fine meshes in the region that is critical or experiencing high flow gradients while use coarse mesh in the regions that has less variation. For compressible flows specifically, regions close to shock wave, rarefaction and contact wave in general have high flow gradients and require very high mesh resolution. Hence, adaptively locating fine mesh into these regions is very useful for accurate capturing of the sudden changes in solution such as shock waves. On the other hand, it is also essential to coarsen refined cells when the gradients at the cells become small. This effectively reduced the total number of cells used, and consequently improves the computational efficiency. In the current adaptive solver, the density gradient is chosen as the refinement indicator because it essentially conveys the flow

changes of rarefaction, shock and contact waves. In contrast, pressure and velocity gradients only reflect the flow changes in the regions of rarefaction and shock normally. However, these variables can also be used as the adaptive criterion if needed. The mesh refinement can also be chosen based on the location when it is needed, for example to cluster fine meshes near the critical geometry.

In summary, the flow chart of the current adaptive solver as described above is given in Figure 2.5. The solution adaption part is developed as a module component in the solver. So it can be by-passed by giving a negative switch indicator. This also facilitates the ease in the debug and troubleshooting for the code development. Furthermore, it makes the comparison for solutions computed on uniform meshes and adaptive meshes very simple and easy.

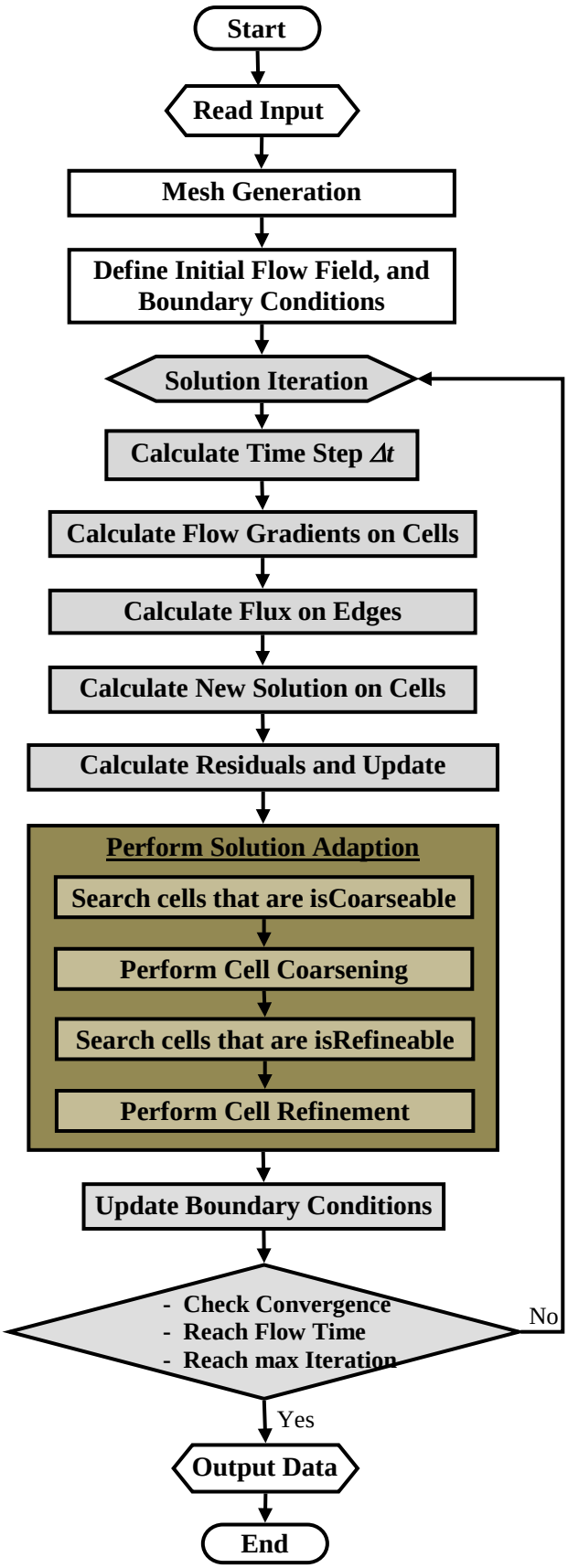


Figure 2.5 Flow chart of the current adaptive Cartesian grid Euler solver

2.6 Test cases and discussion

For compressible flows, shock waves and Riemann discontinuous flow structure are very common. These flow structures involve very high flow gradient and are very challenging for numerical simulation. In this section, results for six compressible flow problems are presented to illustrate the accuracy and effectiveness of the current adaptive solver.

2.6.1 Sod's shock tube problem

The first test case is a modified Sod's shock tube problem. The initial condition inside the o unit length tube is:

$$x < 0.3 \qquad \{\rho_L,\ u_L,\ P_L\} = \{1.0,\ 0.75,\ 1.0\},$$

$$x > 0.3 \qquad \{\rho_R,\ u_R,\ P_R\} = \{0.125,\ 0.0,\ 0.1\}.$$

The Sod's shock tube problem is a one-dimensional problem and the analytical solution is available [35]. The solution has a right shock wave, a right travelling contact wave and a left sonic rarefaction wave. It is solved in 2D with the current adaptive solver in a domain of 1×0.1, and an initial uniform mesh of 100x10 is applied.

Figure 2.6 shows the predicted solution on the initial mesh without refinement and 3-level adaption. The numerical results are compared to the analytical solution at dimensionless time t=0.2. It can be seen that among the profiles of density, pressure, velocity and internal energy, only density profile can reflect all the three waves. It authenticates the choice of density gradient as an adaption indicator. It can be seen from the computed profile that all the three

captured waves compare very well with the exact solution. The solution with 3-level adaption matches to the exact solution much more accurately than that of the initial uniform mesh. The mesh distribution for 3-level adaption is shown in Figure 2.7. It is very clear that fine meshes are adapted to the high flow gradients at the three areas representing the rarefaction wave, the contact wave and the normal shock wave. As a result, it improves the accuracy for the wave capture.

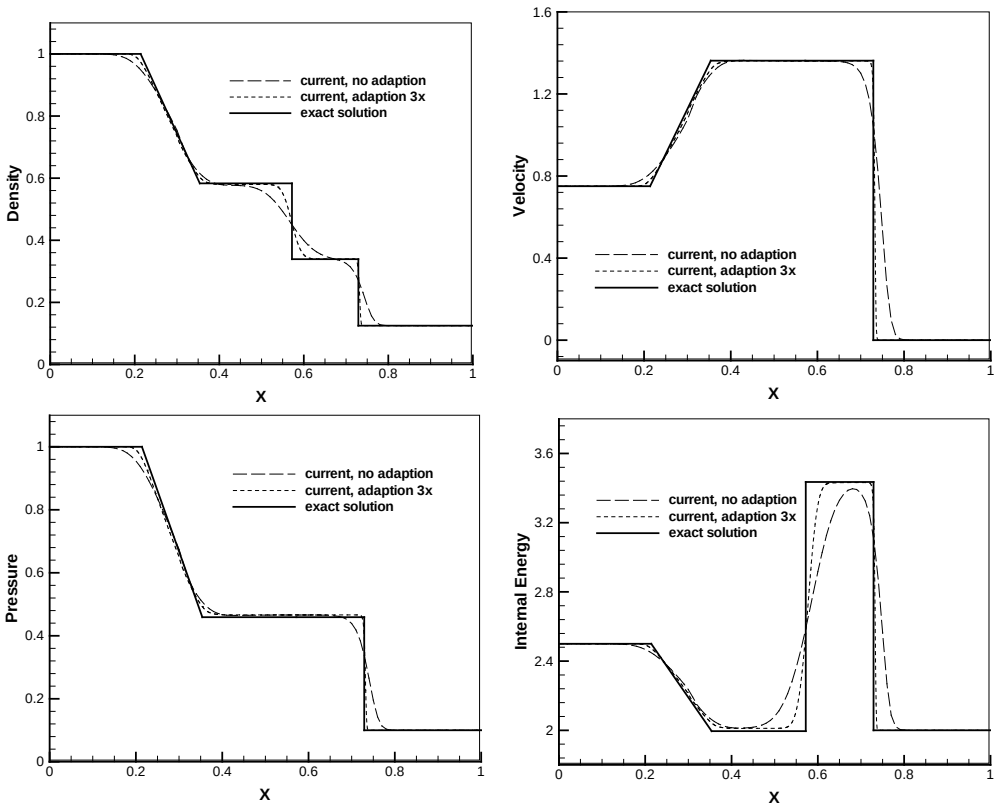


Figure 2.6 Sod's Shock Tube Problem: comparison of predicted solution between uniform no adaption mesh and 3-level adaption mesh

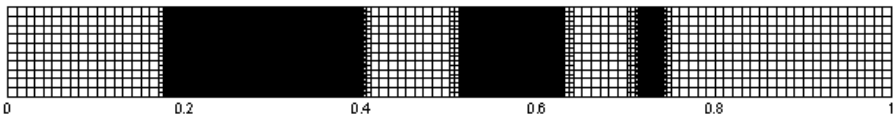


Figure 2.7 Sod's Shock Tube Problem: Mesh distribution with 3-level adaption

2.6.2 Lax shock tube problem

The second test case is the Lax shock tube problem [35], which is very similar to the Sod's shock tube problem. The initial condition of Lax shock tube problem is:

$$x < 0.4 \qquad \{\rho_L,\ u_L,\ P_L\} = \{5.99924,\ 19.5975,\ 460.894\},$$

$$x > 0.4 \qquad \{\rho_R,\ u_R,\ P_R\} = \{5.99924,\ -6.19633,\ 46.0950\}.$$

The Lax shock tube problem is a very severe test. The flow consists of three strong discontinuous travelling waves to the right. The flow is also solved in 2D with the current adaptive solver in a domain of 1×0.1 and an initial uniform mesh of 100x10.

Figure 2.8 shows the predicted solution on the initial mesh without refinement and 3-level adaption, which is compared to the analytical solution at dimensionless time t=0.035. Density gradient is used as the indicator for performing solution adaption. As shown in the figure, the adaptive solver can capture all the three strong travelling waves accurately, especially when 3-level adaption is used. The mesh distribution for 3-level adaption is shown in Figure 2.9. This test case shows that the current adaptive solver is very robust to predict such strong waves.

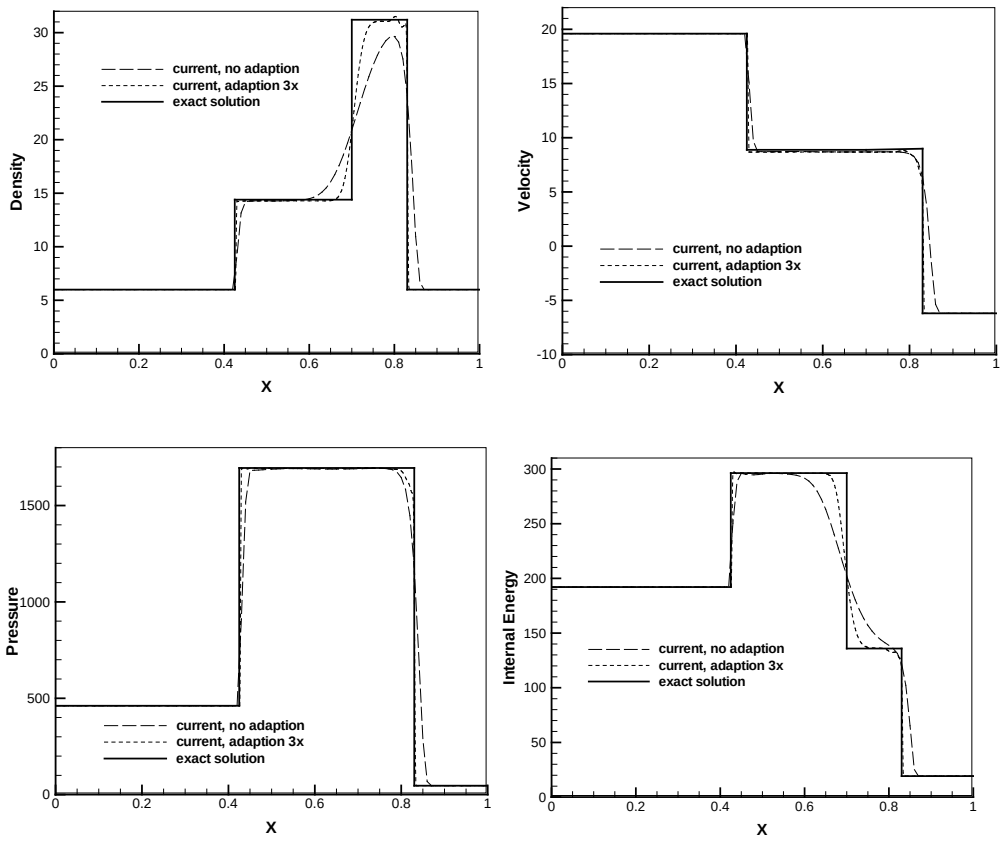


Figure 2.8 Lax Shock Tube Problem: comparison of predicted solution between uniform no adaption mesh and 3-level adaption mesh

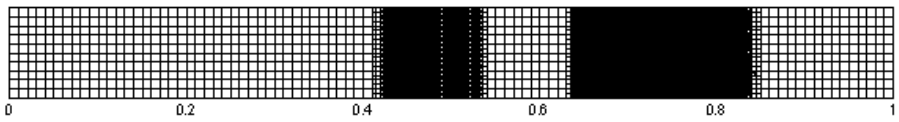


Figure 2.9 Lax Shock Tube Problem: Mesh distribution with 3-level adaption

2.6.3 Two-Dimensional oblique shock wave

A two-dimensional shock wave reflecting from a rigid surface is simulated in this test case. The computational domain is a rectangular domain with length of 4 and width of 1. The initial uniform mesh of 80x20 is used. The left and bottom surfaces are defined as fixed inlet condition, as shown in Figure 2.10. The top surface is defined as wall boundary. Supersonic outflow condition is

defined at the right surface and all flow variables are extrapolated from the interior cells. The initial condition is defined using the left inlet condition.

An incident shock angle of 29° towards the top wall is produced and the free stream Mach number M_∞ is 2.9. The shock is reflected down by the top wall. Figure 2.11 and Figure 2.12 show the simulated density contours in the rectangle domain on the initial uniform mesh and the adaptive mesh. The comparison of the predicted density distribution with the exact solution is presented in Figure 2.13 at y=0.5. The results indicate that the adaptive solver can capture the two incident shock waves sharply and the strength of the two shock waves agrees with the exact solution very well.

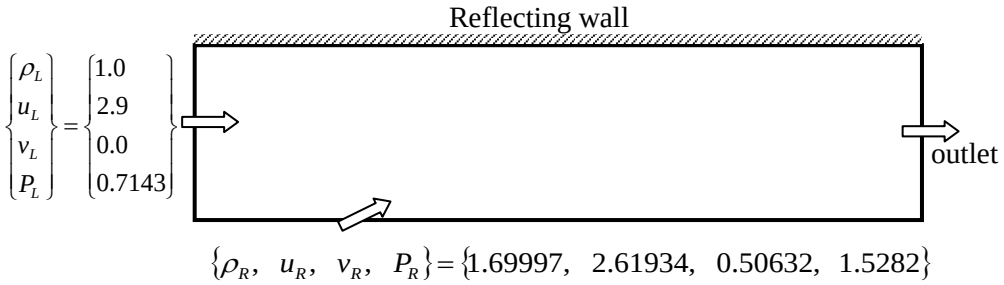


Figure 2.10 Oblique Shock Wave: Computational domain and boundary conditions

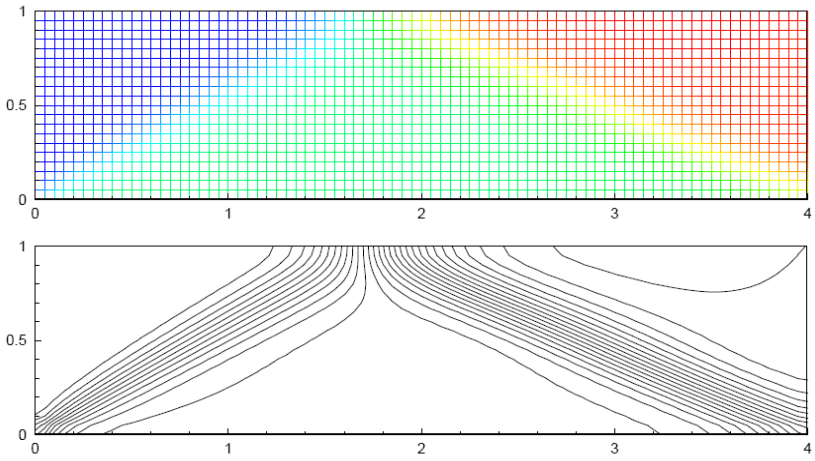


Figure 2.11 Oblique Shock Wave: Mesh distribution (colored by density) and Density contours (1.0 to 2.65, 30 levels) on uniform mesh 80×20.

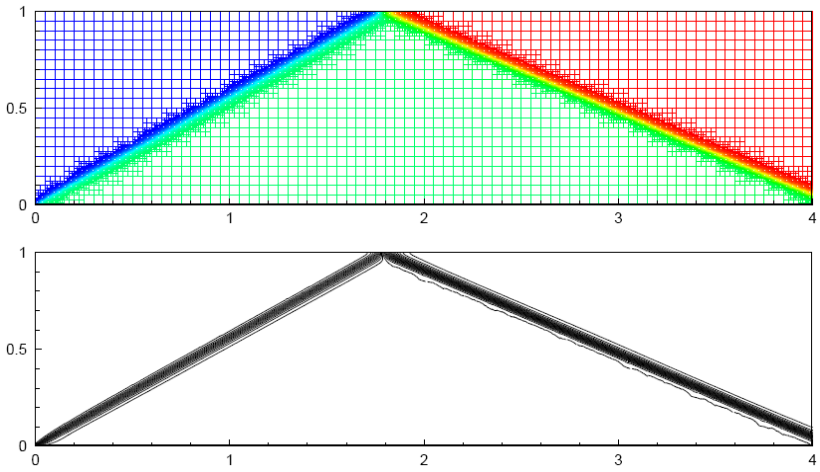


Figure 2.12 Oblique Shock Wave: Mesh distribution (colored by density) and Density contours (1.0 to 2.65, 30 levels) with 3-level adaption on initial Mesh 80×20.

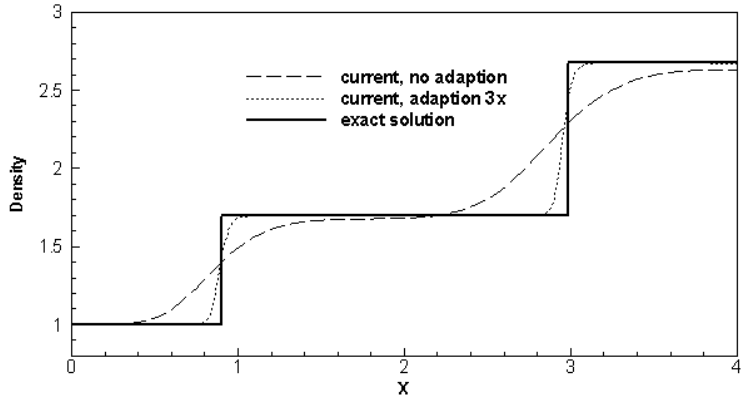


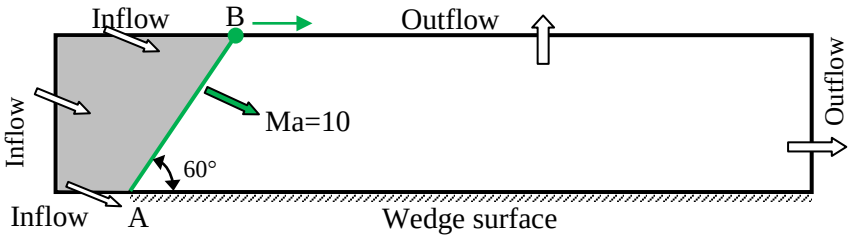
Figure 2.13 Oblique Shock Wave: Comparison of the predicted density profile at y=0.5 with the exact solution

2.6.4 Double Mach reflection

The shock wave strength involved in the previous oblique shock wave reflection problem is relative weak with a low pressure ratio of 2.7. The double Mach reflection problem is another challenging test case studied extensively by many researchers [36]-[39]. The problem is unique according to its high pressure ratio at about 116.5 and a strong normal shock wave with Mach number 10 passing through a 30° wedge. For the convenience of

construction of the computational domain, the reflecting wall or the wedge surface, is put on X-axis starting from X=1/6. The computational domain is chosen to be a rectangle as 4×1. The boundary conditions are as described in Figure 2.14. To be specific, top surface is separated by point-B with the left side of point-B as inflow condition and the right side of Point-B as outflow condition. Point-B will move to the right following the shock wave on the top surface. Position A is aligned to the leading point of the wedge surface, and $\overrightarrow{AB}$ indicates the initial position of the shock wave. Initially a right-moving shock with Ma=10 is positioned at $\overrightarrow{AB}$. The inflow condition is defined as the post-shock condition. Initial uniform mesh of 120×30 is defined in the rectangle domain. The simulation is carried out until a dimensionless time t=0.20.

The computed density contours and the adaptive mesh are shown in Figure 2.15 and Figure 2.16. The main shock and the oblique wave are captured with the initial uniform mesh with no adaption but the detailed shock structure before the main shock near the wedge is smeared. The results with 3-level adaption are in good agreement with other Euler solvers running on finer meshes of 960×240 by Jun et al. [36], and 480x120 by Woodward and Colella [40]. The complex flow structures such as the main three-shock intersection and the structure of the jet formed near the reflecting wall are accurately captured as shown in Figure 2.16.



Inflow: $\{\rho,\ u,\ v,\ P\} = \{8.0,\ 7.1447,\ -4.125,\ 116.5\}$

Outflow: $\{\rho,\ u,\ v,\ P\} = \{1.4,\ 0.0,\ 0.0,\ 1.0\}$

Figure 2.14 Double Mach Reflection: Computational domain and boundary conditions

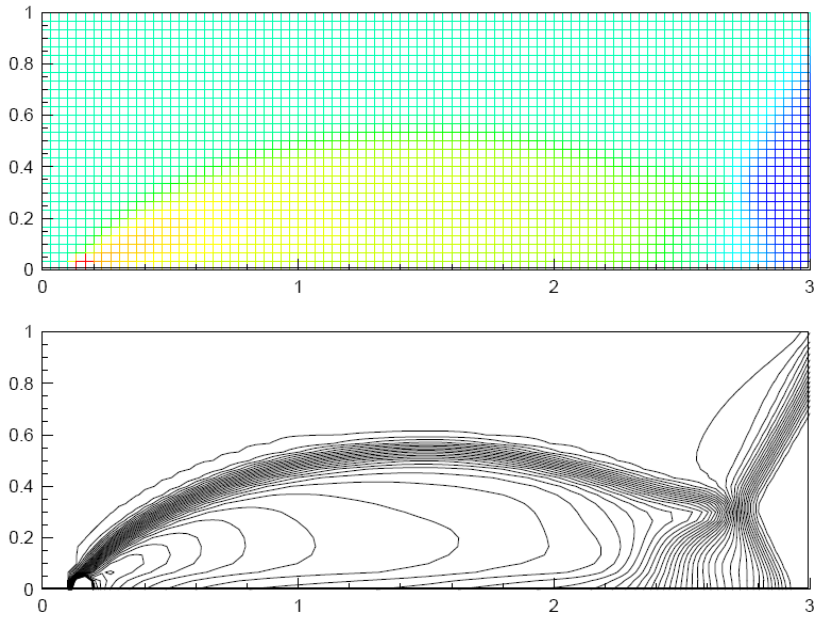


Figure 2.15 Double Mach Reflection: Mesh distribution (colored by density) and Density contours (1.9 to 21, 50 levels) with no adaption on initial Mesh 120×30.

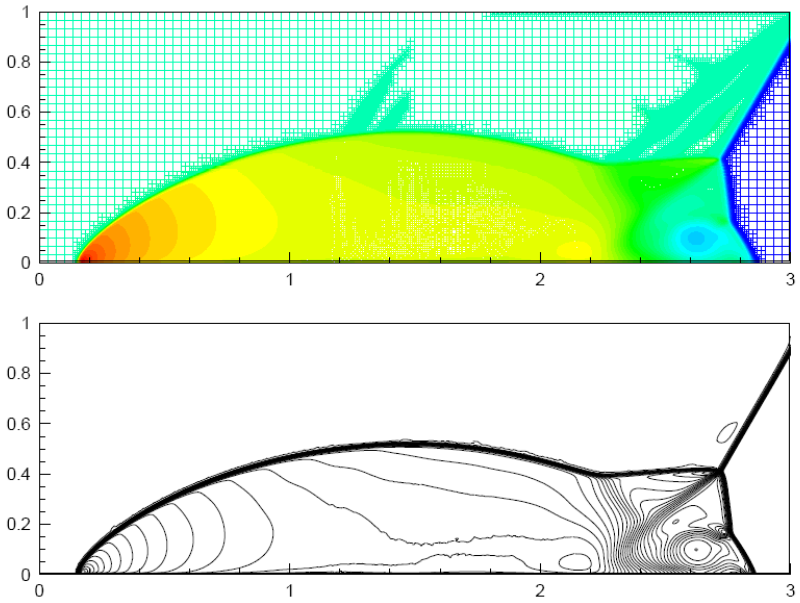


Figure 2.16 Double Mach Reflection: Mesh distribution (colored by density) and Density contours (1.9 to 21, 50 levels) with 3-level adaption on initial Mesh 120×30.

2.6.5 Backward step problem

The backward step problem is to study the shock wave diffraction at a sharp 90° corner. Different shocks from 1.65 to 5.09 were investigated by Sun [38] and Hillier [41]. The computational domain is chosen as 4×4 with a step located at the left-bottom region of the square domain, as shown in Figure 2.17. The boundary conditions to the left of the shock are set as inlet with post-shock conditions. The shock moves at Ma=3 to the right. Initially, a vertical shock is positioned just above the corner, inflow condition is set for the region left to the shock and outflow condition is set for the region right to the shock. Initial mesh is defined as 50×50, with the mesh cells inside the step is being cut out.

The predicted density contours of the shock wave diffraction are shown in

Figure 2.18 and Figure 2.19. The former is obtained on initial uniform mesh and the latter is obtained by 3-level solution adaption. From the comparison, it is noticed that the computed density contours with 3-level adaption is able to reproduce the diffraction shock sharply and the secondary shock waves are also shown. In addition the vortex shock structure at (1, -0.75) is also captured as indicated by the arrow in Figure 2.19. The results are comparable to the results presented by Hillier [41].

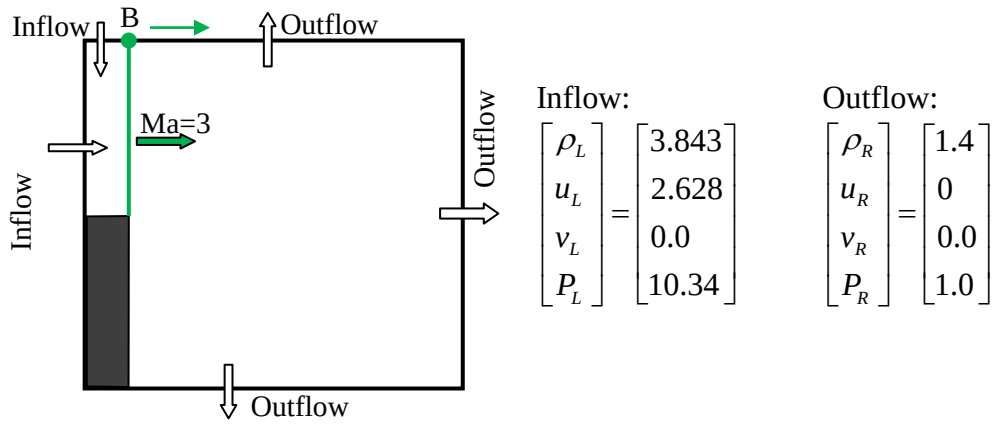


Figure 2.17 Backward step problem: computational domain and boundary conditions

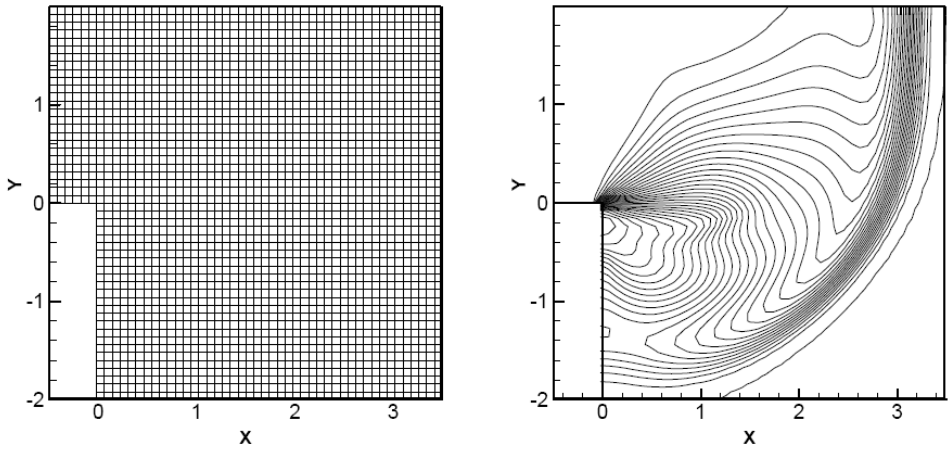


Figure 2.18 Backward step problem: mesh distribution and density contours (0.3 to 3.7, 30 levels) with no adaption on initial mesh of 50×50.

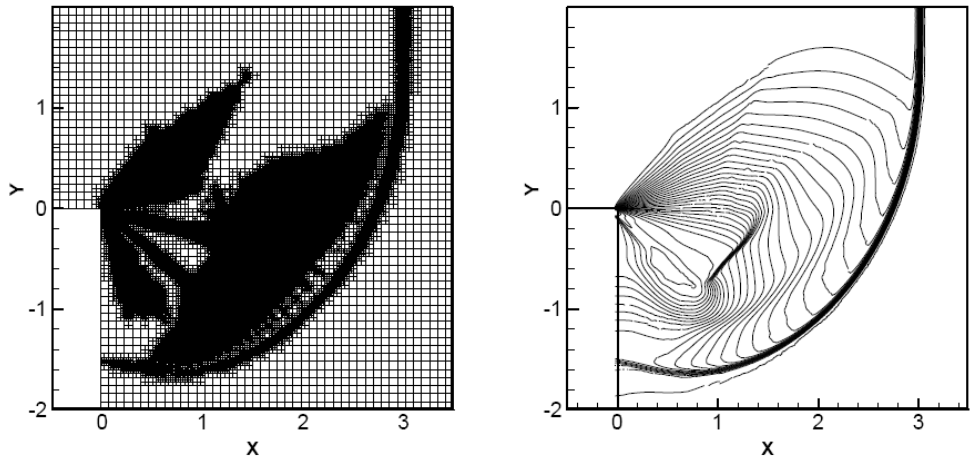


Figure 2.19 Backward step problem: mesh distribution and density contours (0.3 to 3.7, 30 levels) with 3-level adaption on initial mesh of 50×50.

2.6.6 Cylindrical shock explosion

In the Sod's shock tube problem, though it is solved in 2D, the characteristics of the problem are actually in a one dimensional case. The cylindrical explosion problem is like a 2D extension of the Sod's shock tube problem. The computational domain is a 2×2 square with a circle region of radius R=0.4 at the center (Figure 2.20), and inside the circle region, it is filled with high pressure and high density air. The surroundings of the circle are filled with air at low pressure and low density. The explosion of the air inside the circle starts at time t=0 to all directions on the 2D domain. The initial uniform mesh is defined as 50×50.

The predicted density, pressure, velocity and internal energy distribution on line $\overrightarrow{OA}$ in Figure 2.20 at dimensionless time t=0.25 are plotted in Figure 2.21 and compared to the exact solution [35]. The flow structure consists of a circular shock wave travelling away from the center, a circular contact surface travelling in the same direction and a circular rarefaction travelling towards

the center of the circle. This can be observed in the 3D plot of the wave surfaces for density and pressure in Figure 2.22. It can be seen that the predicted profiles with 3-level solution adaption closely match the exact solution. Moreover, the adapted mesh at the solution time t=0.25 is shown in Figure 2.23.

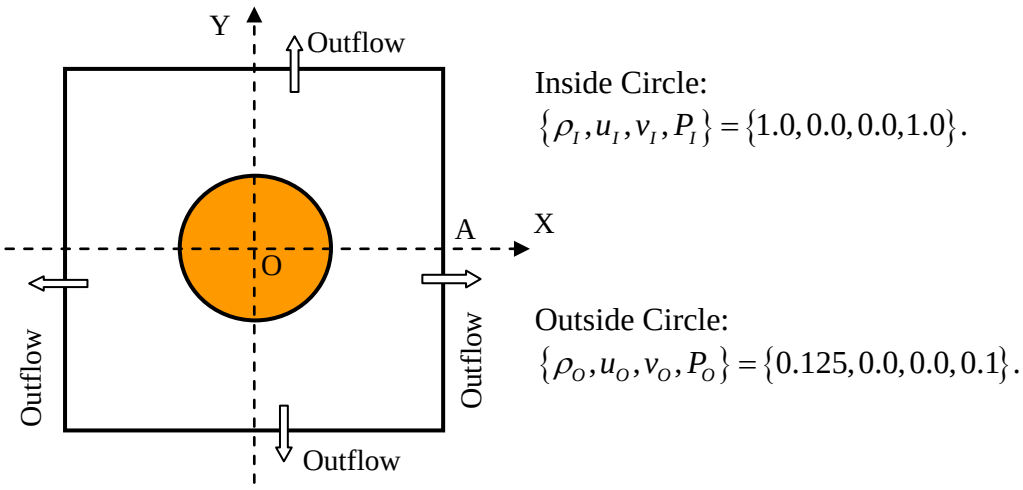


Figure 2.20 Cylindrical explosion: configuration and boundary conditions

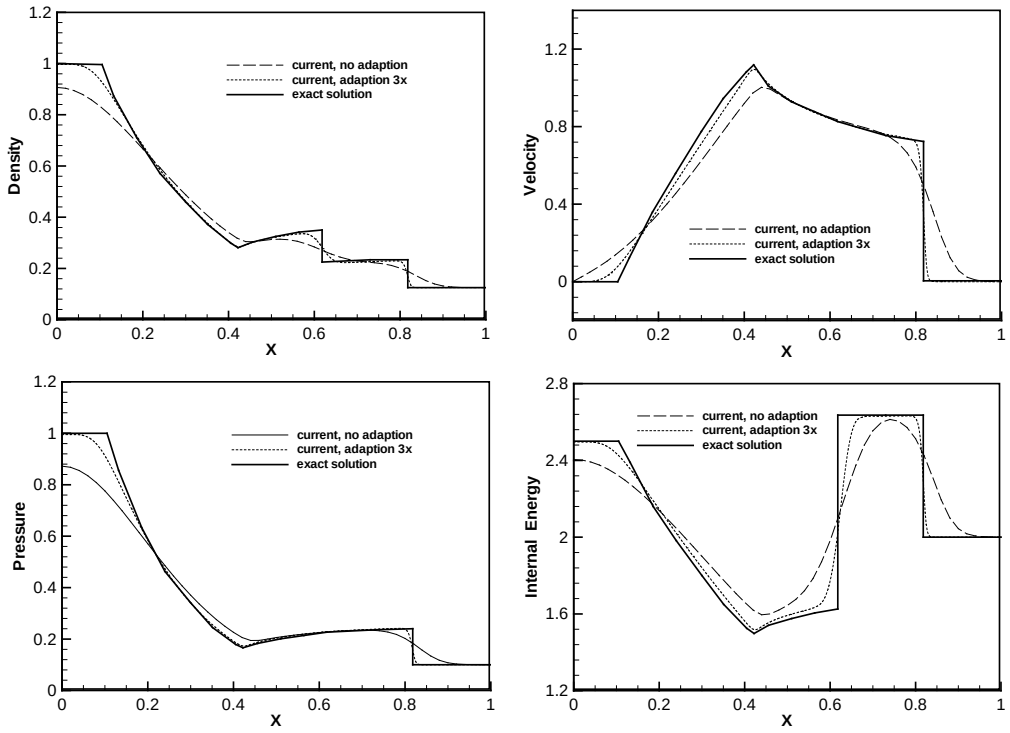
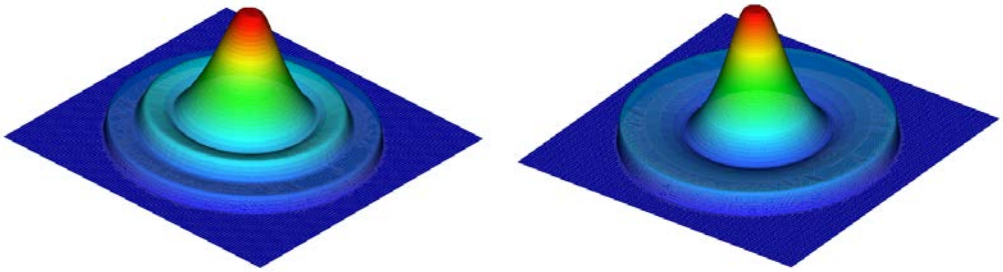


Figure 2.21 Cylindrical explosion: comparison of predicted solution between uniform mesh and 3-level adaption mesh



(a) Density distribution (b) Pressure distribution

Figure 2.22 Cylindrical explosion: density and pressure distribution at time t=0.25

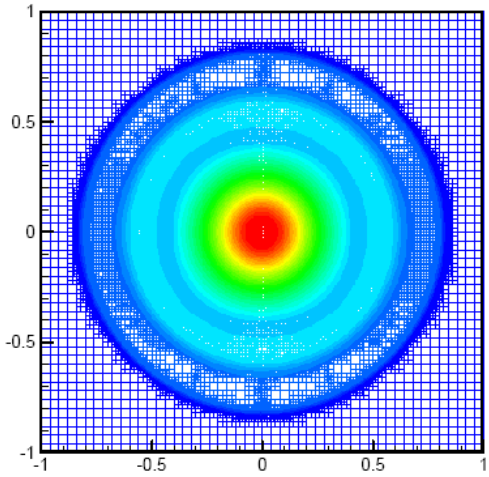


Figure 2.23 Cylindrical explosion: mesh distribution (colored by density) with 3-level adaption on initial mesh of 50×50

2.7 Effectiveness of the adaptive solver

The major advantages of the adaptive solver are its effectiveness and accuracy improvement in critical regions. By adapting finer meshes to the regions that bear high flow gradients, the adaptive solver is able to resolve the significant changes of the flow physics. In addition, considerable amount of computing time can be saved since there is no need to apply fine meshes in the regions with low flow gradients. Table 2.1 lists the computation time for the above six test cases. The comparison is made between the computations on uniform finer meshes and adaptive meshes with the same finest mesh resolution in the computational domain.

Test Case	Initial Mesh Size and CPU Time		Uniform Finer Mesh			Adaptive Mesh		
	dh	Time (s)	h/2	h/4	h/8	1x, h/2	2x, h/4	3x, h/8
1, Sod's Shock Tube	1/100	5	13	101	888	9	53	340
2, Lax Shock Tube	1/100	3	21	175	1516	12	67	446
3, Oblique Shock	1/20	8	72	658	5614	35	216	1109
4, Double Mach Reflection	1/30	7	67	659	5655	41	235	1555
5, Backward Step	1/50	3	26	261	2275	15	96	714
6, Cylindrical Explosion	1/25	1	8	76	654	6	40	297

Table 2.1 Computational time for uniform mesh and adaptive mesh

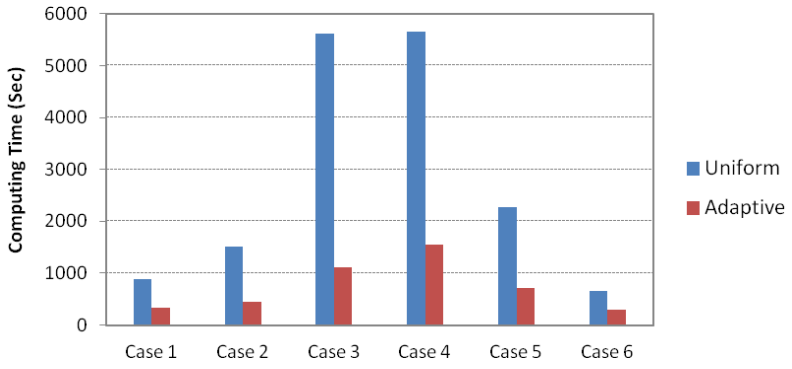


Figure 2.24 Computational time for finest meshes for six test case

From the table, it is observed that for the finest mesh (resolution of h/8) to all the six cases, the computation time required for the uniform mesh is about 2~5 times more than that required for the adaptive mesh, as shown in Figure 2.24. For the mesh resolution of h/4, this drops to 2~3; and again drops to 1.3~2 for mesh resolution of h/2. The comparison indicates that the adaptive solver is more efficient for higher adaptive level. Hence, to obtain high accuracy and save computation time, a relative coarse initial uniform mesh coupled with a higher adaptive level will be more efficient and appropriate.

Moreover, it is also noticed that the performance of the adaptive mesh varies with the problem studied. For example, the oblique shock problem demonstrates very good performance of the adaptive mesh for about 5 times faster on the resolution of h/8. It is only about 2 times faster for the cylindrical explosion problem. The flow structures indicate that the adaption is only required for two straight shocks for the oblique shock problem, but it is demanded for the three circular regions of shock, contact wave and rarefaction for the cylindrical explosion problem. And normally the rarefaction region is relative large in the domain. So larger regions need to be adapted for the cylindrical explosion problem, hence the performance will be relatively low.

2.8 Improvement of solution accuracy by the second-order schemes

The accuracy of the second-order schemes is studied based on three test cases: 1) Sod's shock tube problem, 2) two-dimensional oblique shock wave, and 3) double Mach reflection.

Although the Sod's shock tube problem is a relative simple test case, the presence of right shock wave, right travelling contact wave and a left sonic rarefaction wave make the problem a very suitable validation test case for accuracy study for a compressible CFD solver. In order to make fair comparison, the computational domain is fixed in 1×0.1 and the mesh size is fixed to 100×10. The second-order scheme is implemented with five flux limiter functions shown in the previous section. The solution for all the five flux limiter functions is obtained and plotted in Figure 2.25, including the profile for density, velocity, pressure and internal energy. The exact solution and the first order solution are also plotted on the same chart for comparison. From the comparison made in these plots, it is observed that the solution accuracy is improved obviously by the second-order scheme in the regions near the contact wave and shock wave. All the five flux limiter functions produce similar solution for density, velocity and pressure, except for the internal energy where certain level of oscillation is noticed near the contact wave position. By comparing the local oscillation of internal energy term, the flux limiter functions of Minmod and Sweby are able to produce more stable solution than the other three flux limiter functions of Osher, Van Leer and Van Albada. This verifies that using the flux limiter function does improve the solution accuracy, while its effect makes the solver very sensitive in the regions near the shock wave.

To further study the impact of the flux limiter functions on the solution accuracy, three different values of β in the Osher limiter are tested and compared: 1.1, 1.5 and 1.9, in the range of 1 to 2. The density and energy

profiles are plotted in Figure 2.26 with comparison to the exact solution. The density profile almost maintains with different β values defined in the limiter function. However, the internal energy profile shows obvious difference in the region nears the contact wave, where energy oscillation is observed as shown in Figure 2.25. The level of oscillation becomes stronger with larger β value used in the Osher flux limiter function. This indicates that the prediction of contact wave is very sensitive to the flux limiter function, and the proper choosing of the β value in the function as well.

Moreover, the two-dimensional oblique shock wave problem is a suitable 2D test case to validate the solver. To carry out the study of accuracy improvement for the second-order scheme, the computational domain is fixed to 4×1 and the mesh size is fixed to 160×40. The density contour is plotted to demonstrate the improvement of the accuracy by using the second-order solver. As shown in Figure 2.27, the two oblique shock waves can be captured much sharply by the second-order solver as compared to the solution obtained by the first order solver. The density profile at y=0.5 plotted in Figure 2.28 shows that the second-order solver using all five flux limiter functions is able to produce similar and more accurate solution as compared to the first order solver.

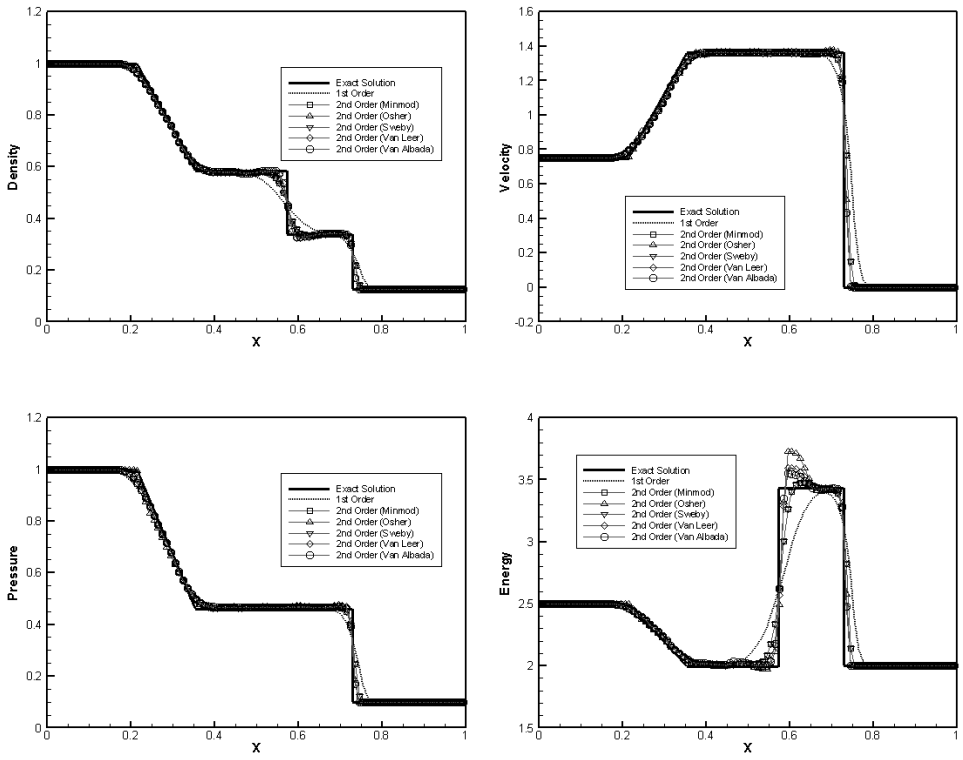


Figure 2.25 Accuracy study for the Sod's shock tube case

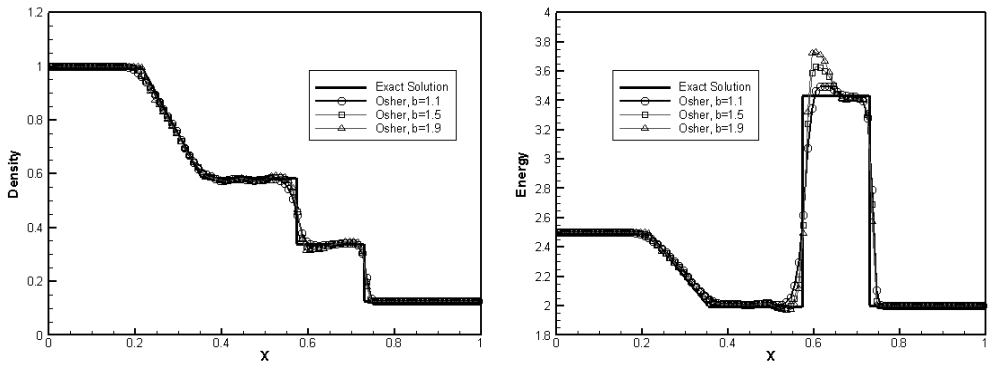


Figure 2.26 Accuracy study for different β values in Osher limiter

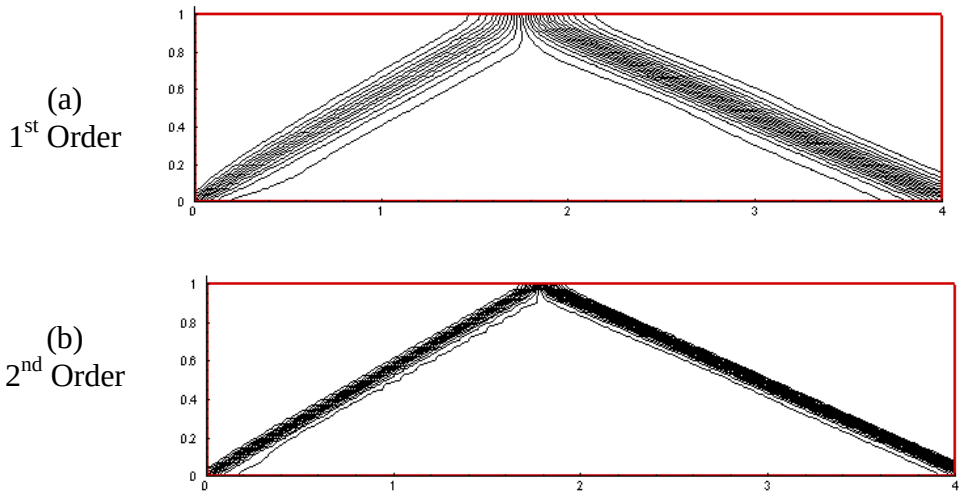


Figure 2.27 Accuracy study for oblique shock wave case (density contours)

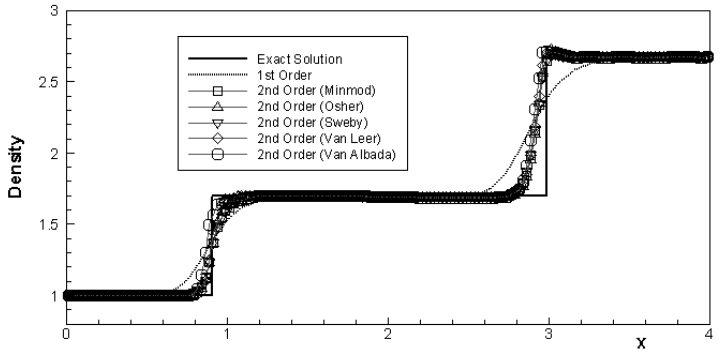


Figure 2.28 Accuracy study for oblique shock wave case (density profile)

The presence of extremely high pressure ratio of 116.5 and strong shock for the double Mach reflection problem makes it a challenging test case for compressible flow solver. In particular, when the second-order scheme or other high order schemes are used to calculate the flux on the interface of the cells, solution stability will become a common problem. Large amount of works have been done on implementation and fine-tuning of the flux limiter function in dealing with the second-order and high order schemes [3][5][42] and many of the tunings were done at the trial and error basis. In the current work, the solution for this test case was successfully obtained using all the flux

limiter functions except the Van Albada limiter. The computational domain is fixed to 4×1 and the mesh size is fixed to 400×100. The density contour is plotted and compared between the solution of the first order scheme and the second-order scheme, as shown in Figure 2.29. From the comparison, it is observed that the second-order solver is able to capture the triangle shock structure behind the moving shock sharply and accurately. This presents the benefit of accuracy improvement by using the second-order scheme.

(a)
1st Order

(b)
2nd Order

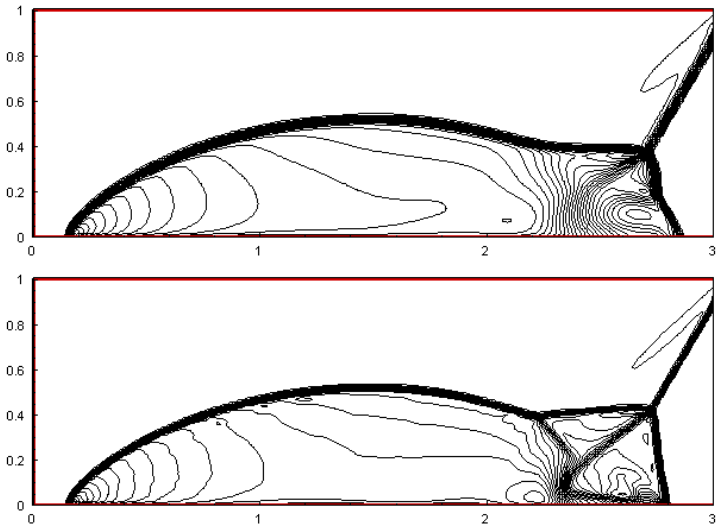


Figure 2.29 Accuracy study for double Mach reflection case (density contours)

2.9 Accuracy and convergence analysis

The numerical solution at four different meshes is computed for the Sod's shock tube problem for accuracy analysis of the current Euler solver using both first-order scheme and second-order scheme. The mesh spacing used for the analysis is 0.1, 0.05, 0.025 and 0.0125. The numerical error of density ρ is quantified using L_2 norm on all the cells in the computational domain.

$$L_2\ Error = \sqrt{\frac{\sum_N \left(\rho_{numerical} - \rho_{exact}\right)^2}{N}}. \tag{2.23}$$

The subscript (numerical) and (exact) in the formula denote the numerical density and the exact solution. The L_2 norm error versus the four different mesh spacing in the log scale is plotted in Figure 2.30. As the slope of the lines show, the accuracy of numerical results is closed to 1 and 2 for the first-order and second-order scheme, respectively. This implies that the overall accuracy of the adaptive solver can achieve second-order accuracy using the implemented second-order scheme with limiter function.

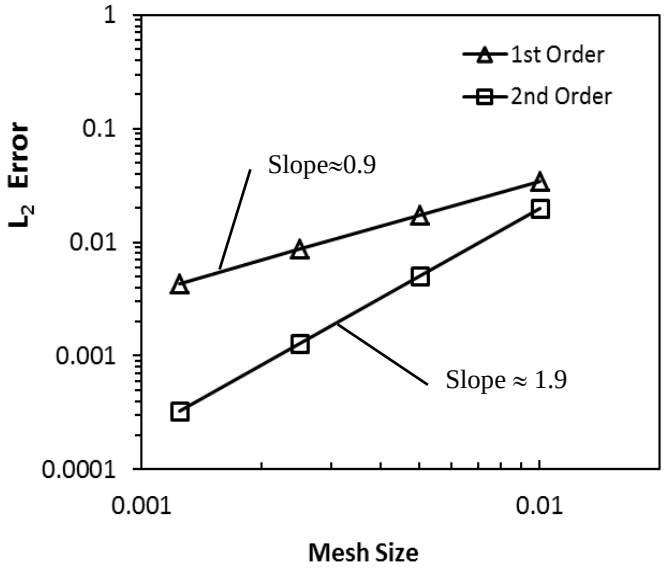


Figure 2.30 Accuracy analysis of the current solver

The oblique shock wave problem is chosen for the convergence analysis of the current solver, as it is a steady-state problem. Numerical solution is computed on uniform coarse mesh 80×20 and uniform fine mesh 320×80. To benchmark the convergence characteristics of the adaptive solver, numerical solution is also computed with 2-level solution adaption through two different approaches.

One approach (case [a]) is to enable the solution adaption from the beginning of iteration and the other approach (case [b]) is to start the solution adaption only after the solution is converged on coarse mesh. In the present study, steady-state solution is considered converged when the maximal residual of the solution is reduced below 10^{-6}.

The four different convergence tracks are plotted in Figure 2.31 as the residual of density versus the number of iteration of the solver. As it shows the uniform coarse mesh solution converges the fastest in only 1800 iterations using about 2 seconds, while the uniform fine mesh solution takes about 6400 iterations and 135 seconds to converge. Usually, large time step can be used on coarse mesh as compared to finer mesh. Hence it takes fewer iterations of solution evolution in marching to the steady-state solution on coarse mesh. Apart from this, fine mesh implies more equations to be solved on the larger number of mesh cells. As a result, the computing time required for fine mesh increases significantly instead of linearly. The comparison of the convergence history between two solution adaption approaches shows no much difference. Both approaches take about 6500 iterations and 34~40 seconds to converge, with the approach [b] takes slightly less computing time to converge. This could be due to the fact that converged coarse mesh solution provides a better “initial condition” for the 2-level adaption solution, while it does not take too long to achieve the converged coarse mesh solution. This could be a useful approach for steady-state flow simulations.

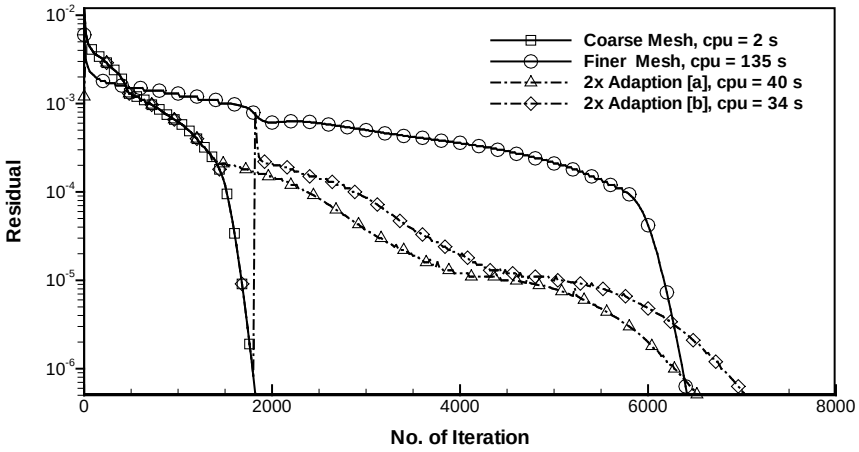


Figure 2.31 Convergence analysis of the current solver

[a], solution adaption is enabled from the beginning of iteration.
[b], solution adaption is enabled only after the coarse mesh solution is converged.

2.10 Conclusions

In this Chapter, an adaptive Euler solver was developed based on finite-volume method. The HLLC scheme is used to compute the flux on cell interface and five second-order flux schemes were implemented and validated. The AMR technique is an important feature implemented on the current Euler solver to improve the accuracy and efficiency of the solver.

The current adaptive Euler solver was validated through six 2D test cases, with the adaptive solutions were obtained with 3-level solution adaption. The present results are compared with the analytical solutions and numerical results in the literation. The comparisons show good agreement. The solution adaptive capability demonstrates remarkable performance not only in the solution accuracy improvement but also in the improvement of solver

efficiency. Five second-order schemes were implemented and their performance for solution accuracy improvement was studied and presented. The accuracy and convergence analyses not only show that the current adaptive Euler solver can achieve close to second-order accuracy, but also demonstrate that the solution can converge efficiently using the solution adaption feature developed in the current solver.

The current adaptive Euler solver has built a concrete foundation for the implementation and validation of the immersed boundary methods for compressible flows.

Chapter 3 Ghost-cell Method-based Adaptive Euler Solver

The Cartesian grid-based adaptive solver presented in Chapter 2 demonstrates good capability and effectiveness in simulating compressible inviscid flows. In order to use the adaptive solver for various generic compressible flow problems, the solver needs to be extended to handle irregular wall boundary surfaces effectively. Such kind of wall boundary surfaces can be a domain boundary surfaces or wall boundary surfaces of immersed solid bodies in the computational domain.

As discussed in the literature review in Chapter 1, there are three methods to represent wall boundaries on non-body-fitted Cartesian grids. Among the three methods, ghost-cell method is relatively less complicated to be implemented on Cartesian grids as the flux calculation is only performed on the mesh cell interfaces. There is no need to form special cut-cells or merge small cut-cells into neighbor parent cells as required in cut-cell method; and neither does it need to develop a grid-less algorithm to solve the governing equations in the grid-less zone as required for grid-less method. Instead, in the ghost-cell method, one only needs to accurately correct flow information near the wall boundary according to the prescribed boundary conditions.

Therefore, the ghost-cell method is adopted and implemented in the current adaptive Euler solver. As the method has been implemented and tested by

many researchers, the implementation of the method in the current adaptive Euler solver enables the validation and performance study for compressible flows involving complex boundaries can be made and compared.

3.1 Ghost-cell method

The ghost-cell method was first proposed by Forrer and Jeltsch [5] and then improved by the curvature-correction symmetry technique (CCST) introduced by Dadone and Grossman [6], [7], [8]. The method is an enhancement to the symmetry technique to take into account the wall curvature for solid wall boundary conditions based on body-fitted grid. The implementation of CCST gives the approximation of pressure, density and normal velocity on two image/ghost cell centers, shown as hollow dots (-1) and (-2) in Figure 3.1.

For cell center (-1):

$$p_{-1} = p_1 - \rho_W \frac{u_s^2}{R_W} \Delta n_1, \tag{3.1}$$

$$\rho_{-1} = \rho_1 \left(\frac{p_{-1}}{p_1} \right)^{1/\gamma}, \tag{3.2}$$

$$\tilde{u}_{-1}^2 = \tilde{u}_1^2 + \frac{2\gamma}{\gamma - 1} \left(\frac{p_1}{\rho_1} - \frac{p_{-1}}{\rho_{-1}} \right), \tag{3.3}$$

$$\tilde{v}_{-1} = -\tilde{v}_1. \tag{3.4}$$

For cell center (-2):

$$p_{-2} = p_2 - \rho_W \frac{u_s^2}{R_W} \Delta n_2, \tag{3.5}$$

$$\rho_{-2} = \rho_2 \left(\frac{p_{-2}}{p_2} \right)^{1/\gamma}, \tag{3.6}$$

$$\tilde{u}_{-2}^2 = \tilde{u}_2^2 + \frac{2\gamma}{\gamma - 1} \left(\frac{p_2}{\rho_2} - \frac{p_{-2}}{\rho_{-2}} \right), \tag{3.7}$$

$$\tilde{v}_{-2} = -\tilde{v}_2. \tag{3.8}$$

Here Δn_1 and Δn_2 are the distances between cell centers $(+1) \to (-1)$ and $(+2) \to (-2)$. The symbol ~ denotes the normal-tangent velocity components on the wall boundary. The tangent velocity on cell centers (-1) and (-2) is the same as it is on cell centers (+1) and (+2).

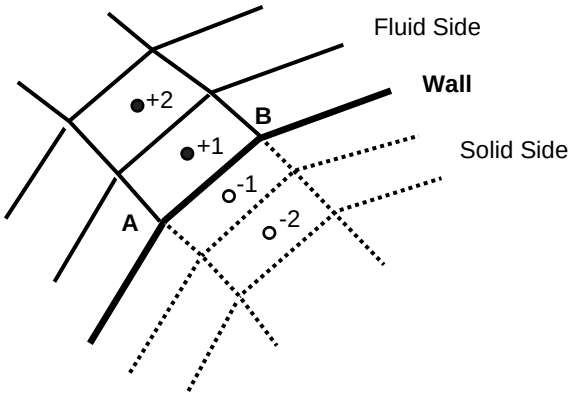


Figure 3.1 Concept of ghost-cell method

The CCST method can be adopted and extended to apply on the Cartesian grid. To make it flexible and simple to implement, the ghost-cells in the solid zone are searched in the local region within double mesh size from the boundary. For convenience, the cells near the wall boundary are named shadow-cells in this thesis. Moreover, the shadow-cells fallen in fluid zone are labeled as fluid shadow-cells, and those fallen in solid zone are named as solid shadow-cells.

In this context, as shown in Figure 3.2, the ghost-cells around wall point-B can be classified into fluid shadow-cell (in green) on one side of the boundary, and solid shadow-cell (in gray) on the other side of the boundary. When enforcing wall boundary conditions, the flow information on the solid shadow-cells are updated using equations (3.1) - (3.4) . Governing equations are solved on the fluid shadow-cells directly, so there is no additional action needed for those cells.

However, as the fluid shadow-cells and the solid shadow-cells under this case may not be right in symmetrical relationship across the wall surface, the pressure, density and normal velocity on the solid shadow-cells cannot be updated directly using equations (3.1) - (3.4) . Hence the symmetrical relationship needs to be built first. This can be done by mirroring either solid shadow-cells into the fluid region, or mirroring fluid shadow-cells into the solid region. As the flow information on solid shadow-cells need to be updated based on the boundary condition and local fluid information, it will be easier to mirror the solid shadow-cells and update the flow information on them directly. To illustrate the process, as shown in Figure 3.2, for the solid shadow-cell (j), if $\vec{jB}$ is normal to wall surface, the cell center (j) can be reflected based on normal direction $\vec{n}$ to (j') into the fluid region. The flow information on position (j') can be obtained using local interpolation with all the fluid shadow-cells of wall point-B. In current study, the simple inverse distance method is used for the interpolation. Once this is done, using the wall boundary condition equations (3.1) - (3.4) , the pressure, density and velocity can be computed and updated for the solid shadow-cell (j). The total energy

will be updated using the equation of state.

As mentioned above, the boundary condition is implemented through updating the flow information on solid shadow-cells from the fluid shadow-cells. Therefore, a quick and universal way to identify whether shadow-cells fall in fluid region or solid region is very crucial.

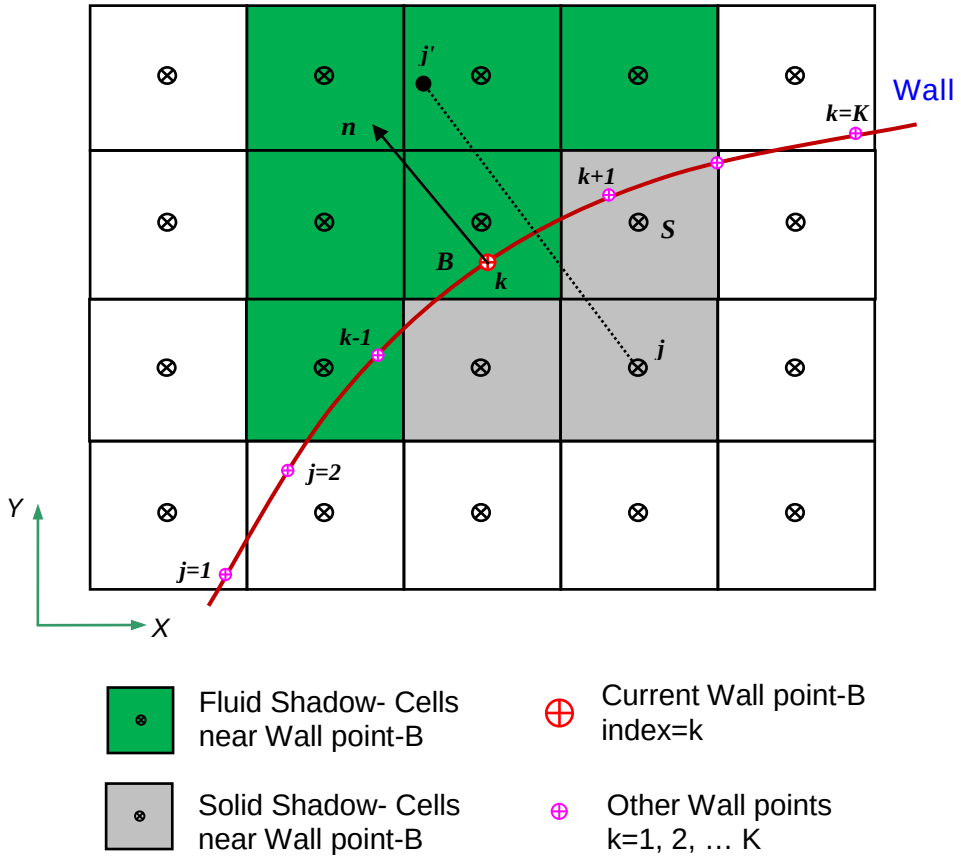
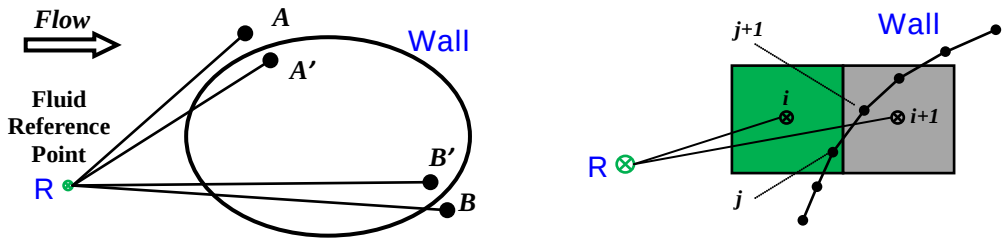


Figure 3.2 Implementation of CCST method on Cartesian grid

It can be found that boundary of a 2D immersed body is normally represented by many short line segments. Imagining walk along these segments from the head to the tail, one can always find that the fluid shadow-cells are always on one side and the solid shadow-cells are on the other side. Inspired by this, an

innovative approach is proposed by introducing a fluid reference point ***R*** in the computational domain and then using the relationship between the line segment of cell center to the fluid reference point and the wall edge to determine the property of the shadow-cell. As shown in Figure 3.3 (a), point A is located in the fluid region as $\overrightarrow{RA}$ does not intersect with the Wall; points A' and B' are in the solid region as $\overrightarrow{RA}'$ and $\overrightarrow{RB}'$ intersect with the Wall once; point B is in the fluid region again as $\overrightarrow{RB}$ intersects with the Wall twice. This is like the situation when walking from the fluid reference point R to the point B, it first passes the wall and goes inside the solid body and then passes the wall again to enter into the fluid region at point B. In general, a point (i) will be in fluid region if $\overrightarrow{Ri}$ intersects with the Wall by an even number of times; and in solid region if by an odd number of times.

Figure 3.3 (b) illustrates the local view and relations of the shadow-cells (***i***) and (***i+1***), the wall segments (***j***)-(***j+1***), and the fluid reference point ***R***. Because $\overrightarrow{R \to i}$ has no intersection point with all wall segments and $\overrightarrow{R \to (i+1)}$ intersects with the wall segment (***j***)-(***j+1***), so cell (***i***) is a fluid shadow-cell and cell (***i+1***) is a solid shadow-cell.



(a) Overview sketch **(b) Close view near the shadow-cells**

Figure 3.3 Concept in determining the property of shadow-cells

3.2 Results and discussion

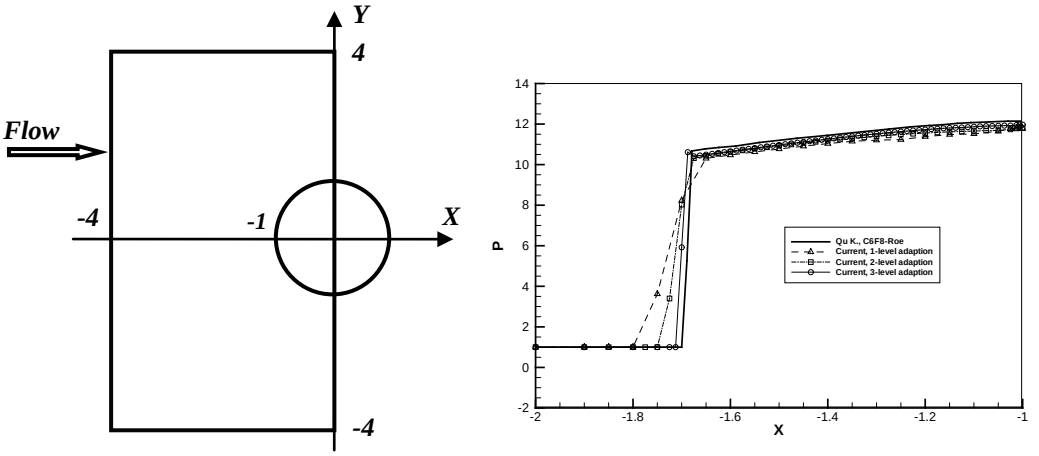
To validate the implementation of ghost-cell method in current adaptive Euler solver, two well-known test cases are experimented numerically. The first test case is a supersonic inviscid flow over a circular cylinder at Mach number 3; the second test case is a transonic flow problem over a channel with a 10% circular bump at the bottom of the channel. The transonic flow over a RAE2822 airfoil and supersonic flow over 3 disks are computed to further demonstrate the capability and robustness of the current adaptive solver.

3.2.1 Supersonic flow over a circular cylinder

Supersonic flow over two-dimensional circular cylinder was studied extensively by many researchers [43], [44]. Here the supersonic flow with inflow of Mach 3 is simulated. The computational domain is an 8×4 rectangle domain on X-Y coordinate system, with a unit cylinder centered at the origin. Half of the cylinder is model as the flow structure upstream of the cylinder is more interested. The sketch of the computational domain and the location of the cylinder are shown in Figure 3.4 (a). The left of the domain is defined as Mach 3 inlet; the other three boundaries are defined as outlet boundary with zero flow gradients. Because the flows at the three outlets are in supersonic flow condition, using zero gradient extrapolation at the outlets is valid. An initial uniform mesh of 40×80 is defined in the rectangle domain. The half cylinder wall is expressed by 315 points with a resolution of about 0.01.

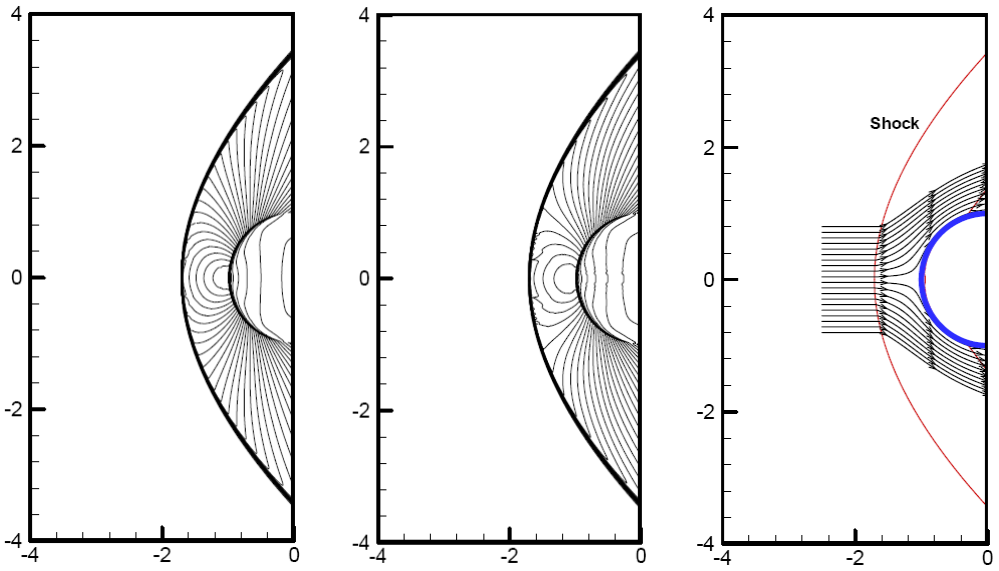
The computed pressure profile along the central line is plotted in Figure 3.4

(b). The predicted shock becomes shaper and the position of the shock is very close to the results obtained by Qu [43] using the Lattice Boltzmann model and Visbal and Gaitonde [44] on a body-fitted grid. The pressure after the shock predicted is about 11.97 on the nose of the cylinder, and matches well to their prediction of 12.10. The pressure and density contours are plotted in Figure 3.5 (a) and (b). The contours indicate that the shock position and the unit circular cylinder outline are computed clearly and sharply. The streamlines shown in Figure 3.5 (c) demonstrate that the air flow passes the circular cylinder very smoothly and with no streamlines penetrating the cylinder surface. This implies that the no-penetration wall boundary condition is satisfied accurately.



(a) Sketch of the Problem (b) Pressure profile along the central line

Figure 3.4 Mach 3 supersonic flow over a circular cylinder



(a) Pressure Contours (b) Density Contours (c) Streamlines

Figure 3.5 Results for Mach 3 supersonic flow over a circular cylinder

3.2.2 Transonic flow over a channel with bump

This problem is the well-known Ni's steady-state test case: a transonic flow in a channel with a 10% thick circular bump on the bottom (or known as GAMM channel). The computational domain is a rectangle of length 3 and height 1. The inlet is at left with a Mach number 0.675, see Figure 3.6. A shock is formed downstream the throat near the bottom wall. The initial mesh is defined as 60×20. Local meshes near the circular bump are pre-adapted by four levels to ensure that the circular bump wall boundary is represented with sufficient Cartesian cells.

The computed Mach number and pressure contours by 2-level adaption based on density gradient in the flow field are presented in Figure 3.7 (a) and (b). The Mach number distribution on the lower wall of the channel computed is

compared with those obtained using unstructured finite volume scheme by Luo et al. [45] and hybrid grid-less method by Luo et al. [11]. The computed result obtained via the current method matches well to those results. As shown in the Figure 3.7 (c) for the Mach number distribution on the lower wall of the channel, the maximal Mach number matches very well, while the shock position is slightly further away downstream to the throat compared to other results.

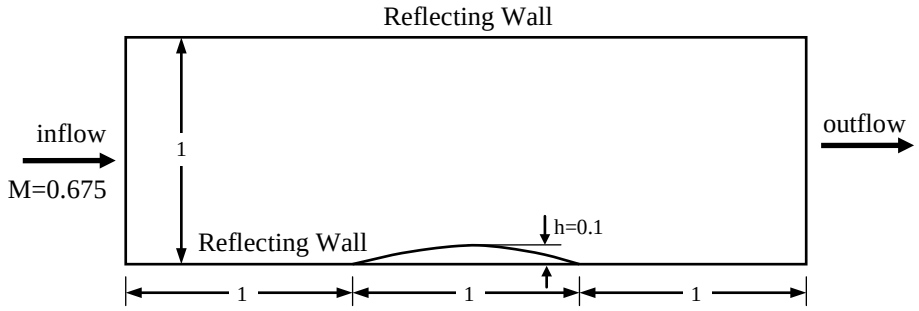
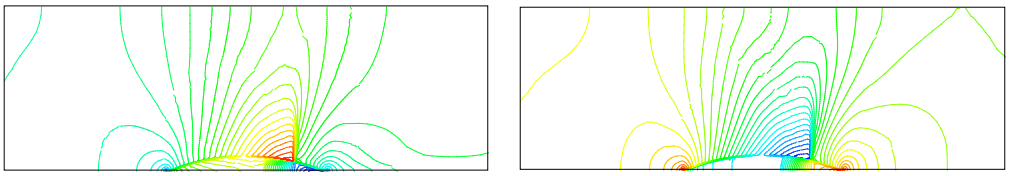
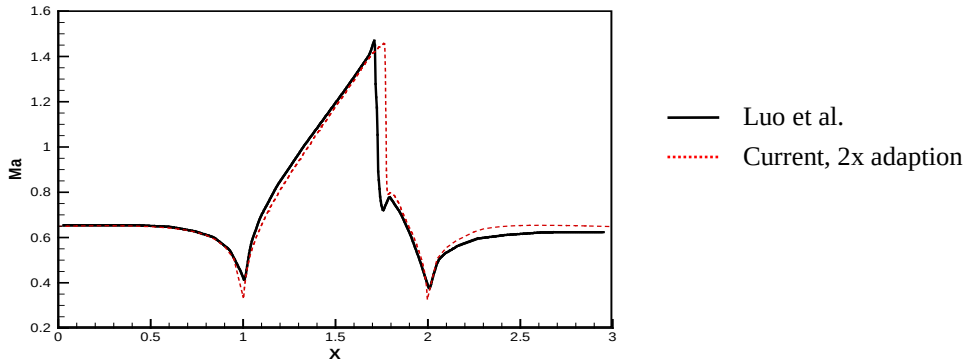


Figure 3.6 Transonic flow in GAMM channel with a 10% circular bump



(a) Mach number contours (b) Pressure contours



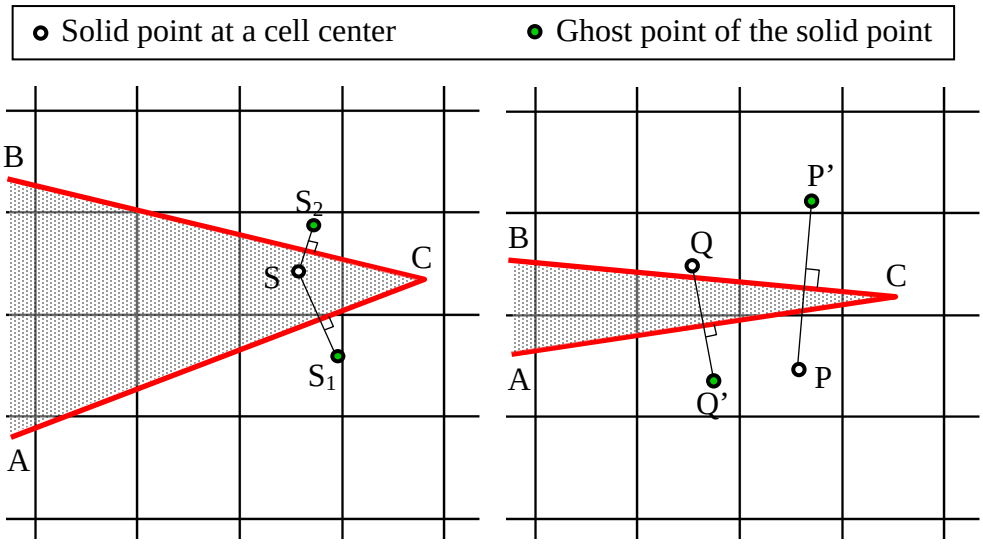
(c) Mach number on the lower wall of the channel

Figure 3.7 Computed Mach number and pressure in GAMM channel

3.2.3 Transonic flow over a RAE2822 airfoil

Transonic flow over RAE2822 airfoil is simulated under the free stream flow condition at Mach = 0.729, and the angle of attack (AoA) at 2.31°. The computational domain is chosen at about 20 times of the chord size away from the airfoil. The coarse uniform Cartesian mesh at size of $1/5$ is used as the background mesh, and the airfoil geometry is adapted by 6-level refinement or fine mesh size at $1/320$. The solution is adapted by 4-level refinement or mesh size at 1/80.

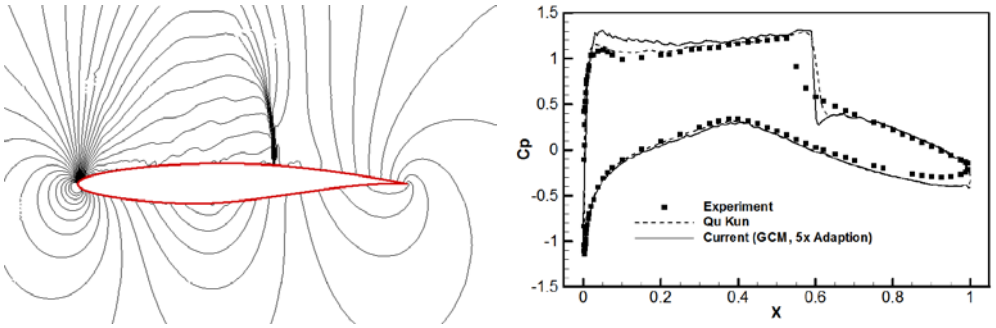
As illustrated in Figure 3.2, the mirror point of the solid point or solid shadow-cell is needed to compute against the wall boundary for each "solid point" at a cell center. For solid body with thin or sharp boundary shape such as airfoil's trailing edge region, there are two special cases where a solid point is possible to have two mirror points or a "fluid point" may become a "solid point" near the boundary. Figure 3.8 (a) shows the case where the solid point (S) has one mirror point (S_1) against wall boundary $\overline{AC}$ and another mirror point (S_1) against wall boundary $\overline{BC}$; and Figure 3.8 (b) shows that the fluid points (P, Q) become "solid point" for the wall boundary $\overline{BC}$ and $\overline{AC}$, respectively. For those special cells and "solid points", two sets of flow values are computed and stored on the solid points (S, P, and Q) according to the corresponding boundary. The flux calculation on the cell interfaces of those cells must use the corresponding values separately. It is noted that the fluid points (P and Q) are in the interior fluid domain and their actual flow values are computed by solving the governing equations.



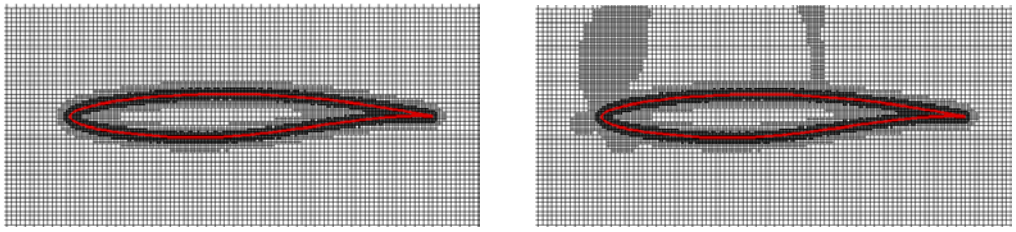
(a) A solid point has 2 ghost points (b) A fluid point may become a "solid" point boundary

Figure 3.8 Special cases for ghost-cell method implementation

The computed results for the transonic flow over a RAE 2822 airfoil are presented in Figure 3.9 by pressure contours, pressure coefficient profile on the airfoil surface, and the initial meshes adapted to the airfoil boundary and the final meshes when the solver is converged. The computed pressure coefficient profile (solid line) is compared with the experimental data (solid dot) and the numerical result obtained on C-type body-fitted grid (dash line) by Qu [43]. The current result is comparable to the numerical result obtained by Qu [43] and Tullio et al. [64]. The shock wave position matches exactly. The shock wave predicted on the upper surface of the airfoil is slightly different from the experimental data because the flow in the experiment is a turbulent flow in which the shock wave interacts with the boundary layer. Therefore, its stiffness and location are different from the numerical simulation.



(a) Pressure contours (b) Pressure coefficient profile



(c) Initial mesh adapted to airfoil (d) Final mesh adapted to solution

Figure 3.9 Computed results for flow over a RAE2822 airfoil (M=0.729, AoA=2.31°)

3.2.4 Mach 3 flow over three disks

This example shows that the method can be applied to problems with multiple immersed solid bodies. The wall boundary surfaces that form the solid bodies are treated separately, and the fluid shadow-cells and solid shadow-cells can be identified according to each wall boundary surface. The problem is a 2D supersonic flow (with $M_\infty = 3$) past three disks. The three disks are centered at (-1.3, -1.0), (-1.0, 0.8) and (1.3, 0.2), with radii of 0.3, 0.3, and 0.4, respectively. The computational domain is 8×8 with initial uniform mesh of 50×50, see Figure 3.10. The computed density contours and the final solution adapted meshes are plotted in Figure 3.11. The predicted shock waves agree with the numerical result obtained by Sjögreen and Petersson [46]. From the

solution adaptive mesh distribution in Figure 3.11, it is confirmed again that the current adaptive solver is very efficient in capturing shock waves for compressible flows. To obtain a fine mesh solution at size of $1/40$, it requires to solve the problem on 102,400 cells. In comparison, it only requires 36,600 cells approximately if 3-level adaption is used in the current adaptive solver. The computing time for the adaptive solution is only about $1/30$ of that required for the same resolution uniform mesh. The results of this problem demonstrate the robustness and efficiency of the current adaptive ghost-cell solver to handle compressible flow problems with multiple solid bodies.

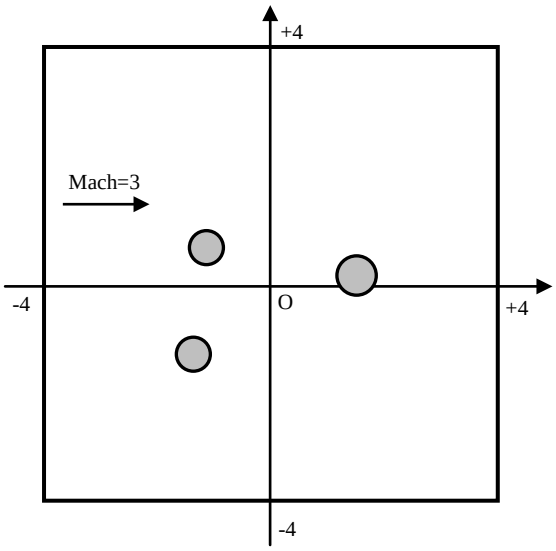


Figure 3.10 Computational domain for Mach 3 flow over 3 disks

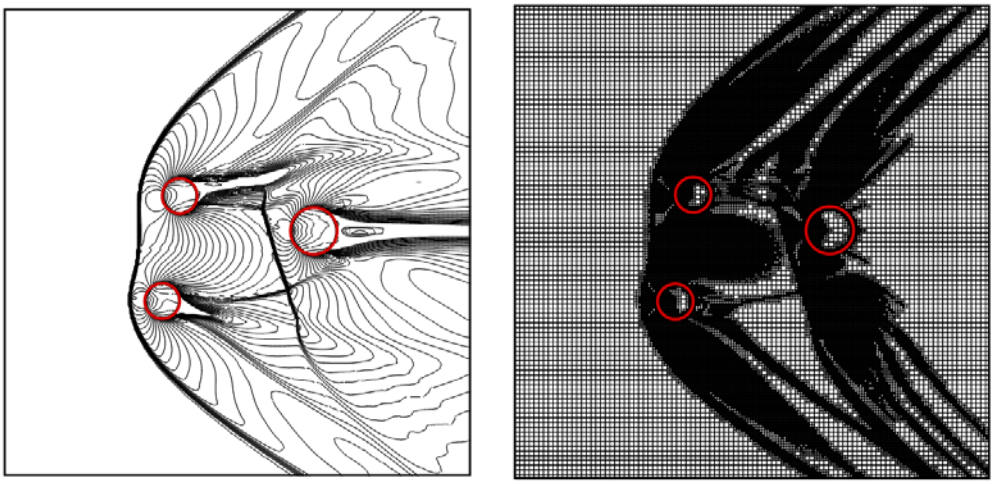


Figure 3.11 Computed density contours and solution adaptive mesh for Mach 3 flow over 3 disks

3.3 Conclusions

The ghost-cell method is implemented with the current adaptive Euler solver to simulate compressible flows with irregular wall boundaries in this Chapter. The implementation of ghost-cell method is relatively simple compared to cut-cell method and grid-less method. In the ghost-cell method, shadow cells near the wall boundary are identified in the region near boundary within two mesh spacings. Thereafter, the shadow cells are further classified as the fluid shadow-cells and the solid shadow-cells through the ray tracing method from a known reference fluid point. With the fluid/solid shallow-cells identified, the wall boundary condition is then enforced by correcting the flow information on the solid shadow-cells. To perform the correction, the mirror point of the center of a solid shadow-cell is computed against with the boundary in the first place; next the flow information at the mirror point is interpolated using the

flow data at the relevant fluid shadow-cells; finally the flow information is reflected back to the solid shadow-cell through the boundary condition relationship.

The developed adaptive ghost-cell solver is validated by a supersonic flow over a circular cylinder and a transonic flow over a channel with bump. The computed results are in good agreement with the numerical results available in the literature. Moreover, transonic flow over a RAE2822 airfoil and Mach 3 supersonic flow over 3 disks are also simulated. The results demonstrate that with special treatment for the solid shadow-cells near the thin sharp boundary surfaces, the method is able to simulate compressible inviscid flows over 2D airfoil accurately. Furthermore, simulation of compressible flow with multiple wall boundaries was performed and promising results was obtained.

In summary, through the numerical experiments tested in this chapter, it can be seen that the ghost-cell method can provide accurate results for various problems. However, in the implementation of ghost-cell method, there is a need to identify whether the shadow-cells are in fluid zone or solid zone and to compute the mirror point against the boundary. Moreover, the boundary curvature has to be computed in order to enforce the boundary condition. Because of those requirements, the ghost-cell method is still complicated and limited to be applied for many other compressible flow problems.

Chapter 4 Flux Correction-Based Immersed Boundary Solver

In the ghost-cell method implemented in the previous chapter, the wall boundary condition is enforced by satisfying the no-penetration condition for wall. The relationship between the pressure gradient and the flow curvature is governed by the streamline curvature theorem and constant entropy near the wall interface. The method is demonstrated to be able to produce quite accurate numerical solutions. In order to satisfy the relationship between the local pressure gradient and the flow curvature, the wall boundary curvature is required to be calculated. The solution accuracy will be affected by the calculated curvature which sometimes can be less accurate. In addition, it is mandatory to know whether a shadow cell is located in the solid domain or in the fluid domain during the implementation, and also to find out the mirror/ghost point against the wall boundary interface.

The requirement of calculating the curvature of the wall boundary and determining the location of the shadow cells and their mirror position makes the implementation tedious and less convenient, besides the possible accuracy impact due to the inaccurate calculation of the wall boundary curvature. The ideal situation is that the immersed wall boundary condition can be satisfied without the need of calculation of wall curvature and advance determination of the location of the shadow cells. To achieve this objective, new approaches shall be introduced to implement the wall boundary condition.

There are four conservative variables in 2D Euler equation and five original flow variables. The equation system is closed with the equation of state for air. To implement the wall boundary condition, four variables have to be updated according to the role or the effect of the boundary and the last variable can be simply determined by the equation of state. The wall boundary condition is treated as reflection wall in most Euler compressible flow solvers. The commonly adopted method in enforcing the boundary condition is through the manipulation of velocity field (or the two velocity components for 2D flows), pressure, density and temperature/energy at the cells opposite the boundary, which is the same as that used for body-fitted solver and the ghost-cell method presented in Chapter 3.

As discussed, this approach is not simple and tedious to be implemented in Cartesian grid solver. It will be simple and efficient if the boundary condition can be enforced in the similar manner as that in the IBM approaches that have been successfully implemented and applied for incompressible viscous flows. In those IBM approaches, the implementation is carried out by correcting the flow field on the cells in the vicinity of the boundary without the need to know if they are fallen in fluid domain or solid domain. The target is to satisfy the no-slip boundary condition either directly or indirectly. The no-slip boundary condition means that the velocity on the boundary is known, no matter whether the boundary is stationary or moving. For compressible inviscid flow, the boundary condition on wall becomes no-penetration which means that no fluid should flow across the boundary but fluid can flow freely along the boundary. So the velocity on the boundary is unknown and it changes

according to flow conditions. This is the challenge to adopt and implement the concept of traditional IBM approaches for incompressible flows in a compressible inviscid flow solver. Besides the no-penetration condition which only governs the velocity field, other boundary conditions that govern pressure, density and temperature for compressible fluid must be satisfied as well. This contributes more challenges for the implementation.

From the viewpoint of fluid dynamics for compressible inviscid flows, the behavior of a wall boundary is to prevent fluid from entering the wall boundary interface, which is known as the no-penetration condition. An ideal and adiabatic wall boundary also functions as an interface with zero mass flux and zero energy flux. Therefore, if no-penetration condition and zero fluxes' condition on the wall boundary can be enforced, the influence of wall boundary to the fluid domain is fully counted as the velocity, density and temperature that are related to the three conditions are satisfied. The pressure of the fluid can be determined by the equation of state.

4.1 Flux correction-based Immersed Boundary Method

In this chapter, a novel flux correction-based immersed boundary method, named in short form as FC-IBM, is proposed to enforce the wall boundary conditions on the current Cartesian grid-based adaptive Euler solver. The velocity field is corrected based on no-penetration condition.

The implicit velocity correction-based immersed boundary method, or IVC-IBM as named by the authors, was proposed by Shu and his co-workers [22],

[33] to solve the incompressible flow around a cylinder and a swimming fish. Unlike the conventional IBM where the effect of rigid body or wall boundaries on the surrounding flow is modeled through a forcing term calculated in advance and then formulated to correct for the surrounding velocity field, the authors proposed an innovative implicit velocity correction-based IBM, where the velocity is corrected via enforcing the physical boundary condition. Their results demonstrate that the IVC-IBM produces more reasonable results that can accurately satisfy the physical boundary conditions on the wall boundary surfaces for the incompressible viscos flows.

Inspiring by their work on modeling no-slip wall boundary for the incompressible viscous flows, the IVC-IBM is adopted and modified in order to model the no-penetration wall boundary condition for compressible inviscid flows. Although the velocity on the no-penetration wall is not known and varies according to the flow conditions, the no-penetration condition implies that the normal velocity on the boundary must be zero, or $V_n = 0$. However, as the normal velocity computed from the Cartesian cells may not be zero, hence the goal for velocity correction is to enforce this condition by correcting the velocity at the nearby Cartesian cells. Similarly, the zero mass flux and zero energy flux will be enforced on the boundary.

4.1.1 Velocity correction

For no-penetration wall boundary, the velocity boundary condition on the wall point-B as illustrated in Figure 4.1 is that the normal velocity should be zero, i.e. $V_n = 0$. So the purpose of the velocity correction is to satisfy this condition.

To achieve this goal, the normal velocity on the boundary needs to be computed from the nearby cells. For convenience, two-dimensional situation is considered here for illustration. Let $\langle n_x,\ n_y \rangle$ denote the normal direction on the wall point-B, and then $\langle n_y,\ -n_x \rangle$ will be the tangent direction, see Figure 4.1.

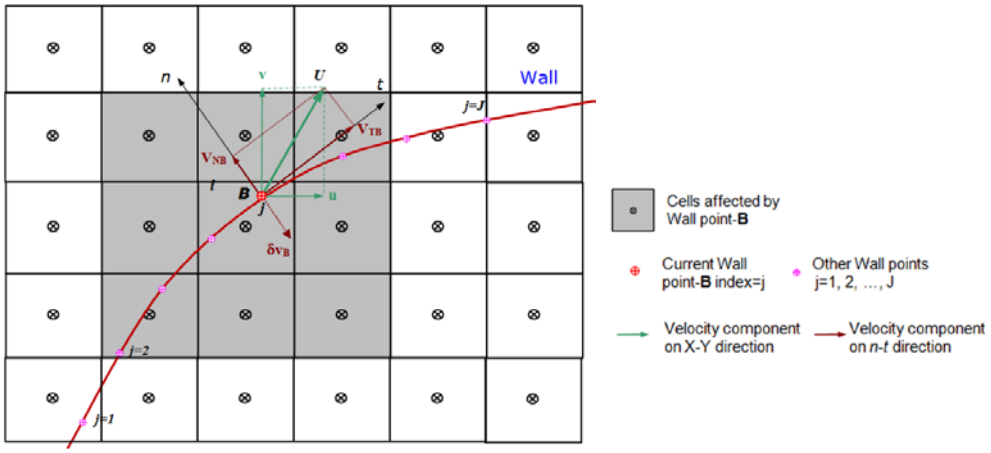


Figure 4.1 Illustration of velocity correction using IBM

To start the velocity correction, the shadow cells around the wall point-B within double mesh size in both X and Y directions are identified. The shadow cells here indicate the cells that affect and determine the flow information on wall boundary. The continuous delta function $D_{i,j}$ which was proposed by Peskin [21] is used to interpolate the velocity on the boundary from the nearby shadow cells. The delta function $D_{i,j}$ is given by

$$\delta(r)=\begin{cases}\dfrac{1}{4}\left[1+\cos\left(\pi|r|/2\right)\right], & |r|\le 2,\\ 0, & |r|>2.\end{cases} \tag{4.1}$$

$$D_{i,j}\left(x_i-X_B\right)=\delta\left[\left(x_i-x_B\right)/h\right]\times\delta\left[\left(y_i-y_B\right)/h\right]. \tag{4.2}$$

Using all the shadow cells and the delta function $D_{i,j}$, the velocity at wall point-B is estimated as

$$\begin{cases} u_B = \sum u_i \cdot D_{i,j} \\ v_B = \sum v_i \cdot D_{i,j} \end{cases}, \quad i \in \{\text{all cells marked in shadow}\} \tag{4.3}$$

Then (V_{NB}, V_{TB}) can be obtained by transforming the velocity components from X-Y coordinates to ***n-t*** coordinates on the wall point-B.

$$\begin{cases} V_{NB} = n_x u_B + n_y v_B, \\ V_{TB} = n_y u_B - n_x v_B. \end{cases} \tag{4.4}$$

The normal velocity V_{NB} obtained from the above equation may not be zero. To enforce the no-penetration condition, a negative normal velocity correction ($-V_{NB}$) is introduced to the wall point-B and the tangent velocity just maintains.

$$\begin{cases} \delta V_{TB} = 0 \\ \delta V_{NB} = -V_{NB} \end{cases} \rightarrow \quad \delta V_B = -V_{NB}. \tag{4.5}$$

This correction of velocity component needs to be distributed back into all the shadow cells hence to make the local velocity field satisfy the no-penetration condition. Let $\delta V_B = -V_{NB}$ denote the velocity correction on point-B, and then the following velocity correction shall be added to all the shadow cells nearby point-B:

$$\delta \vec{u} = \sum \delta V_B D_{i,j}. \tag{4.6}$$

To perform the above equation, the velocity correction δV_B at point-B is first transferred back to X-Y system as

$$\begin{cases} \delta u_B = n_x \cdot \delta V_B + n_y \cdot 0 \\ \delta v_B = n_y \cdot \delta V_B - n_x \cdot 0 \end{cases} \rightarrow \begin{cases} \delta u_B = -n_x \cdot V_{NB} = -n_x \cdot \left(n_x u_B + n_y v_B \right) \\ \delta v_B = -n_y \cdot V_{NB} = -n_y \cdot \left(n_x u_B + n_y v_B \right) \end{cases} \tag{4.7}$$

and then re-distributed to the shadow cells as velocity components in X and Y directions, which are used in the solver.

$$\begin{cases} \delta u_i = \delta u_{B,j} D_{i,j}, \\ \delta v_i = \delta v_{B,j} D_{i,j}. \end{cases} \tag{4.8}$$

Because a shadow cell can be shadowed by a few wall points, the overall correction at a shadow cell from all the nearby wall points will be:

$$\begin{cases} \delta u_i = \sum_j \delta u_{i,j} = \sum_j^J \left(\delta u_B \cdot D_{i,j} \right) \\ \delta v_i = \sum_j \delta v_{i,j} = \sum_j^J \left(\delta v_B \cdot D_{i,j} \right) \end{cases} \tag{4.9}$$

Finally, the corrected velocity $\left(u_i^*, v_i^* \right)$ at the shadow cells will be updated as below:

$$\begin{cases} u_i^* = u_i + \delta u_i, \\ v_i^* = v_i + \delta v_i. \end{cases} \tag{4.10}$$

Because the velocity at a shadow cell may be corrected by multiple wall points, the corrected velocity field may not fully satisfy the no-penetration boundary condition. Instead of using the implicit velocity correction method as proposed by Shu et al. [22] for incompressible flows, an explicit method is used via iterative approach for the velocity correction. The whole procedure for the velocity correction consists of the following steps:

1) For wall point-B, or index-j, calculate $\left(u_B, \ v_B \right)$ via equation (4.3) from the shadow cells to point-B.

2) Calculate $(\delta u_B,\ \delta v_B)$ via equation (4.7) .

3) Calculate $(\delta u_i,\ \delta v_i)$ via equation (4.8) for all the shadow cells of the wall point-B.

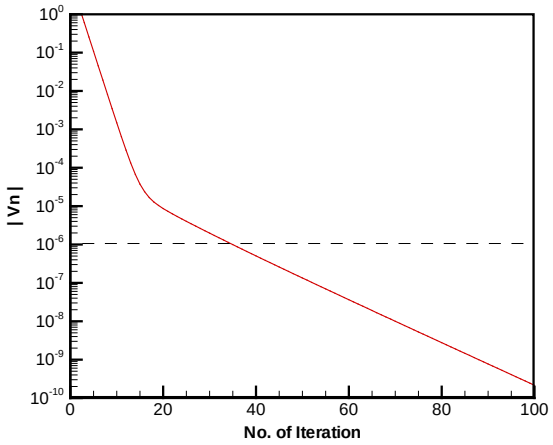
4) Repeat the steps 1), 2) and 3) for all wall points j=1, 2, … , J.

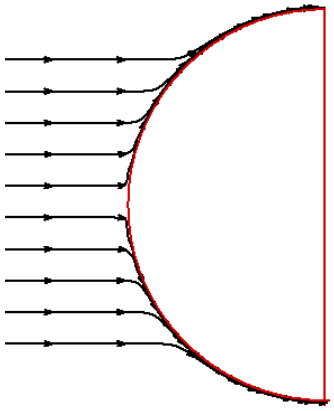
5) Calculate $(\delta u_i,\ \delta v_i)$ via equation (4.9) for all the shadow cells considering the velocity correction contribution from the nearby wall boundary points.

6) Calculate the corrected velocity (u_i^*, v_i^*) via equation (4.10) for the shadow cells, and the set the corrected velocity (u_i^*, v_i^*) as the new velocity on the shadow cells.

7) Repeat steps 1) to 6), until the no-penetration condition satisfied, or $V_{NB} = 0$.

Experiments show that no-penetration boundary condition can be well satisfied by running the above iterative velocity correction process in about 50 to 100 iterations when the average normal velocity drops below the magnitude of 10^{-6}, or 0.0001% of the mainstream flow velocity. The iterative velocity correction method is validated by plotting the average normal velocity magnitude on the wall boundary versus the number of iteration, as shown in Figure 4.2 (a). The average normal velocity on the wall boundary is monitored and used to determine the satisfaction of the no-penetration condition in the current work. The Mach 3 flow over a circular cylinder problem is used for this validation. A uniform velocity field at U=3 is defined as initial condition in the domain. Figure 4.2 (b) shows the streamlines after the velocity correction for the entire cylinder boundary, which shows no flow penetration.





(a) Normal velocity correction log (b) Streamlines over cylinder

Figure 4.2 Demonstration of explicit velocity correction method

4.1.2 Flux correction

The focus now turns to find an alternate method for the correction of pressure, density and energy on the shadow cells near the wall boundary. Besides the streamline curvature theorem and constant entropy associated to the wall boundary in the development of Euler solver, the wall boundary is also treated by satisfying zero flux for the inviscid flow.

Without the immersed wall(s) in the domain, the following Euler equations are fully satisfied if the boundary conditions are properly applied.

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}=0, \tag{4.11}$$

$$U=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ E\end{bmatrix},\quad F=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ u\left(E+p\right)\end{bmatrix},\quad G=\begin{bmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ v\left(E+p\right)\end{bmatrix}. \tag{4.12}$$

When the immersed wall(s) are introduced in the domain, the flow field

nearby the boundary is changed due to the effect of wall boundary conditions. Let (δU, F', G') denote the changes of the flow field for the conservative state and the flux terms due to the immersed wall(s), (U, F, G) and (U^*, F^*, G^*) denote the original conservative terms and the combined conservative terms, respectively. As the combined conservative terms also satisfy the governing equation (4.11) , the following equation can be derived

$$U^* = U + \delta U, \qquad F^* = F + F', \qquad G^* = G + G', \tag{4.13}$$

$$\frac{\partial U^*}{\partial t} + \frac{\partial F^*}{\partial x} + \frac{\partial G^*}{\partial y} = 0 \quad \rightarrow \quad \frac{\partial (U + \delta U)}{\partial t} + \frac{\partial (F + F')}{\partial x} + \frac{\partial (G + G')}{\partial y} = 0. \tag{4.14}$$

By shifting the two conservative terms (F, G) to the right side of the equation, it becomes

$$\frac{\partial (U + \delta U)}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = -\left(\frac{\partial F'}{\partial x} + \frac{\partial G'}{\partial y} \right). \tag{4.15}$$

Comparing equation (4.15) to equation (4.11) , the term on the right side of the equation is purely contributed from the immersed wall(s) and is named to be $\delta \vec{F}^w$, which will cause the change of state represented by δU .

$$\frac{\partial (\delta U)}{\partial t} + \frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = -\left(\frac{\partial F'}{\partial x} + \frac{\partial G'}{\partial y} \right) = -\nabla \cdot \delta \vec{F}^w. \tag{4.16}$$

Considering equation (4.11) , the above equation will be simplified to

$$\frac{\partial (\delta U)}{\partial t} = -\nabla \cdot \delta \vec{F}^w = -\left(\frac{\partial F'}{\partial x} + \frac{\partial G'}{\partial y} \right). \tag{4.17}$$

The δU in the above equation can be solved via finite volume method in a cell.

$$\frac{\delta U}{dt}A = -\int_L \delta \vec{F}^w ds \quad \rightarrow \quad \delta U = -\frac{dt}{A}\int_L \delta \vec{F}^w ds, \tag{4.18}$$

where A and L is the area of the cell and immersed boundary in the cell, respectively.

To a shadow cell near the wall boundary, the contribution of $\delta \vec{F}^w$ cannot be calculated directly. On the wall boundary, $\delta \vec{F}_n^w$ can be calculated using the normal velocity and flow variables calculated through the nearby Cartesian shadow cells. Then the $\delta \vec{F}_n^w$ on the wall boundary can be distributed back onto the nearby shadow cells, or Cartesian cells, where the governing equations are solved.

Following the finite volume method in solving the Euler equation, $\delta \vec{F}_n^w$ on wall boundary are assessed as

$$\delta \vec{F}_n^w = \begin{bmatrix} \rho^w v_n^w \\ \rho^w u^w \vec{v}_n^w + p^w \\ \rho^w v^w \vec{v}_n^w + p^w \\ \left(E^w + p^w\right)\vec{v}_n^w \end{bmatrix}. \tag{4.19}$$

In the equation above, the superscript (w) denotes the values of the variables on wall boundary, and the subscript (n) denotes the values for velocity projected on the normal direction. As the velocity components are corrected via the enforcement of no-penetration condition, the contribution of $\delta \vec{F}^w$ to the two momentum equations can be ignored. So the the contribution of $\delta \vec{F}^w$ for the mass conservation equation and energy conservation equation on the

wall boundary will be taken into account. Hence, $\delta\vec{F}_n^w$ for the mass conservation equation and energy conservation equation on the wall boundary will be calculated only

$$\begin{bmatrix} \delta\vec{F}_n^w(1) \\ \delta\vec{F}_n^w(4) \end{bmatrix} = \begin{bmatrix} \rho^w\vec{v}_n^w \\ \left(E^w + p^w\right)\vec{v}_n^w \end{bmatrix}. \tag{4.20}$$

The values of flow variables on the wall boundary are calculated using the delta function as given in subsection 4.1.1 before the velocity correction is performed.

The flux contribution $\delta\vec{F}_n^w$ calculated in equation (4.20) is on the wall boundary, or on the Lagrange nodes. Using the delta function proposed by Peskin, the flux contribution $\delta\vec{F}_n^w$ introduced by the wall boundary to the conservative variables *U(1, 4)* can be distributed onto the nearby Cartesian Eulerian shadow cells via

$$\delta U_i = \frac{dt}{dh^2}\sum_j \left(D_{i,j}\delta\vec{F}_n^w \cdot dS_j^w\right). \tag{4.21}$$

In the equation, it is assumed that the mesh spacing in X and Y direction is identical, hence $A = dh^2$; the $D_{i,j}$ is the delta function interpolation between the Cartesian cell (*i*) and the nearby Lagrange nodes (…, *j-1, j, j+1,* …); and the dS_j^w is the average arc length between two successive wall nodes (j)→(j+1), as shown in Figure 4.3.

Once the conservative variables *U(1, 4)* are updated, the density and energy

values are updated as well. Finally the pressure can be updated by the equation of state.

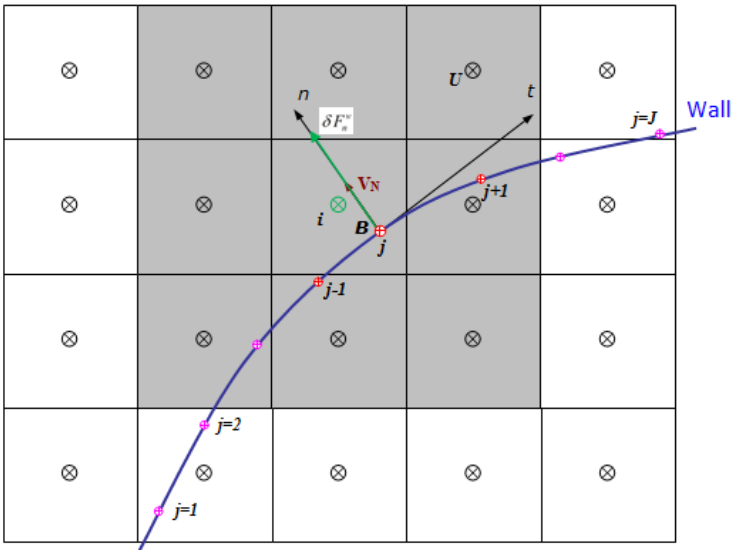


Figure 4.3 Calculation of normal flux on the wall boundary

Eventually, the conservative variables *U(1, 4)* are updated by

$$U_i = U_i^* + \delta U_i, \qquad with \quad \begin{bmatrix} U(1) \\ U(4) \end{bmatrix} = \begin{bmatrix} \rho \\ E \end{bmatrix}. \tag{4.22}$$

The flow chart plotted in Figure 4.4 summarizes the whole approach in implementing no-penetration condition and zero normal flux condition on the wall boundary. In all the steps of implementing the no-penetration condition and the zero normal flux condition, the normal direction of the wall boundary and the delta function interpolation between the wall node and the surrounding shadow cells are needed only. There is no need to compute the curvature of the boundary and this avoids the process to identify whether the shadow cells are in fluid domain or solid domain.

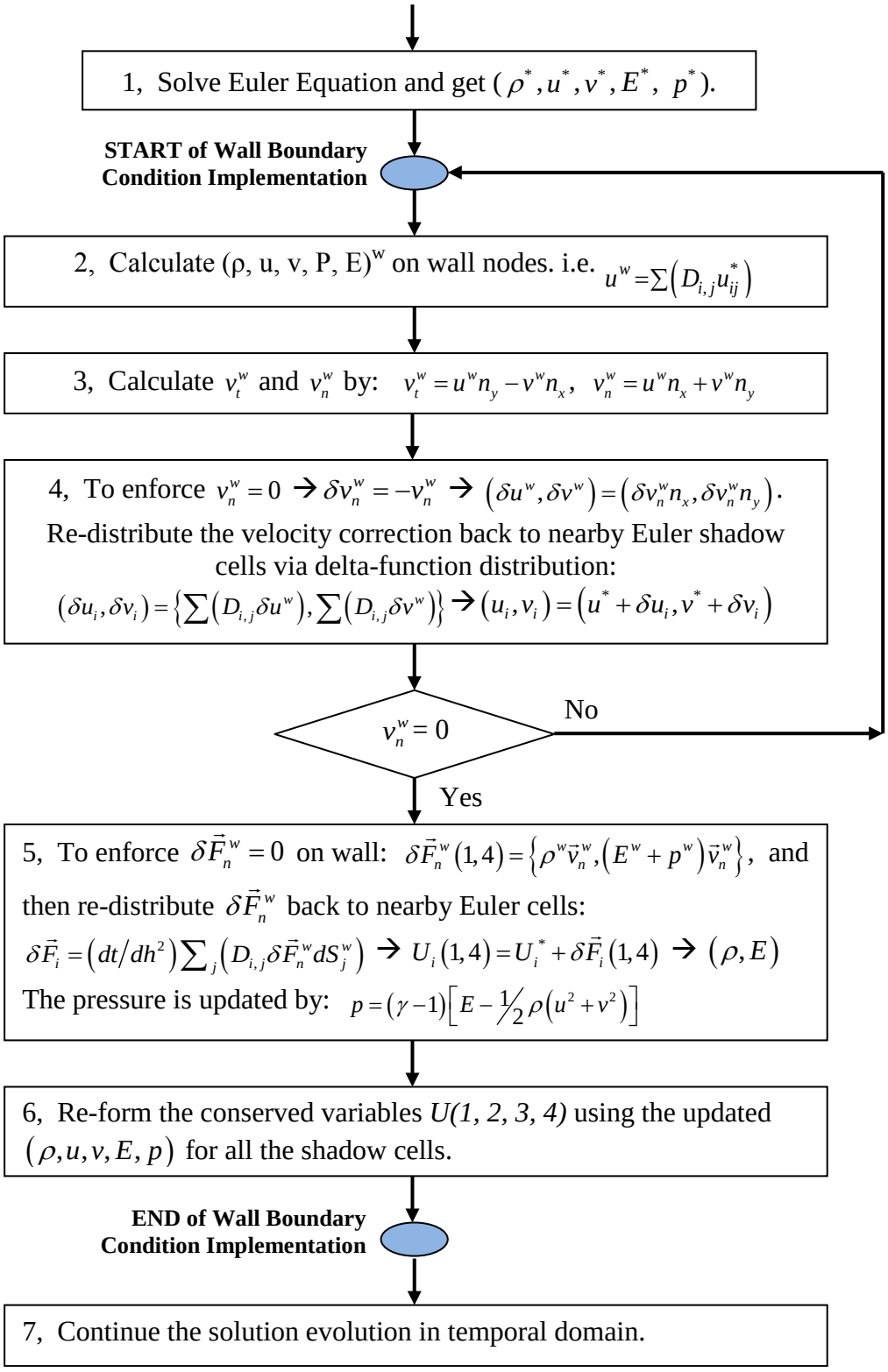


Figure 4.4 Implementation of no-penetration condition and zero normal flux condition for wall boundary

4.2 Validation analysis

The Mach 3 supersonic flow over a 2D circular cylinder is used to validate the proposed method. The computational domain and boundary condition are defined the same as those described in subsection 3.2.1. To get better understanding of the new approach in the implementation of immersed wall boundary condition, the solver is run on three different mesh sizes as shown in Table 4.1. The results are also used for the analysis of grid independent solution. Both first order flux scheme and second-order flux scheme are tested and compared as well.

dh	Mesh Size	No. of Nodes in Diameter
1/10	40 × 80	20
1/20	80 × 160	40
1/40	160 × 320	80

Table 4.1 Mesh configuration for grid independent solution study

Using the first order flux scheme, the four pressure profiles on the central line before the cylinder are plotted for the comparison, as shown in Figure 4.5. The pressure contours, Mach number contours and streamlines are plotted in Figure 4.6 based on the finest mesh of dh=1/40. The distribution of pressure and Mach number agrees with the solution obtained via the ghost-cell method presented in Chapter 3 and the results published by Visbal and Gaitonde [44] using sixth-order Roe scheme on BFC mesh. The streamlines plotted in Figure 4.6 (b) demonstrate the fluid flows over the cylinder wall smoothly and

closely, which indicates the no-penetration condition of the immersed cylinder wall boundary being well satisfied. The pressure profile plotted in Figure 4.5 shows that with the finer mesh, the shock wave can be captured much sharply and closely to the published numerical solution that was obtained by C6F8-Roe method [44], and the pressure profile behind the shock wave matches better to the numerical solution with finer mesh. The position of the shock wave converges at around X=-1.67 using the current solver with the first-order flux scheme. This is very close to that reported in the numerical solution, X=-1.7. The maximum pressure predicted after the shock wave and before the cylinder wall is 11.88, about 2% lower than the exact solution (12.15). This could be due to the dissipation losses introduced by the numerical viscosity in the solver implemented with the first-order flux scheme.

It is observed that the pressure value in the cells just before the cylinder wall at X=-1 drops, as shown in the pressure profile in Figure 4.5. This is different to the solution trend obtained using the traditional body-fitted grid [43], [44] or the ghost-cell method as presented in Chapter 3. In those solutions, the pressure reaches the maximum value on the cylinder wall at X=-1. The difference indicates the influence of the pressure distribution by the implementation of the immersed wall boundary condition, though the no-penetration condition is well satisfied. An acceptable reason for the pressure loss near the cylinder wall is due to the damping effect introduced by the implementation of IBM wall boundary. This is the characteristic of the IBM and cannot be avoided. On the other hand, the comparison of the pressure profile on different mesh sizes also implies that the impact of the pressure loss

becomes weaker with finer mesh.

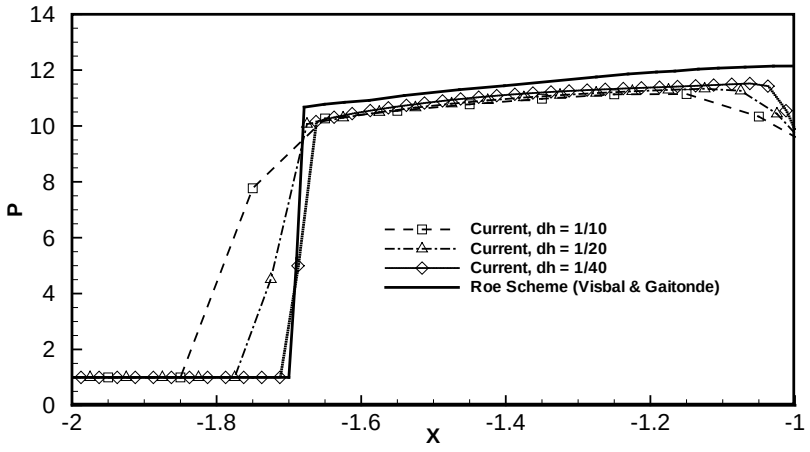
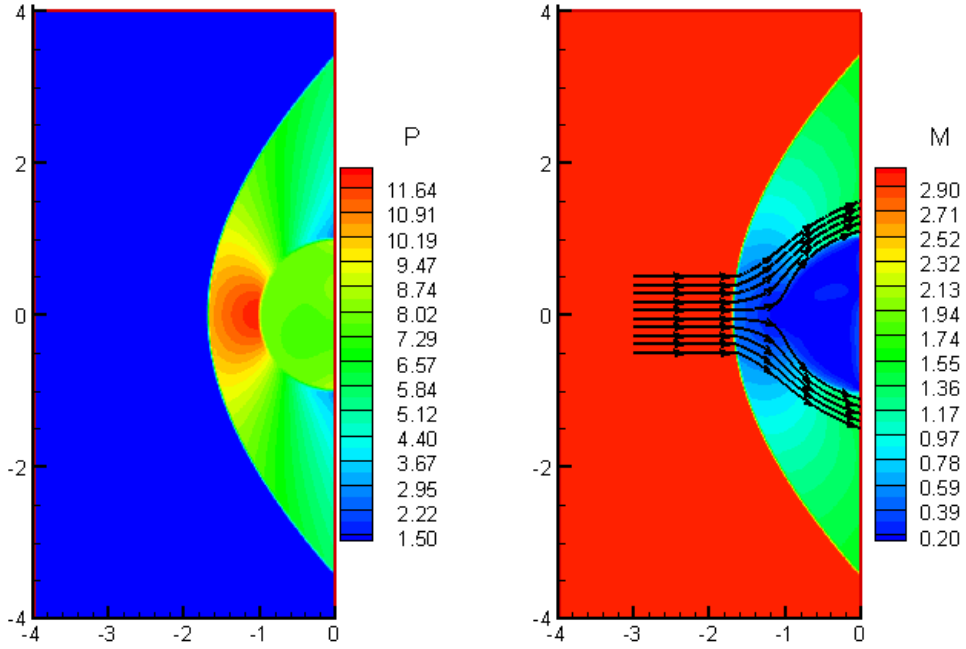


Figure 4.5 Comparison of pressure profile with Roe scheme (1st order scheme)



(a) Pressure Contours (b) Mach number contours & Streamlines

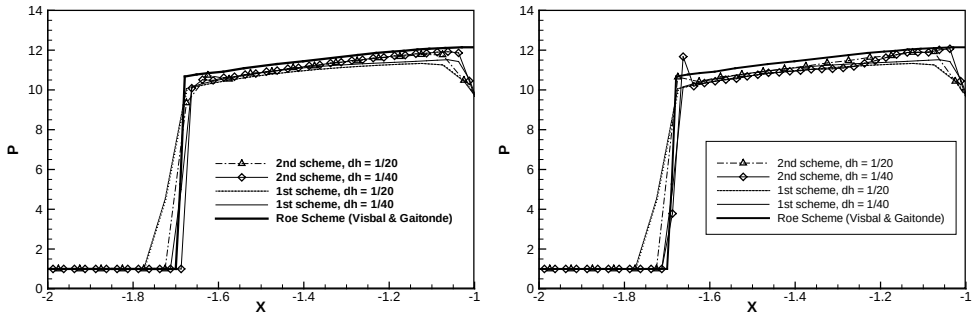
Figure 4.6 Pressure contours (1st order scheme)

The same problem is also solved by the second-order flux scheme to improve the accuracy of the solution and also to study the accuracy difference among

different flux limiter functions. All the five flux limiter functions are tested. The pressure profile obtained from using the limiter functions of Minmod, Osher and Sweby is more accurate and close to the exact solution, as shown in Figure 4.7. The shock wave captured by the second-order flux scheme is sharper, and the maximum pressure behind the shock wave is closer to the exact solution. However, pressure oscillation occurs just at the shock wave position when the limiter functions of Van Leer and Van Albada are used. Similar phenomenon of pressure oscillation is observed when higher β value (i.e., β=1.9) is defined for the limiter functions of Osher and Sweby. When β=1.1, the pressure oscillation at the shock wave position is not found. This illustrates that flux limiter functions and the value of the damping factor must be chosen and tested carefully when the second or higher-order flux schemes are used to compute the solution.

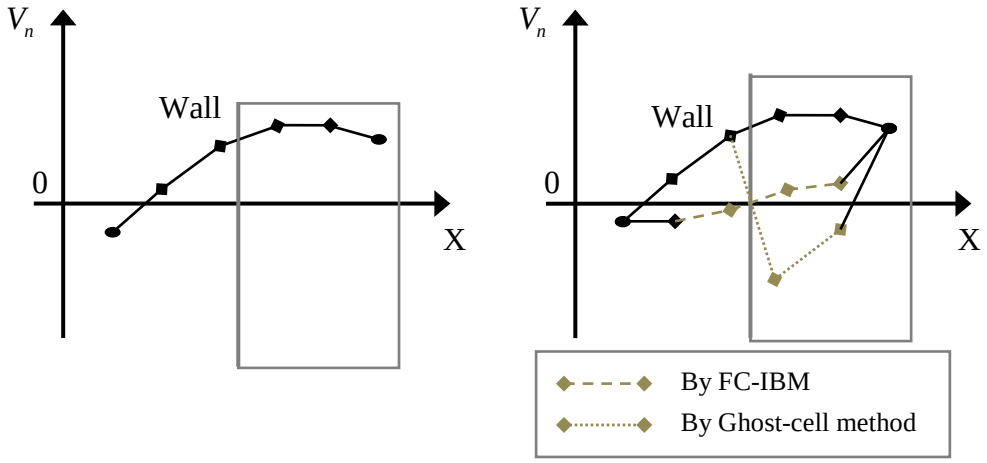
The new approach for IBM implementation is well validated by the Mach 3 supersonic flow over a 2D cylindrical circle. The shock wave and pressure profile obtained through the new approach agree well with the actual solution. However, it is noted that velocity and pressure near the wall boundary are affected slightly due to the implementation of no-penetration wall boundary condition. This is a unique feature that is introduced by the implementation of the immersed boundary method, as the velocities at the Eulerian cell centers at the both sides of the wall boundary are corrected in order to satisfy the no-penetration wall boundary condition. In the conventional ghost-cell method or symmetrical wall boundary method, the velocities at the Eulerian cell centers that are inside the solid domain will be corrected. Figure 4.8 illustrates the

difference in the normal velocity correction by IBM and ghost-cell method. Though both methods can enforce normal velocity $V_n = 0$, the velocity fields near the wall boundary are different. The velocity distribution is relatively smooth after the correction by the current IBM, and a sudden velocity change usually exists after the correction by ghost-cell method or symmetrical method. The sharp velocity change indicates that greater deceleration is produced to the fluid, and the smooth velocity distribution indicates that lower deceleration is produced to the fluid. At this point of view, the implementation of the IBM weakens the "reflection" condition near the wall boundary. It is known that pressure will increases in the situation of flow deceleration. This may explain why the flow reaches the maximum pressure point at the cylinder boundary in the implementation of ghost-cell method and symmetrical wall boundary method, as the sharp deceleration always exists on the boundary. In the current implementation, the velocity near the boundary is smoothed and no deceleration exists, hence the pressure near the wall boundary drops.



(a) Minmod Limiter (b) Van Leer Limiter

Figure 4.7 Comparison of pressure profile (2^{nd} order scheme)



(a) Normal velocity before correction (b) Normal velocity after correction

Figure 4.8 Normal velocity correction for immersed wall boundary

4.3 Numerical test cases and results

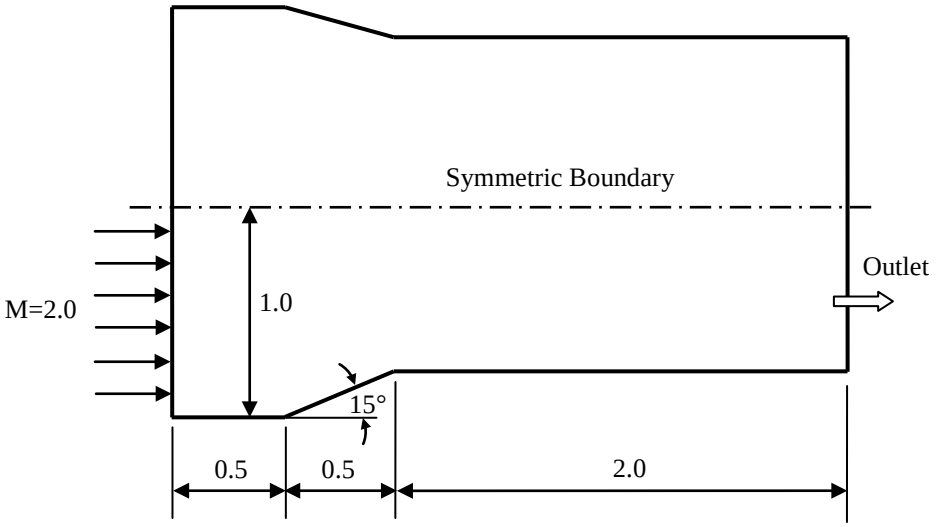
The new approach implemented is validated by the Mach 3 supersonic flow over a 2D circular cylinder. Compared to the conventional implementation by ghost cell method or symmetrical method, the new approach is much easier and simple. In this section, the method is used to compute other test cases and the results are discussed. All the test cases presented in this section are two-dimensional problems.

4.3.1 Supersonic flow over a wedge

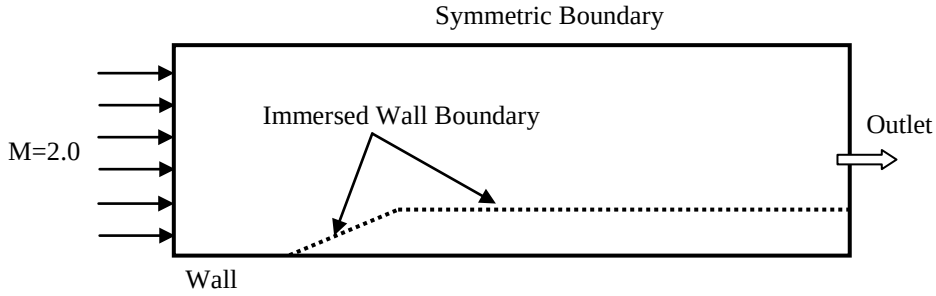
To further test the new approach, a supersonic flow over a wedge is considered. The incoming supersonic flow is at M_∞=2. The original configuration of the problem is a 2D supersonic flow in a symmetric convergent channel, where both the top and bottom walls are bent inward to form a 15º convergent section, as shown in Figure 4.9 (a). As the geometry of

the physical domain and boundary condition are symmetrical, only the bottom half of the physical domain is considered as the computational domain. The central line of the channel becomes the top boundary of the computational domain, and the symmetrical boundary condition is defined. A rectangle domain in size of L×H=3×1 represents the computational domain; the wedge is modeled as the immersed wall boundary as indicated in Figure 4.9 (b). The left boundary of the domain is defined as supersonic flow inlet with M_∞=2; the bottom boundary is defined as wall; and the right boundary is defined as outlet.

The computational domain is meshed as coarse uniform mesh of 120×40, or dh=0.025. The height of the wedge is 0.134. So there are only 5 coarse cells at the height of the wedge. To resolve the wedge wall shape accurately and improve the accuracy of the influence to the mainstream flow by the immersed wedge wall, fine meshes are adapted along the wedge wall at the beginning of the solution. Figure 4.10 shows the final solution-adapted mesh distribution for (a) coarse mesh, (b) 1-level adaption mesh and (c) 2-level adaption mesh. The mesh distribution clearly demonstrates that fine meshes are adapted to the shock wave position, where high flow gradient exists. The final number of mesh cells for three cases is 6573, 10974 and 26850, respectively. The corresponding contours of Mach number are plotted in Figure 4.11 for the three different meshes. From the contours of Mach number, it is observed that the shock wave is captured shaper and shaper from coarse mesh to one level adaption and further to 2-level adaption.



(a) Original configuration of the convergent channel



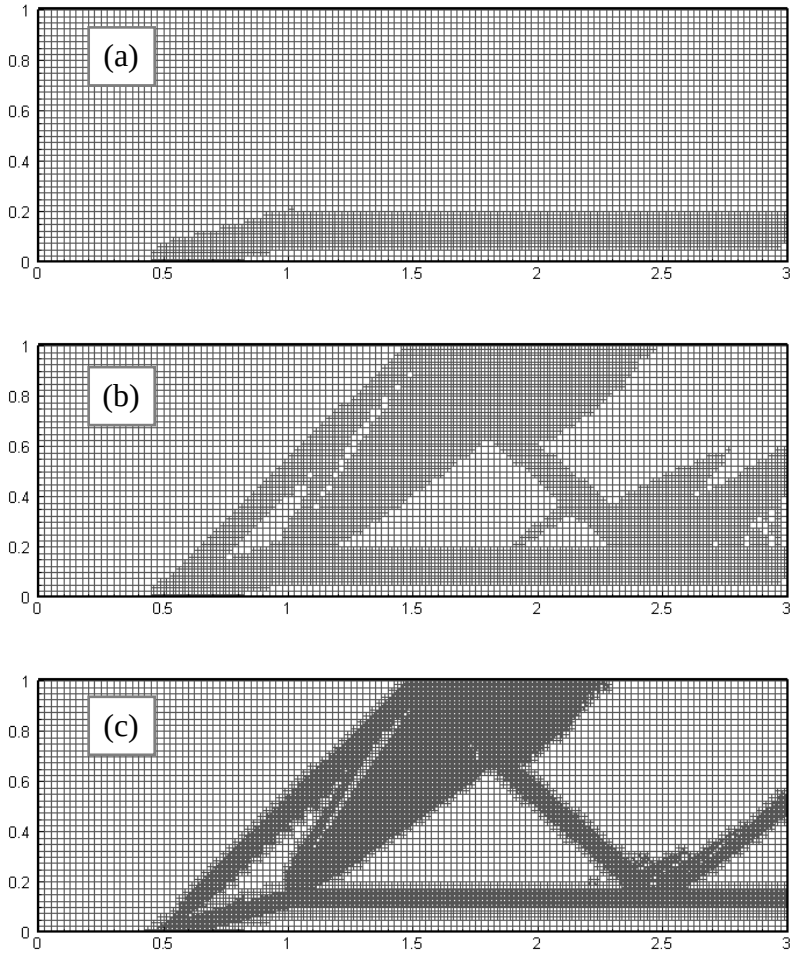
(b) Computational domain and boundary condition for the wedge and channel

Figure 4.9 Configuration and boundary condition for supersonic flow over a wedge

The Mach number after the wedge shock (location B as indicated in Figure 4.12) and the angle of the wedge shock (angle β as indicated in Figure 4.12) are often used for accuracy assessment to this case. The analytical Mach number after the wedge shock at location B is 1.44 ([47], [48]) and the prediction by the current method is 1.497, 1.495 and 1.472 for uniform coarse mesh, 1-level adaption and 2-level adaption solution, respectively. The theoretical value of the angle β of the wedge shock emitting from the front wedge point is 45.38º. The angel β of the wedge shock predicted by the current solver is 45.09º, 45.14º, and 45.82º on three different mesh sizes. The

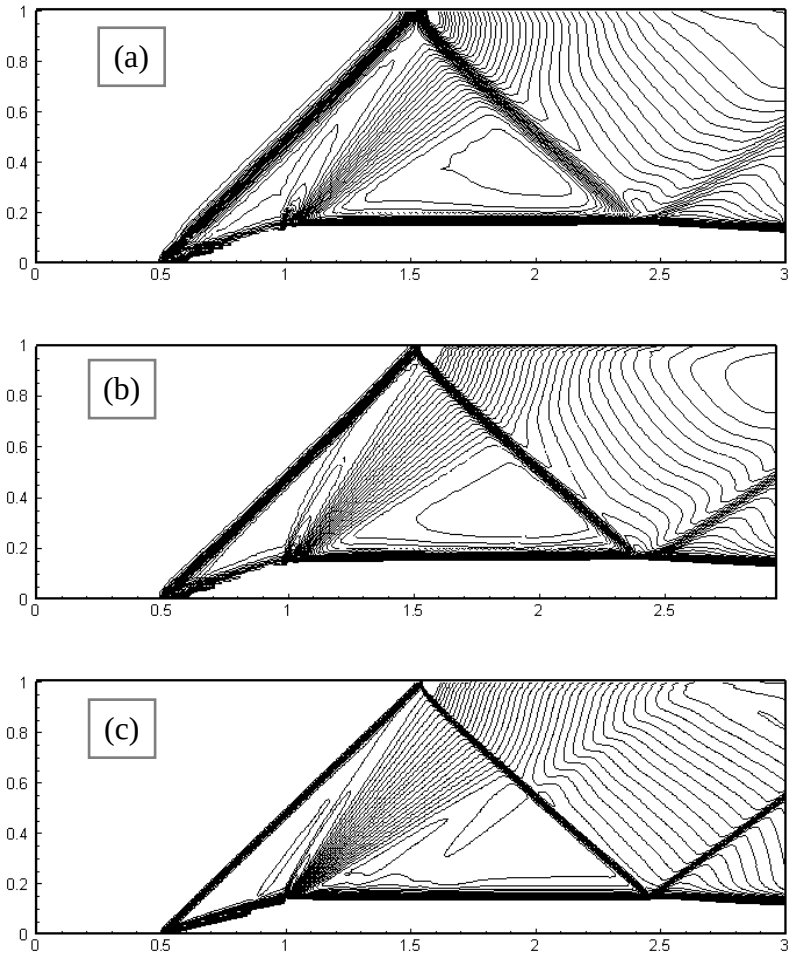
results are in good agreement with the theoretical analysis, which shows the capability of the current solver with new approach of IBM implementation in solving supersonic flow over a wedge.

With the solution adaption capability in the current solver, the computing time for obtaining the 1-level adaption solution is 567 seconds, only 125 seconds more than that needed for the coarse mesh solution. The computing time for the 2-level adaption solution is 2517 seconds, which is much less than 6140 seconds needed for the same uniform finer meshes.



(a) adapt to wedge wall by 1-level only, (b) adapt to wedge wall and solution by 1-level, (c) adapt to wedge wall and solution by 2-level.

Figure 4.10 Solution adaptive mesh for supersonic flow over a wedge



(a) adapt to wedge wall by 1-level only, (b) adapt to wedge wall and solution by 1-level, (c) adapt to wedge wall and solution by 2-level.

Figure 4.11 Contours of Mach number for supersonic flow over a wedge

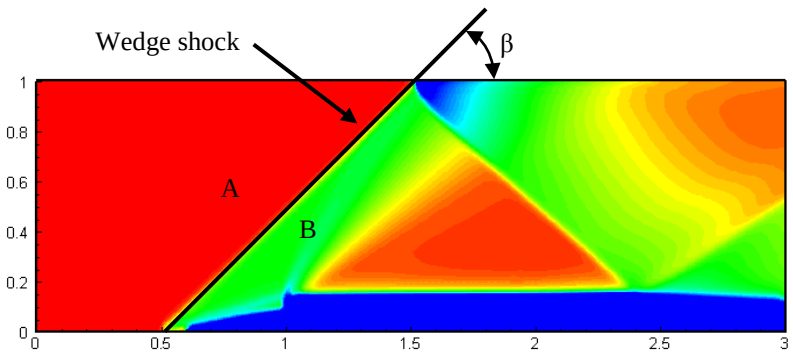


Figure 4.12 Wedge shock and solution parameters for supersonic flow over a wedge

4.3.2 Supersonic flow over a double-ellipse

In this subsection, the supersonic flow over a double-ellipse is simulated. The immersed wall boundary is represented by two elliptical segments in this problem. They are defined by:

$$\begin{cases} x \le 0 & \begin{cases} y \le 0: & (x/2.4)^2 + (y/0.6)^2 = 1, \\ y \ge 0: & (x/1.4)^2 + (y/1.0)^2 = 1, \end{cases} \\ 0 \le x \le 0.6 & \begin{cases} y \le 0: & y = 1.0, \\ y \ge 0: & y = -0.6. \end{cases} \end{cases} \tag{4.23}$$

The geometry of the double-ellipse is similar to the aircraft nose, as shown in Figure 4.13. The incoming supersonic flow of M=2.0 passes through the double-ellipse at the angle of attack 20º. The computational domain is set as a rectangle 6×8, with the double-ellipse wall immersed at the right zone of the domain. The left and bottom boundaries are defined as supersonic inlet, and the right and top boundaries are defined as outlet.

As the incoming flow is supersonic at Mach number 2.0, a bow shock will be generated before double-ellipse wall. Due to asymmetrical outline of the double-ellipse wall boundary and the incoming flow at an angle of attack 20º, the shock generated at the bottom and top part of the double-ellipse wall will be different. Because of the characteristic of the supersonic flow, another shock will be emitted from the intersection corner of the two elliptic segments.

The solution is obtained on coarse uniform mesh of 60×80, and further refined by 1-level and 2-level solution adaption with density gradient as the refinement indicator. Figure 4.14 shows the mesh distribution for (a) coarse

mesh, (b) 1-level adaption mesh and (c) 2-level adaption mesh. The corresponding pressure contours for different mesh sizes are plotted in Figure 4.15. The figure clearly shows that sharper shocks can be captured with the finer meshes enabled by the solution adaption. By choosing the solution adaption variables and regions to adapt carefully, finer meshes will be generated in the zones near the shocks and hence to improve the accuracy in resolving the sharp changes before and after the shocks. The total number of mesh cells for the converged solution is 8376 and 22485 for 1-level adaption and 2-level adaption, respectively, or 1.7x and 4.7x more than the coarse uniform mesh. The computing time for the 1-level adaption and 2-level adaption is about 1.6x and 6.3x more than it for the solution on coarse uniform mesh. Longer computing time demanded for the finer mesh with 2-level adaption is partially due to more mesh cells adapted in the domain, and also partially due to smaller time step which has to be used on finer meshes for temporal evolution of the solution.

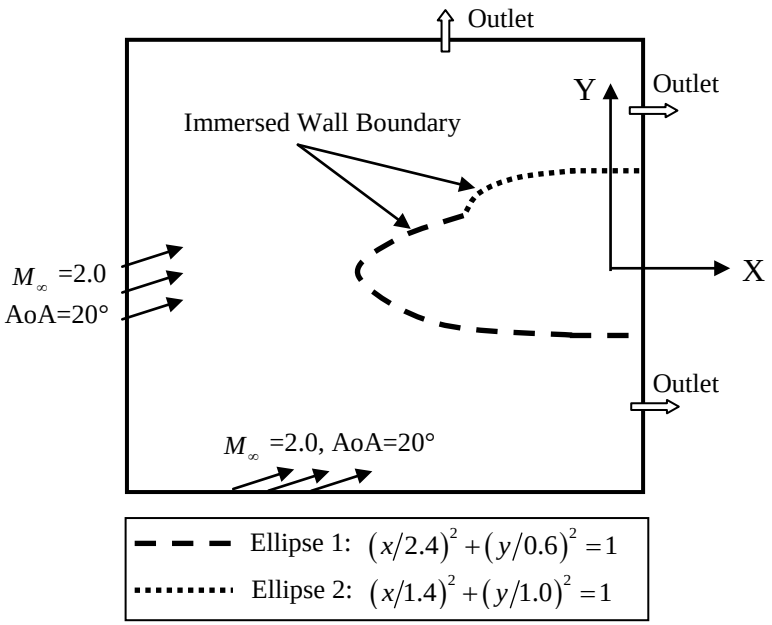
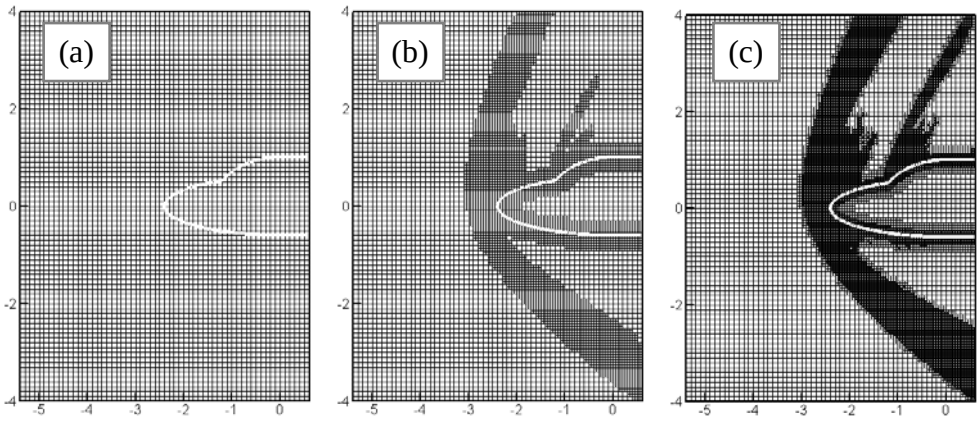
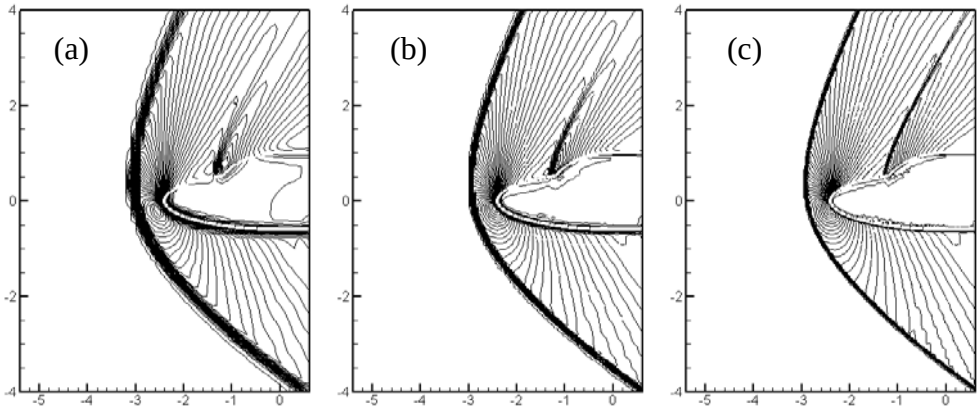


Figure 4.13 Configuration for supersonic flow over a double-ellipse



(a) Uniform mesh of 60×80, (b) 1-level adaption, (c) 2-level adaption

Figure 4.14 Double-ellipse case: solution adaptive mesh.



(a) Uniform mesh of 60×80, (b) 1-level adaption, (c) 2-level adaption

Figure 4.15 Double-ellipse case: pressure contours

From the contours of pressure plotted in Figure 4.15, it is observed that the bow shock before the double-ellipse wall and the wedge shock emitted from the corner of the two ellipse walls are captured and they maintain at the location consistently. As the current solver is based on finite volume method, smooth contours and nearly perfect shock profile are obtained with fine mesh. This agrees well with the finite volume solution obtained by Arminjon et al. [49]. The pressure coefficient C_p on the double-ellipse wall boundary is plotted in Figure 4.16. The C_p profiles obtained on three different meshes

match well with the results obtained by Arminjon et al. based on body-fitted mesh using finite volume method [49]. With finer mesh enabled by solution adaption, the C_p profile near the corner region of the two elliptical walls can be predicted more accurately.

The study of the supersonic flow past a double-ellipse demonstrates that the new approach can be used to simulate the supersonic flow over such kind of bluff body conveniently. Using the solution adaption function in the current solver, both the resolution of the wall boundary and the accuracy of the solution can be improved significantly and efficiently.

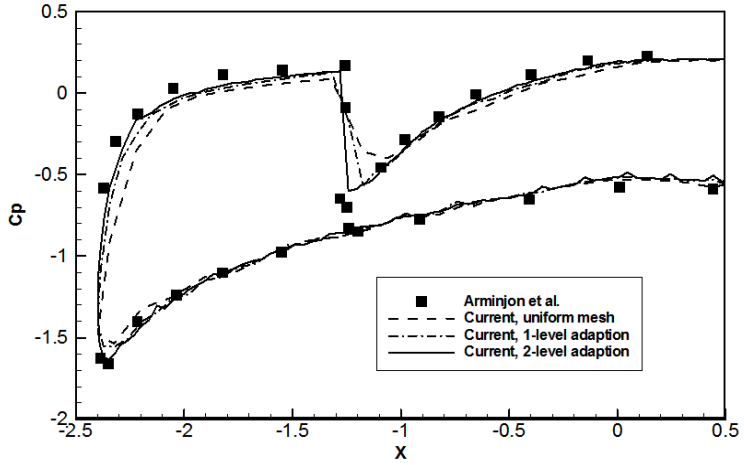


Figure 4.16 Double-ellipse case: pressure coefficient profile

4.3.3 High speed flow over a NACA0012 airfoil

Numerical simulation of high speed flow over airfoil is extremely interested by many researchers and engineers. High flow gradients, shocks and thin geometry interface are common problems associated with aerodynamics of

airfoil and require a great attention. Traditionally, either O-type or C-type of grid is generated around the airfoil geometry such that the grids can conform to the airfoil geometry nicely. This is especially important for the turbulent simulation and flow the prediction of friction force.

Due to the presence of thin geometry interface near the trailing edge of airfoil, special consideration was introduced to enforce the wall boundary condition at the corner near the trailing edge. Usually, additional cells or virtual cells are introduced for the upper edge of the airfoil and the bottom edge of the airfoil separately, so the wall boundary condition for the upper and bottom edges can be satisfied fully without interface. Such special local treatment can be found in many literatures ([9], [18]) and was also illustrated in Chapter 3. The advantage of the special local treatment is that the wall boundary condition on the edges near the sharp corner can be fully satisfied; however it is tedious and great attention must be taken to ensure that all the actual cells and additional/virtual cells are correctly used in the solver and the boundary condition implementation.

In this section, the high speed flows over an airfoil are simulated to further validate and study the new approach in IBM implementation. The airfoil profile is NACA0012. Flow conditions for subsonic, transonic and supersonic cases will be considered. The incoming free-stream Mach number for subsonic flow is 0.7, transonic flow is 0.8 and supersonic flow is 1.2. The angle of attack is 0^o for all the three flow conditions. Transonic flows Mach number of 0.7, with the angle of attack at 1.49^o and 4^o are solved for further analyses.

The solutions for the flows with the angle of attack at 0º, incoming free-stream Mach number at 0.7, 0.8 and 1.2 are plotted in Figure 4.17 in pressure contours and pressure coefficient profiles on the airfoil surface. The pressure coefficients are compared with experimental data.

For the subsonic case with flow condition as Mach=0.7 and AoA=0.0º, the flow around the airfoil are in subsonic range and the pressure changes along the airfoil surface are gently as shown in Figure 4.17 (a). The predicted pressure coefficient on the airfoil surface is in good agreement with the experimental data quoted by Lee et al. [50] and Yoshihara and Sacher [51]. For the flow condition with the free-stream Mach number increased to 0.8 and AoA at 0.0º, the supersonic flow and a strong shock are generated near the mid zone of the airfoil chord, as indicated in Figure 4.17 (b). The pressure coefficient, the strength and the location of the shock wave position match accurately with the experimental data and numerical results obtained by Lee et al. [50]. When the incoming free-stream is at supersonic condition with Mach number at 1.2, as indicated in Figure 4.17 (c) a strong bow shock wave is generated in front of the airfoil head and a pair of oblique shock waves are generated at the trailing edge of the airfoil. Compared to the numerical results obtained by Lee et al. [50], the location of the bow shock is predicted accurately but the oblique shock waves are slightly located before the trailing edge. The pressure coefficient distribution on the airfoil surface also indicates that the oblique shock occurs at about 90% of the chord length. This also causes the lower pressure coefficient near the trailing edge as compared to the experimental data. However, the pressure coefficient before the trailing edge

still matches with the experimental data.

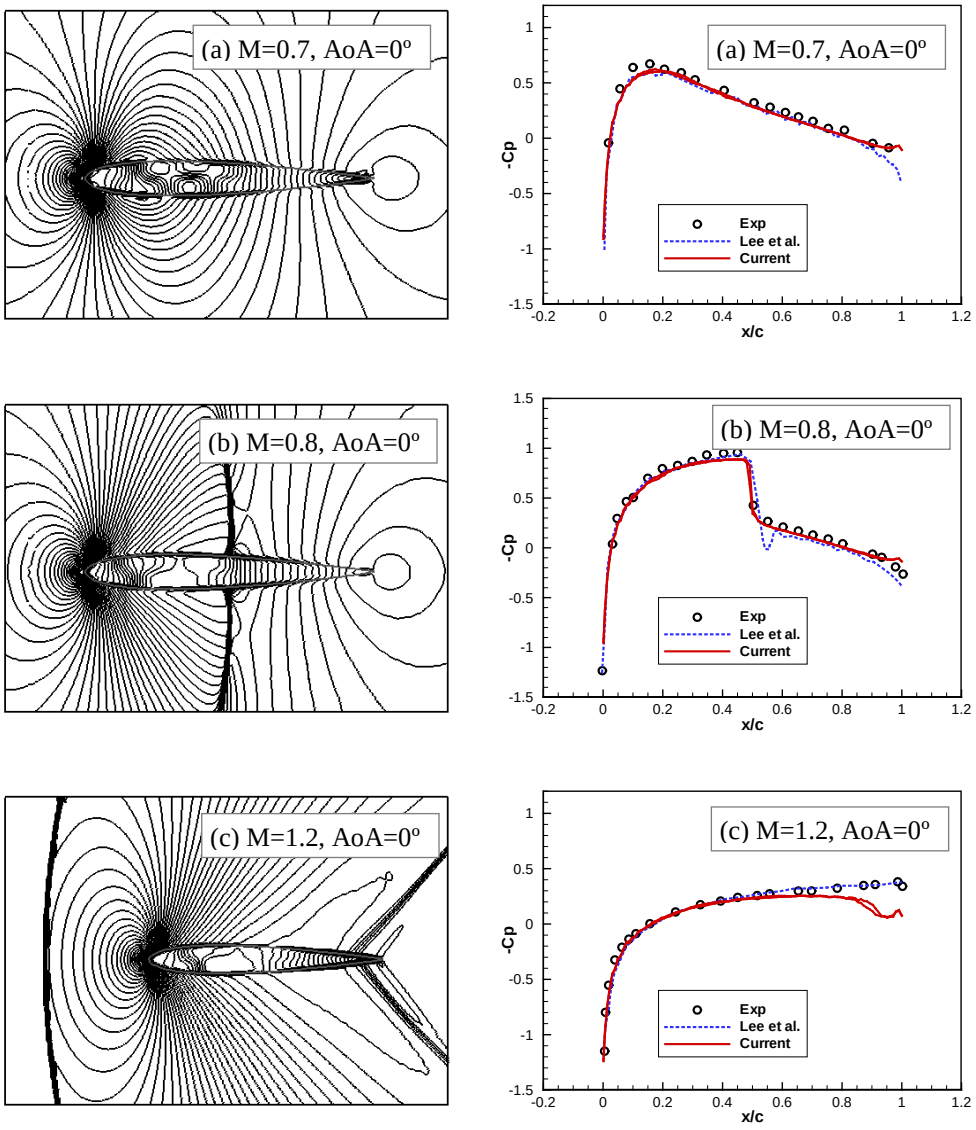


Figure 4.17 Pressure contours and coefficient profile for NACA0012 airfoil at AoA=0.0°

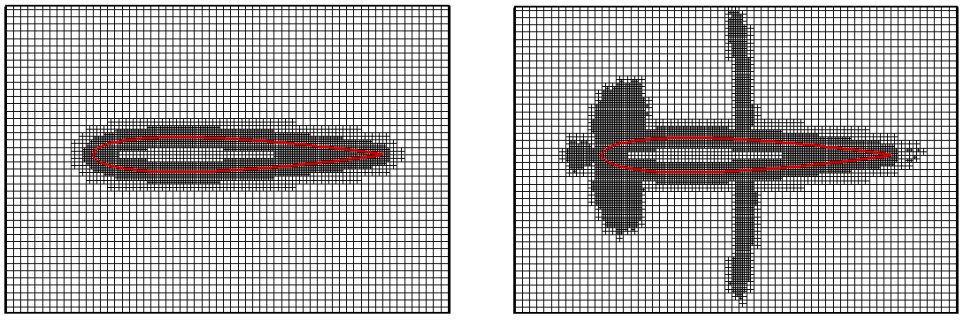
The pressure coefficients are predicted accurately for flow over the NACA0012 airfoil under the conditions of Mach number 0.7 and 0.8, AoA=0.0°. This demonstrates that the new approach of FC-IBM implementation is able to simulate the subsonic flow and transonic flow over the airfoil accurately. The inconsistency of the oblique shock waves predicted

in supersonic flow case implies that the conventional implementation of IBM, or the present FC-IBM, which is widely applied for incompressible flows, has some influence on the enforcement of the boundary condition near the trailing edge of the airfoil. From this point of view, special treatment for upper surface and lower surface is necessary when the local flow speed is supersonic.

In the high speed flows over the airfoil, the developed solution adaptive feature in the current solver demonstrates very good performance in shortening the computing time. The computational domain for airfoil dynamic analyses is usually chosen as 15x or 20x of the airfoil chord length to ensure that the pressure far-field boundary condition can be used accurately. In such a relatively big computational domain, local mesh refinement or mesh cluster are used to refine the meshes around the airfoil so that the airfoil geometry can be represented in acceptable resolution. Usually such fine meshes are also extended to the far-field boundaries. Hence the number of cells in the computational domain is usually large and leads to longer computing time. However, the flow changes are mainly occurred not far away from the airfoil.

Using the solution adaptive feature, the finer meshes can be adapted to the airfoil surface first to resolve the airfoil geometry in acceptable resolution at the beginning of the simulation. While the solution evolution is carried out, finer meshes are adapted to the regions bearing high flow gradients, for example high density gradient for the airfoil dynamics analyses. Figure 4.18 shows the solution adaptive mesh distribution for the analysis: (a) the initial mesh distribution with finer meshes adapted around the airfoil; (b) the final

mesh distribution for flow condition with Mach=0.8 and AoA=0.0º, where finer meshes are well adapted to the regions near the airfoil nose and the location of the shock waves. The computing time is cut down significantly in obtaining a more accurate solution based on finer mesh sizes.



(a) Initial mesh: adapted to airfoil (b) Final mesh: adapted to the solution

Figure 4.18 Demonstration of solution adaptive mesh for high speed flow over NACA0012 airfoil

The solution for subsonic flow under flow condition with the incoming free-stream Mach number of 0.7 and the angle of attack at 1.49º is plotted in Figure 4.19. The pressure contours in Figure 4.19 (a) show that the predicted flow structure is similar to that obtained by Lee et al. [50]. The pressure coefficient on the airfoil in Figure 4.19 (b) shows that the predicted pressure coefficient on the bottom airfoil surface matches with the experimental data but the pressure coefficient predicted on the upper airfoil surface is obviously lower. The difference of the pressure coefficient obtained on the airfoil surface for AoA=1.49º can be understood from the flow structure inside and around the airfoil.

As the NACA0012 airfoil is a symmetrical airfoil, the flow structure should be symmetrical when the angle of attack is at 0.0º. Under this condition, the

splitting point of the flow occurs at the leading edge of the airfoil and the merging point of the flow falls at the trailing edge of the airfoil. This is clearly observed from the plot of streamlines inside the airfoil as shown in Figure 4.20 (a), where a pair of closed recirculation flow structures is formed symmetrically inside the airfoil. When the angle of attack is at 1.49º, the splitting point of the flow near the leading edge shifts anti-clockwise slightly, similarly, the merging point of the flow also moves slightly away from the trailing edge in anti-clockwise direction. As a result, the recirculation flow structure inside the airfoil becomes asymmetrical as shown from the plot of streamlines in Figure 4.20 (b). Streamlines pass through the bottom of the airfoil surface nicely, but depart a bit near the trailing edge at the top of the airfoil surface. This indicates that a small recirculation zone is formed at the top airfoil surface near the trailing edge, which will influence the flow in this region and eventually affect the solution on the top of the airfoil surface. When the angle of attack is increased to 4.0º, the recirculation zone at the top airfoil surface near the trailing edge becomes much larger and the flow pattern has been alternated extremely, as illustrated in the plot of streamlines in Figure 4.20 (c).

The results for the NACA0012 airfoil at the angle of attack of 0.0º, 1.49º and 4.0º show that though the no-penetration wall boundary condition can be fully satisfied, the implementation of FC-IBM introduces viscous effect near the wall boundary. This eventually affects the solution accuracy for the current Euler solver as the viscosity of flow is not considered. In the implementation of FC-IBM for simulations of incompressible viscous flows, the actual

viscosity of flows dominates and the numerical viscous effect introduced by the IBM could be very minor and has little impact on the solution accuracy.

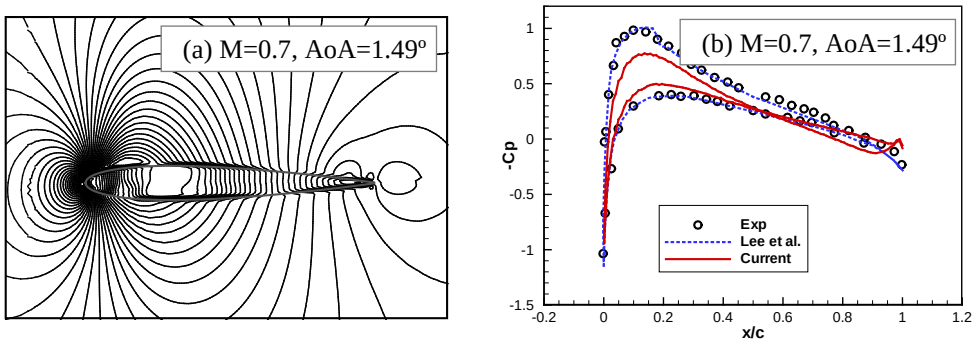


Figure 4.19 Pressure contours and coefficient profile for NACA0012 airfoil (AoA=1.49º)

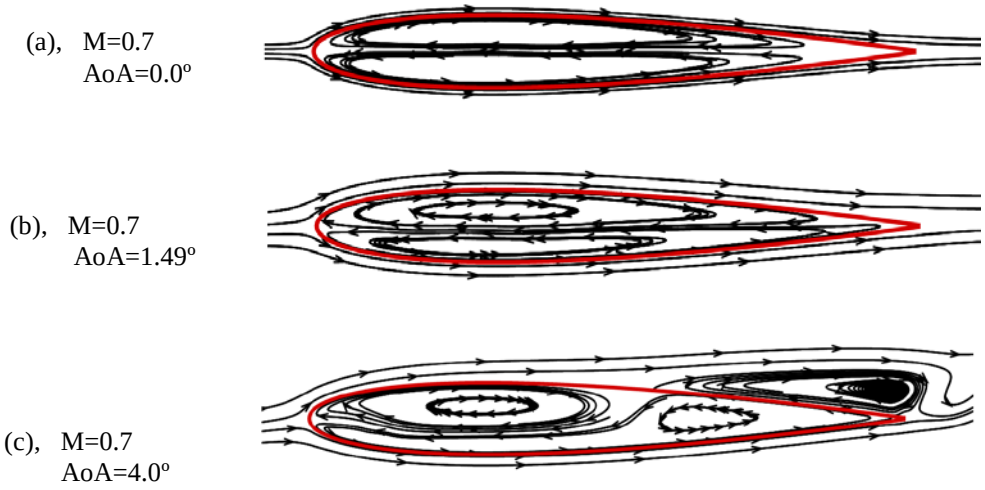


Figure 4.20 Streamlines inside NACA0012 airfoil at different angles of attack

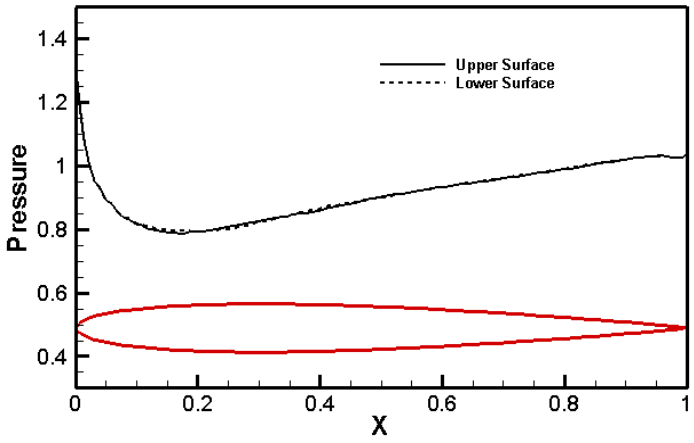
In the Navier-Stokes equation (4.24) , the divergence of stress at the right side of the equation consists of pressure gradient and viscosity terms. In Euler solver, the viscosity term $\mu\nabla^2 v$ is not considered. So the solution is determined by the convective acceleration term $\nabla\cdot\left(\rho v\vec{V}\right)$ and the pressure

gradient ($-\nabla p$). Under this condition, when there is numerical viscosity introduced to the solver, it will function as viscosity term and contribute to the divergence of stress in the equation. When strong convection and pressure gradient present, such numerical viscosity introduced will be relatively weak and is negligible; however weak convection and small reverse pressure gradient present, then such numeral viscosity introduced will change the state of the original equation and produce unexpected solution.

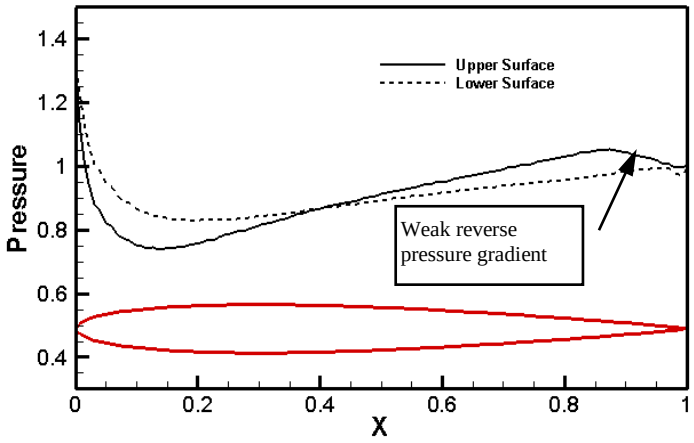
$$\frac{\partial \rho v}{\partial t} + \nabla \cdot \left(\rho v \vec{V} \right) = \overbrace{\underbrace{-\nabla p}_{\substack{\text{Pressure} \\ \text{Gradient}}} + \underbrace{\mu \nabla^2 v}_{\text{Viscosity}}}^{\text{Divergence of stress}} . \tag{4.24}$$

From the Mach number contours plotted in Figure 4.17 and Figure 4.19, it shows that strong and rapid convection occurs near the leading edge of the airfoil and relatively weak convection occurs near the trailing edge of the airfoil. The pressure distribution on the airfoil surface plotted in Figure 4.21 also shows that strong reverse pressure gradient presents near the leading edge of the airfoil for all the three conditions with AoA=0°, 1.49° and 4.0°. Weak reverse pressure gradient presents near the top trailing edge of the airfoil when AoA is not zero. As the analysis in the previous paragraph, under this situation, the numerical viscosity will contribute to the final solution and produce unexpected result. From the understanding of the viscous flow, the recirculation flow pattern usually occurs behind a non-streamlined object. This explains why flow recirculation is observed in the results of the airfoil flow when AoA is not zero, but not observed in the condition of AoA=0°.

(a), M=0.7
AoA=0.0º



(b), M=0.7
AoA=1.49º



(c), M=0.7
AoA=4.0º

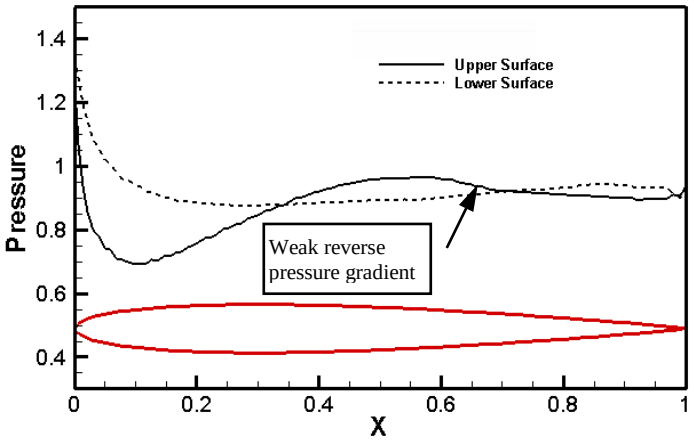


Figure 4.21 Pressure distribution on the surface of a NACA0012 airfoil at different angles of attack

4.4 Conclusions

A new flux correction-based immersed boundary method (FC-IBM) is presented for simulation of compressible inviscid flows. The concept of FC-IBM is to enforce the boundary condition by correcting the conservative terms and velocity field at the Cartesian grids near the boundary to satisfy the zero mass flux, zero energy flux and no-penetration conditions. The unique advantage of the present FC-IBM is that it avoids the tedious process to compute the boundary curvature and to identify whether the shadow cells are in fluid domain or solid domain.

The new method is implemented in the developed adaptive Euler solver. To validate and test the new method, supersonic flows over a circular cylinder, a wedge and a double-ellipse structure, transonic and subsonic flows over a NACA0012 airfoil are simulated. The results obtained from the new method are in good agreement with the available data in the literature. The benefit of the developed adaptive Euler solver is demonstrated in terms of accuracy improvement and computational efficiency. However, numerical viscosity effect is noticed for the method and this may affect the solution in the case that weak convection and small reverse pressure gradient exist.

Chapter 5 Local Domain Free Discretization - Immersed Boundary Euler Solver

In the implementation of 2D velocity correction based IBM (IBVCM) proposed by Shu et al. [22] [33], two mesh points are identified near each intersection point between the wall boundary and horizontal or vertical mesh lines and then the velocity on the wall intersection point is obtained via linear interpolation. The obtained velocity may not satisfy the physical velocity condition for wall boundary. Hence a correction is needed and applied to two mesh points to enforce the physical velocity condition being satisfied. In this correction process, special care is taken when one mesh point near the wall boundary is on both horizontal line and vertical line.

In almost all the studies of IBM for incompressible viscous flows, the velocity of the immersed wall boundary is usually known for either stationary body or moving body. This allows the implementation of local domain free discretization (DFD) and IBM for immersed wall boundary condition can be done directly and easily. However, for inviscid flows the velocity on the immersed wall boundary is not known for both stationary body and moving body. This becomes the big challenging issue in enforcing the immersed wall boundary condition for numerical simulation of inviscid flows. Moreover, the pressure and density conditions have to be addressed separately for compressible flow simulation, and this is usually not a concern for incompressible flows.

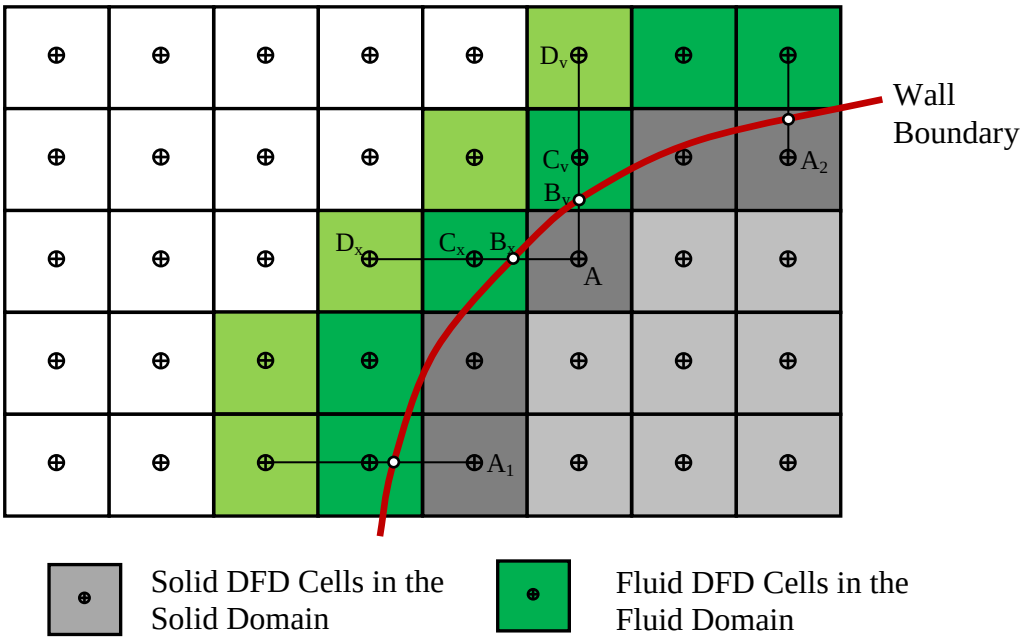
The fundamental feature of wall boundary condition for inviscid flow is no-penetration of flow in consideration of velocity field and mass conservation. The no-penetration of flow means that no fluid flows through the wall boundary, or the normal velocity on the wall boundary surface has to be zero. This feature is valid for both viscous flows and inviscid flows. For viscous flows, the tangential velocity on the wall boundary surface also becomes zero due to the viscous friction of fluid. So the wall boundary condition for viscous flows in fact consists of no-penetration (zero normal velocity) condition and no-slip (zero tangential velocity) condition. Since the normal velocity and tangential velocity on the wall surface are zero, the velocity components in Cartesian coordinate system are zero as well. For inviscid flows, the no-penetration condition stands similarly as that for viscous flows. As the viscous friction of fluid flow is neglected, the wall is considered as slip wall, which means zero gradient of tangential velocity on the normal direction of wall boundary surface. Therefore, from the viewpoint of velocity field and mass conservation, the wall boundary condition for inviscid flows shall satisfy zero normal velocity and zero gradient of tangential velocity in normal direction as viscosity of the fluid is not considered.

5.1 Local DFD (LDFD) method

Inspired from the implementation of local DFD method in solving the incompressible natural convection problems in concentric annulus [15] and unsteady flow around an oscillating circular cylinder [17], and the implementation of IBVCM in simulation of incompressible viscous flows around a circular cylinder [22], the concept of the local DFD method is

introduced into the current compressible Euler solver in enforcing the no-penetration and slip wall boundary conditions on the immersed wall boundary. To make it easier in the illustration of the implementation procedure, it is assumed that the wall boundary is stationary so there is no need to consider the moving velocity of the wall in calculating the normal and tangential velocity on the wall surface.

As illustrated in Figure 5.1, the cells fallen inside the solid domain and near the wall boundary are identified and tagged as solid DFD cells. For every solid DFD cell, two fluid DFD cells are identified across the wall boundary in either X direction or Y direction. For certain cases, there are two pairs of fluid DFD cells in both X direction and Y direction, such as solid DFD cell (A) in Figure 5.1. In the X direction, two fluid DFD cells are tagged as cell-(C_x) and cell-(D_x). The cell centers of cells (A, C_x, D_x) are on the X direction line and the intersection point of the X direction line and the wall boundary is (B_x). In the vertical direction along Y axis, two fluid DFD cells are tagged as cell-(C_y) and cell-(D_y). The cell centers of cells (A, C_y, D_y) are on the Y direction line and the intersection point of the Y direction line and the wall boundary is (B_y). For solid DFD cell (A_1), the fluid DFD cells only exist in X direction; and for solid DFD cell (A_2), the fluid DFD cells only exist in Y direction.



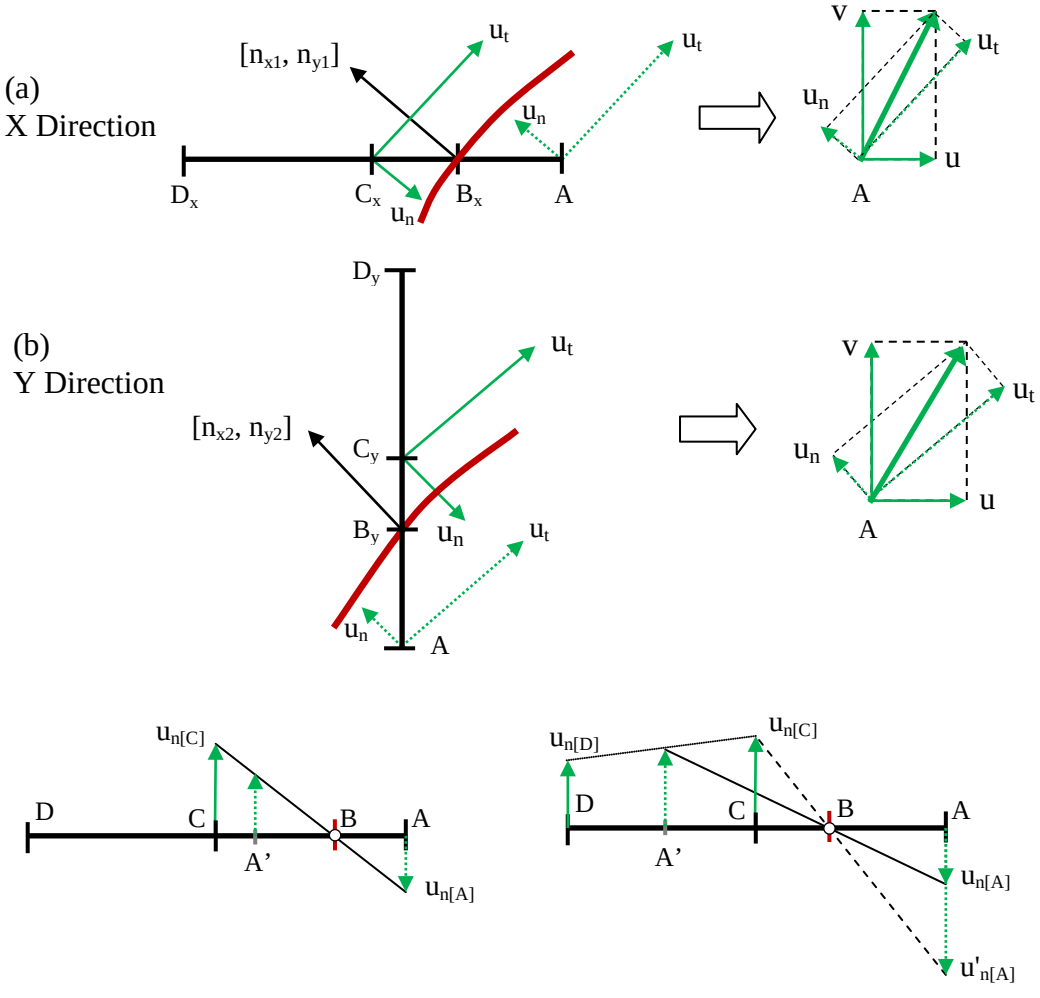


Figure 5.1 Illustration of Implementation of One-Sided local DFD Method

5.1.1 Velocity boundary condition

When fluid DFD cells exist in X direction for a solid DFD cell, the no-penetration and slip wall boundary condition is enforced by correction of the velocity at the solid DFD cell using the fluid DFD cells in X direction. The normal velocity and tangential velocity at cell-(C_x) and cell-(D_x) can be obtained via the velocity transformation based on the local normal direction vector (n_{x1}, n_{y1}) at wall point (B_x).

$$\begin{cases} v_t = u \cdot n_{y1} - v \cdot n_{x1}, \\ v_n = u \cdot n_{x1} + v \cdot n_{y1}. \end{cases} \tag{5.1}$$

To satisfy the no-penetration condition, or $v_n\big|_B = 0$, the normal velocity at the solid DFD cell (A) can be extrapolated from cell-(C_x) and the intersection point (B_x), as shown in Figure 5.1 (a) and (c).

$$v_n\big|_A = -\left(\overline{AC}/\overline{BC}\right)\cdot v_n\big|_B - \left(\overline{AB}/\overline{BC}\right)\cdot v_n\big|_C = -\left(\overline{AB}/\overline{BC}\right)\cdot v_n\big|_C . \tag{5.2}$$

In the equation above, $\overline{AB}$, $\overline{BC}$ and $\overline{AC}$ represent the horizontal distances along X direction among cell-(A), cell-(C_x) and intersection point (B_x). For the convenience, the subscript ($_x$) is omitted.

To avoid large extrapolation error when the intersection point (B_x) is closer to cell-(C_x), or $\overline{AB} >> \overline{BC}$ as shown in Figure 5.1 (d), the normal velocity at the solid DFD cell (A) will be extrapolated from cell-(C_x), cell-(D_x) and the intersection point B:

$$v_n\big|_A = -\left[v_n\big|_C + \frac{\overline{AB} - \overline{BC}}{\overline{CD}} \cdot \left(v_n\big|_D - v_n\big|_C \right) \right]. \tag{5.3}$$

The slip boundary condition is satisfied by simply assigning the tangential velocity at the solid DFD cell (A) using the tangential velocity at cell-(C_x).

$$v_t\big|_A = v_t\big|_C . \tag{5.4}$$

The normal velocity $v_n\big|_A$ corrected via equations (5.2) and (5.3) and the tangent velocity $v_t\big|_A$ corrected via (5.4) are able to satisfy the no-penetration and slip wall boundary condition at wall point (B_x). Therefore the corresponding Cartesian velocity components at the solid DFD cell (A) can be obtained as:

$$\begin{cases} u_1 = v_n\big|_A \cdot n_{x1} + v_t\big|_A \cdot n_{y1}, \\ v_1 = v_n\big|_A \cdot n_{y1} - v_t\big|_A \cdot n_{x1}. \end{cases} \tag{5.5}$$

Following the similar procedure, when fluid DFD cells exist in Y direction for a solid DFD cell, the no-penetration and slip wall boundary condition is enforced by correction of the velocity at the solid DFD cell using the fluid DFD cells in Y direction through equations (5.1)-(5.5). The length of $\overline{AB}$, $\overline{BC}$, $\overline{AC}$, and $\overline{CD}$ in the equations will be replaced by the vertical distance along Y direction among cell-(A), cell-(C_y), cell-(D_y) and intersection point (B_y), as shown in Figure 5.1 (b). The local normal direction vector is denoted as (n_{x2}, n_{y2}) of point (B_y), and used to replace (n_{x1}, n_{y1}).

$$\begin{cases} u_2 = v_n\big|_A \cdot n_{x2} + v_t\big|_A \cdot n_{y2}, \\ v_2 = v_n\big|_A \cdot n_{y2} - v_t\big|_A \cdot n_{x2}. \end{cases} \tag{5.6}$$

Consider that the solid DFD cell (A) in Figure 5.1 is connected to the fluid DFD cells in both X direction and Y direction, the velocity at cell (A) needs to take into account the corrected velocity that satisfies the wall boundary condition in both directions. A weighting factor is then introduced to X direction and Y direction as w_1^* and w_2^*, respectively. They satisfy the relationship of:

$$w_1^* + w_2^* = 1. \tag{5.7}$$

Then the velocity at the solid DFD cell (A) will be:

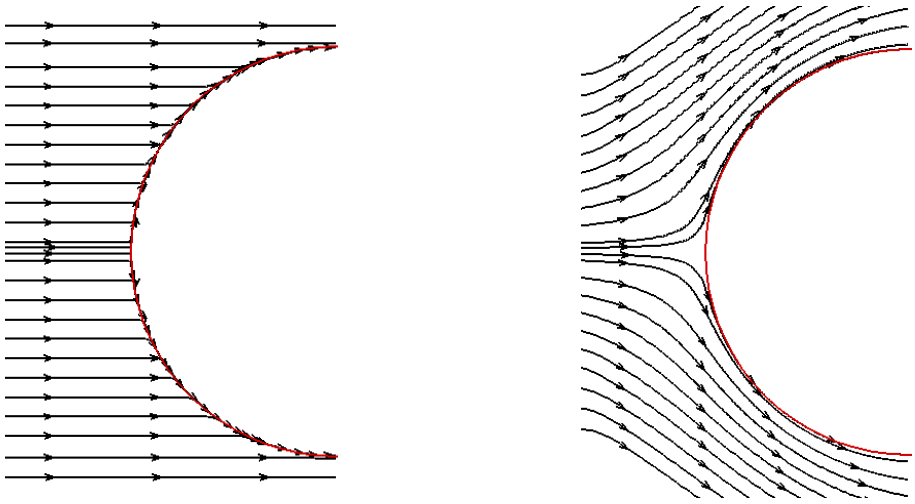
$$\begin{cases} u_A = w_1^* \cdot u_1 + w_2^* \cdot u_2, \\ v_A = w_1^* \cdot v_1 + w_2^* \cdot v_2. \end{cases} \tag{5.8}$$

Taking the average of the contribution from velocities in both X and Y directions is the simplest weightage. The weighting factors in this case are:

$$w_1^* = 0.5, \qquad w_2^* = 0.5. \tag{5.9}$$

The average weighting factor is tested by the Mach 3 flow over a circular cylinder. To demonstrate the effect on velocity correction in enforcing the no-penetration wall boundary condition, the streamlines are plotted around the cylinder, as shown in Figure 5.2. Streamline plotted in Figure 5.2 (a) shows that the no-penetration wall boundary condition can be well satisfied after first time step of solution evolution. Streamline plot for the converged solution in

Figure 5.2 (b) shows that the no-penetration wall boundary condition is perfectly satisfied. This comparison demonstrates that using the simple average weighting factors is an effective weightage method for velocity correction in enforcing the no-penetration wall boundary condition. Hence it is used for all the case studies in this thesis.



(a) Streamline after one time step (b) Streamline for converged solution

Figure 5.2 Demonstration of average weighting method for velocity correction

5.1.2 Pressure and density boundary condition

As wall boundary is considered as no-penetration and slip wall for inviscid flows, the pressure and density on the wall boundary are approximated as:

$$\frac{\partial p}{\partial n} = 0, \qquad \frac{\partial \rho}{\partial n} = 0. \tag{5.10}$$

Using the simple extrapolation, the pressure and density at solid DFD cell (A) are set to the same value as the nearest fluid DFD cell (C_x) or (C_y). Taking into account the weightage when a solid DFD cell is connected to the fluid DFD cell in both X direction and Y direction, the pressure and density on solid DFD

cell (A) are corrected by:

$$\begin{cases} P_A = w_1^* \cdot P\big|_{Cx} + w_2^* \cdot P\big|_{Cy}, \\ \rho_A = w_1^* \cdot \rho\big|_{Cx} + w_2^* \cdot \rho\big|_{Cy}. \end{cases} \tag{5.11}$$

The energy on the solid DFD cell (A) is then updated with the corrected values of velocity, density and pressure using the equation of state for ideal gas.

$$E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2\right). \tag{5.12}$$

Finally, the conservation terms in Euler equations will be updated on all the solid DFD cells at which all the flow variables are corrected according to the wall boundary condition.

5.2 Local DFD-Immersed Boundary Method (LDFD-IBM)

In the local DFD method proposed for compressible inviscid flows in the previous section, it saves the need to calculate the curvature of the wall boundary surface. However, the implementation of the local DFD method requires the determination whether a DFD cell is in either fluid domain or solid domain. Though it is not difficult to make the determination using the method described in the subsection 3.1, if such determination can be avoided, the solver will be much simpler and easier in actual application.

To improve the local DFD method and make the method to be easier and simpler in implementation and more generic to handle any irregular immersed wall boundaries, the local DFD-based immersed boundary method, named as

LDFD-IBM is proposed. Figure 5.3 illustrates the implementation of the LDFD-IBM. In this method, the X direction DFD cells are identified when the X direction line between the two successive centers intersects with the wall boundary. The two cells and the intersection point (W_x) are recorded as (B_x-C_x-W_x). The (-X) neighbor cell of (B_x) and the (+X) neighbor cell of (C_x) are found and also recorded as (A_x-B_x-C_x-D_x-W_x). The four cells and the intersection point form a typical local DFD unit for the implementation of wall boundary condition on wall point (W_x). Similarly, on Y direction, a typical local DFD unit (A_y-B_y-C_y-D_y-W_y) can be identified and recorded for the implementation of wall boundary condition on wall point (W_y). As the structure of the local DFD unit is similar on X direction and Y direction and the implementation procedure is also similar, the local DFD unit (A_x-B_x-C_x-D_x-W_x) on X direction is used for the illustration of implementation. For the convenience, the subscript ($_x$) will be omitted in following description.

In the local DFD unit (A-B-C-D-W), cells B and C are next to the wall boundary. So the flow variables on cells B and C need to be corrected in order to enforce the boundary condition on the wall. To correct the flow variables at cell B, a mirror point B' of the center of cell B is first found between the wall point W and cell D on the opposite side of the wall boundary; next the flow variables on the mirror point B' can be interpolated by using the wall point W and cell D; finally the flow variables on point B' are mirrored back to cell B based on no-penetration condition or zero normal gradient approximation. The step to correct the flow variables on cell B are summarized as below:

1) Find the mirror point B' of the cell B versus the wall boundary point W.

2) Calculate the normal velocity $v_n|_D$ and tangential velocity $v_t|_D$ on Cell D from equation (5.1).
3) Calculate the normal velocity at point B' by interpolation using equation (5.13).
4) Mirror the normal velocity at point B' back to cell B using equation (5.14) to enforce the no-penetration condition.
5) Set the tangential velocity at cell B the same as it at cell D to enforce the slip condition.
6) Calculate the corrected velocity in Cartesian coordinate system from equation (5.15).
7) Assume that the energy and density at cell B are kept the same, and update the pressure from the equation of state.
8) Update all the conservative terms in Euler equations using all the corrected flow variables.

$$v_n|_{B'} = \left(\overline{WB'}/\overline{WD}\right)\cdot v_n|_D = \left(\overline{WB}/\overline{WD}\right)\cdot v_n|_D , \tag{5.13}$$

$$v_n|_B = -v_n|_{B'} , \qquad v_t|_B = v_t|_D , \tag{5.14}$$

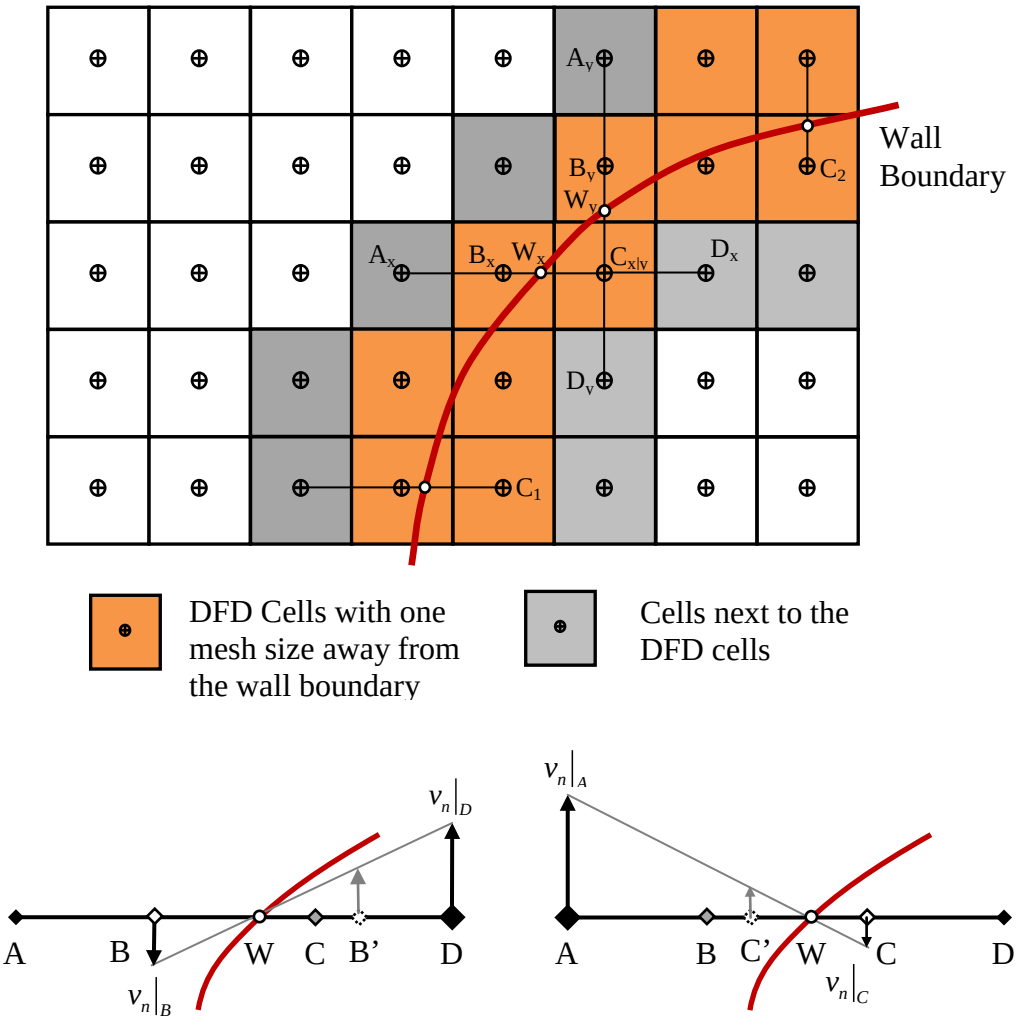
$$\begin{cases} u_B = v_n|_B \cdot n_{x1} + v_t|_B \cdot n_{y1}, \\ v_B = v_n|_B \cdot n_{y1} - v_t|_B \cdot n_{x1}. \end{cases} \tag{5.15}$$

Repeat the above steps, the flow variables at cell C can be corrected by the wall point W and cell A on the opposite side of the wall boundary.

As flow variables on cells near the wall boundary are corrected using the variables on the opposite side of the wall during the process of enforcing the wall boundary condition, the actual implementation includes the correction of

DFD cells from left side to right side and also from right side to left side on X direction. This is the concept of the proposed LDFD-IBM, inspired from the similar implementation for incompressible viscous flow simulation [15], [16], [17].

When a DFD cell is identified in the X direction local DFD unit and also in the Y direction local DFD unit, the weightage method introduced by equations (5.7)-(5.9) will be applied to calculate the averaged values on this DFD cell.



(a) Velocity Correction for DFD cell (B), from Right to Left

(b) Velocity Correction for DFD cell (C), from Left to Right

Figure 5.3 Illustration of Implementation of LDFD-IBM

5.3 Numerical validation and comparison

The LDFD-IBM is proposed based on the LDFD method with the consideration of the advantage of conventional IBM, in which it is not required to identify whether the cells near the wall boundary are in the fluid domain or solid domain. As described in the previous two subsections, the implementation of LDFD-IBM and LDFD method for compressible inviscid flows is quite similar. The difference is that the correction will be carried out for DFD cells in the solid domain in the implementation of LDFD method and for all DFD cells near the boundary for LDFD-IBM. To validate the two methods proposed, the following four compressible flow problems are solved by both methods and cross comparison is made and presented in this chapter.

5.3.1 Mach 3 supersonic flow over a circular cylinder

The Mach 3 supersonic flow over a circular cylinder has been studied by many researchers and many results are published in the literature and compared with the analytical data. The comparison can be made conveniently against the results available in the literature.

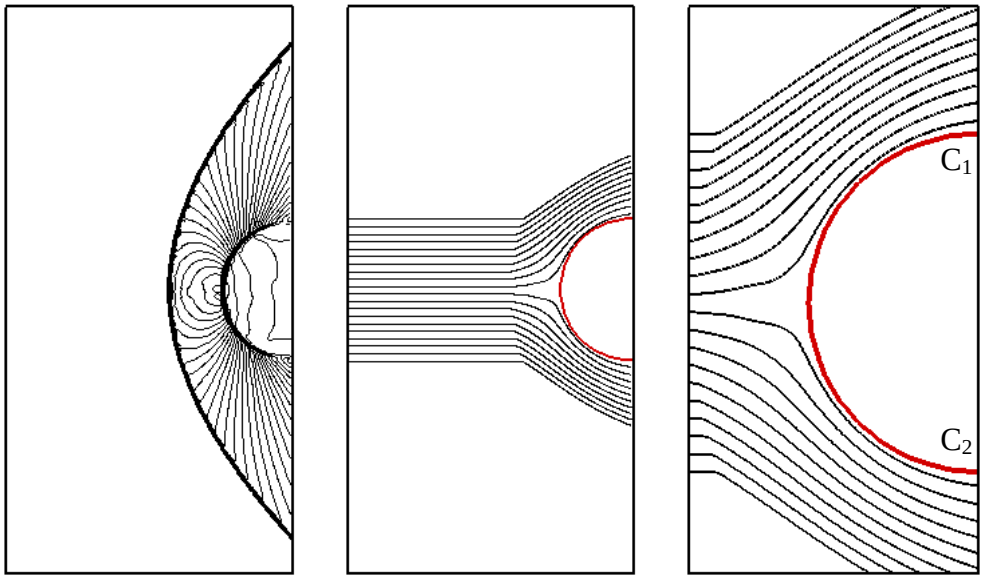
The computational domain and boundary condition for the Mach 3 supersonic flow over a circular cylinder is the same as stated in subsection 3.4.1. Uniform coarse mesh is defined in the computational domain, and the cylinder wall boundary is adapted with finer mesh cells to improve the resolution for wall boundary recognition.

The pressure contours and streamlines around the cylinder are plotted in

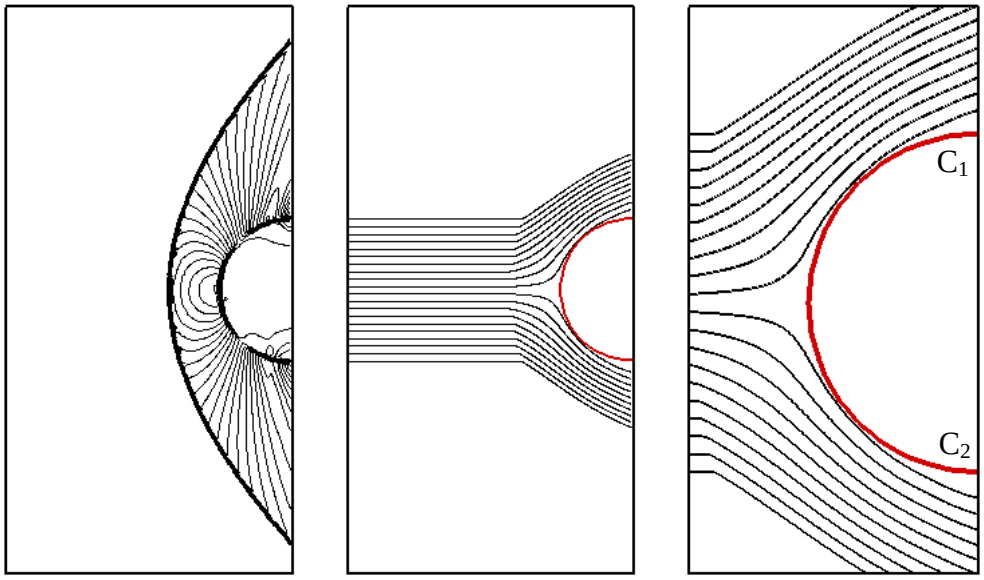
Figure 5.4. From the pressure contour plots, it can be observed that both LDFD method and LDFD-IBM are able to capture the bow shock wave upstream of the cylinder identically. By comparing the pressure distribution along the central line along X direction, the shock wave positions predicted by both methods are identical, as shown in Figure 5.5, and they are in good agreement with the results obtained by Qu [43], Fiorina and Lele [52] based on the body-fitted mesh solver. The pressure after the shock wave recovers and matches the values that were predicted by Qu and Fiorina and Lele. However, the pressure distribution just before the cylinder is noticed differently between the two methods. For LDFD method, the pressure value reaches the maximum towards the cylinder wall at $X = -1$; while for LDFD-IBM, the pressure value reduces suddenly just before the cylinder wall at $X = -1$. This indicates that LDFD-IBM may introduce minor energy loss cross the immersed wall boundary, possibly due to the fact that artificial flow information inside the solid domain is used for the correction of the flow information in the fluid domain during the implementation of boundary condition for the cylinder wall.

The streamlines around the cylinder show that the no-penetration boundary condition is well satisfied for both methods, as no penetration flow is observed. However, the minor difference of the streamlines is noticed near the corners (C_1 and C_2 in the zoom-in view of the streamlines) of the semi-circle and the right domain boundary. In LDFD method, streamlines near the corners follow the cylinder wall boundary closely; while in LDFD-IBM streamlines near the corners deviate from the cylinder wall. This could be a result of the interaction

between the implementation of domain boundary condition and LDFD-IBM on the cylinder wall boundary. Obviously, this interaction is only happened locally near the two corners and it has minimal influence on the capture of shock wave before the cylinder in the mainstream.



(a) LDFD Method: Pressure Contours, Streamlines and Zoom-in view



(b) LDFD-IBM: Pressure Contours, Streamlines and Zoom-in view

Figure 5.4 Comparison of results obtained by LDFD method and LDFD-IBM for Mach 3 flow over a circular cylinder

In comparison of the conventional implementation by ghost cell method or symmetrical method, the new methods of LDFD and LDFD-IBM are very easy to implement. There is no need to calculate the curvature of the immersed wall boundary and to do bilinear interpolation. Besides the ease of implementation, accurate results are obtained for the Mach 3 supersonic flow over a circular cylinder using both methods of LDFD and LDFD-IBM and demonstrate the methods are promising. Hence the methods will be further tested for other compressible flows.

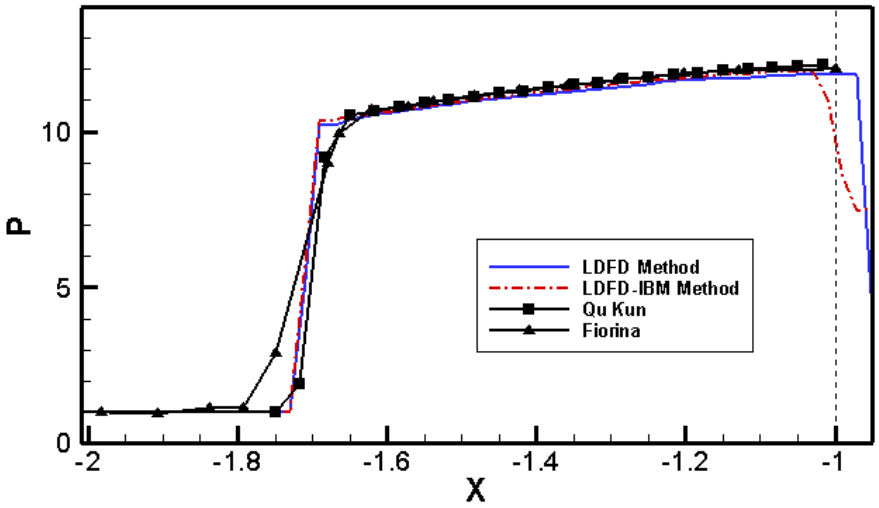


Figure 5.5 Comparison of pressure profile along the central line obtained by LDFD method and LDFD-IBM for Mach 3 flow over a circular cylinder

5.3.2 Supersonic flow over a wedge

The supersonic flow of Mach 2 over a wedge is solved by both LDFD method and LDFD-IBM. The configuration and the boundary condition of the problem are the same as presented in subsection 4.3.1. In order to resolve the wedge wall boundary and the shock waves in the channel domain accurately, 2-level adaption is used. The Mach number contours are plotted in Figure 5.6 for comparison. The wedge shock emitting from the front wedge point and the

two reflection shock waves predicted are almost identical. The angle of the wedge shock predicted by the LDFD method is 45.39°, which is in very good agreement with the theoretical value of 45.38°. The angle of the wedge shock predicted by LDFD-IBM is 46.63°, also in fair agreement with the theoretical value.

The results show that both LDFD method and LDFD-IBM are able to simulate the supersonic flow over a wedge structure accurately.

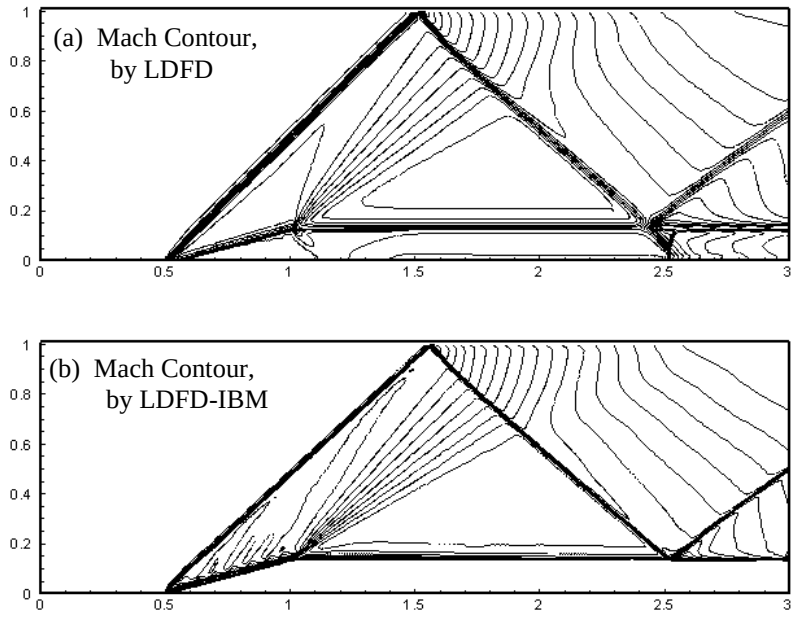


Figure 5.6 Comparison of Mach number contours obtained by LDFD method and LDFD-IBM for Mach 2 flow over a wedge

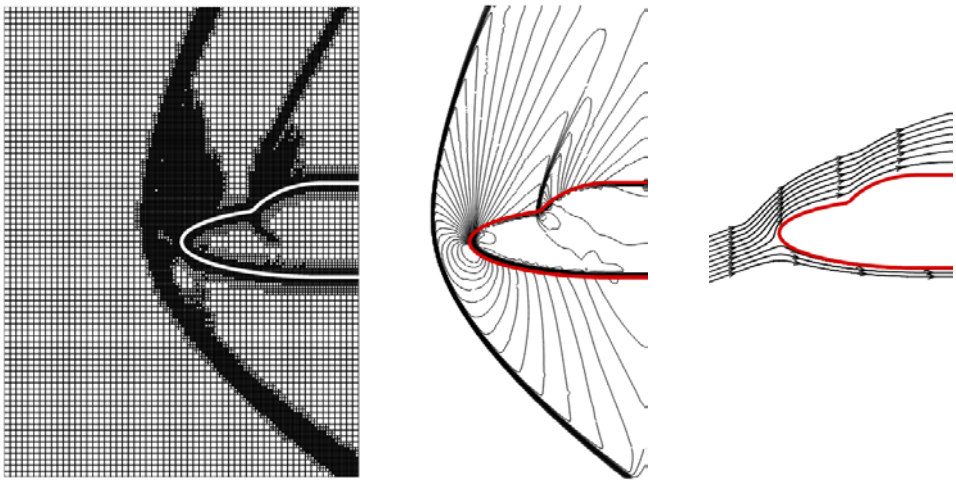
5.3.3 Supersonic flow over a double-ellipse

The supersonic flow past a double-ellipse is solved by the FC-IBM and the results are presented in subsection 4.3.2. The same problem is also solved by the proposed LDFD method and LDFD-IBM. Uniform coarse mesh with size

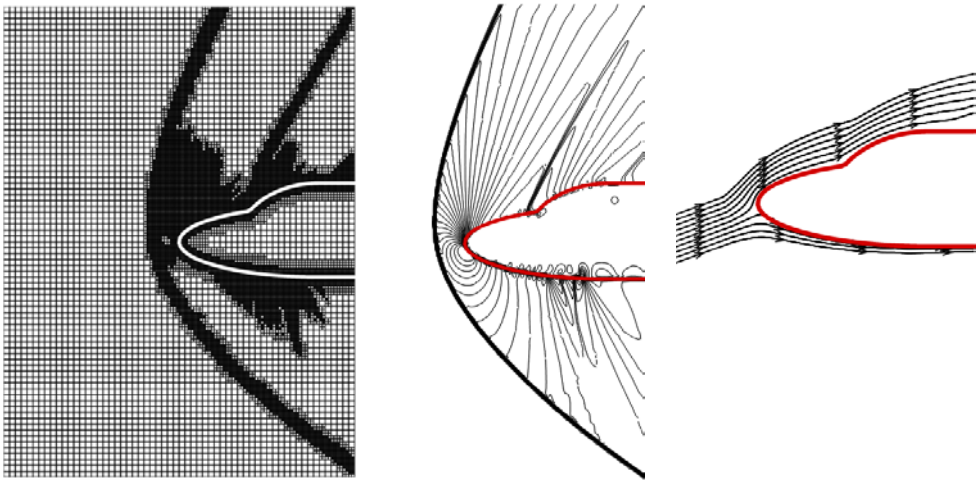
of 1 is defined in the computational domain. Two levels of adaption are used to resolve the double-ellipse wall boundary accurately and also applied to enhance the resolution for capturing the bow shock wave generated in front of the double-ellipse structure. The results of the solution adaptive mesh, pressure contours, streamlines over the double-ellipse and the pressure coefficient profile obtained by both methods are plotted in Figure 5.7 and Figure 5.8. The bow shock wave in front of the double-ellipse structure and the inclined shock omitted from the intersection corner of the two ellipses are captured sharply with fine meshes adapted to the high density gradients in those zones. The plot of streamlines shows that air past the boundary smoothly and indicates that the no-penetration boundary condition is well satisfied. The pressure coefficient profile is compared with the numerical results obtained on body-fitted grid by Arminjon et al. [49] and shows good agreement.

Next, to further test and validate the two methods, a hypersonic flow in the condition of Mach=8.15 and AoA=30° over the double-ellipse is solved. Because of the high Mach number and large angle of attack, the flow tends to be unstable and chaotic when it interacts with the ellipse wall structure. Such numerical simulation requires a robust solver and an appropriate method for implementation of boundary conditions. The results are successfully computed by the current LDFD method only. As the shock wave is very strong and closer to the ellipse structure at high Mach number condition, 3-level adaption is used to improve the mesh resolution between the shock wave and the boundary. The results are presented in Figure 5.9. The solution adaptive mesh in Figure 5.9 (a) shows that only fine meshes can capture the shock wave

sharply, as shown in Figure 5.9 (b). Streamlines plotted in Figure 5.9 (c) show that no flow penetration is observed, implying that the no-penetration boundary condition is also fully satisfied under the hypersonic condition. The pressure coefficient is compared with the numerical data obtained by Zeeuw and Powell [4], Bramkamp et al. [53] and Ganesh et al. [63], as plotted in Figure 5.9 (d). The comparison demonstrates that the current results match very well with those numerical results.



(a) Solution adaptive mesh (2x), pressure contours and streamlines computed by LDFD method.



(b) Solution adaptive mesh (2x), pressure contours and streamlines computed by LDFD-IBM.

Figure 5.7 Comparison of results obtained by LDFD method and LDFD-IBM for flow over a double-ellipse structure (Mach=2, AoA=20°)

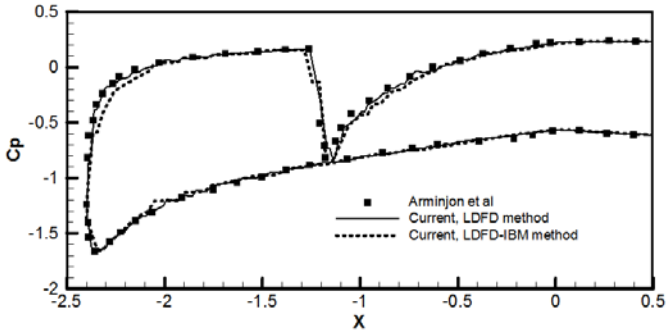
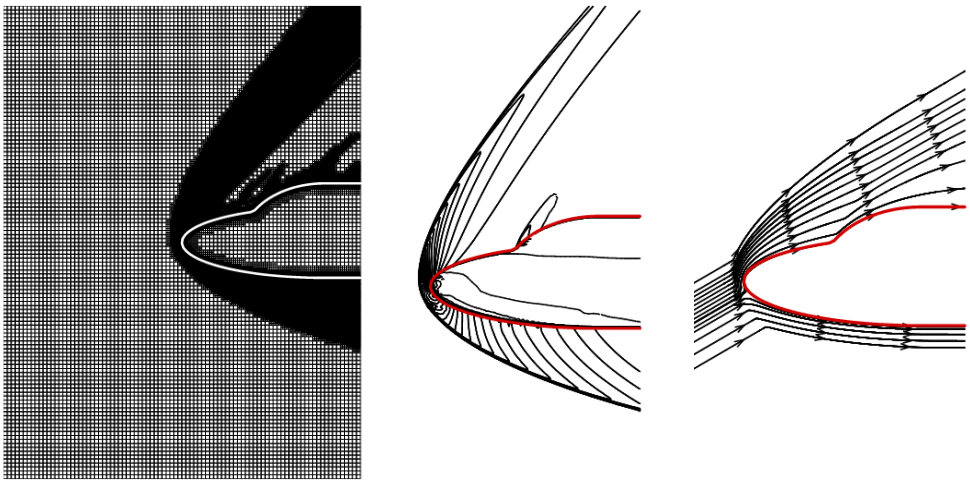
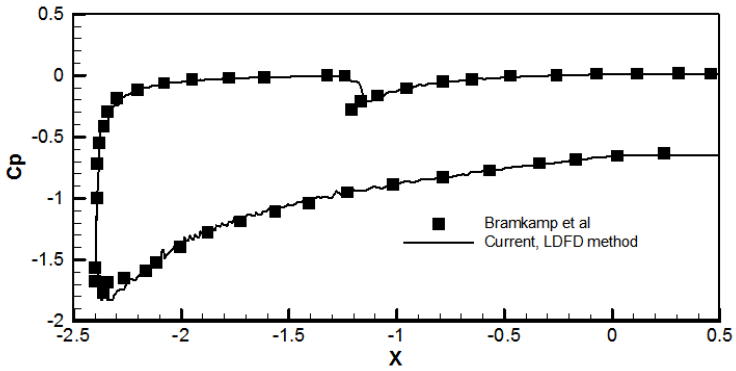


Figure 5.8 Pressure coefficient profile on the boundary for flow over a double-ellipse structure (Mach=2, AoA=20°)



(a) Solution adapted mesh (3x) (b) Pressure contours (c) Streamlines



(d) Pressure coefficient profile on the boundary surface and comparison

Figure 5.9 Computed results by LDFD method for hypersonic flow over a double-ellipse structure (Mach=8.15, AoA=30°)

5.3.4 Transonic flow in a channel with bump

The literal meaning of immersed boundary implies that the boundary if fully submerged inside the computational domain. For the GAMM channel case, though the wall boundary represented by the 10% arc (as shown in Figure 3.6) is immersed inside the 3×1 rectangular Cartesian domain, the two ends of the boundary connect with the physical boundary of the domain. The conventional immersed boundary methods are not suitable to be implemented for such cases with the immersed wall boundary intersected over the physical boundary. While the current LDFD method and LDFD-IBM can still be implemented theoretically for this problem, as the corresponding DFD cells in X direction and Y direction can be identified as described in Sections 5.1 and 5.2.

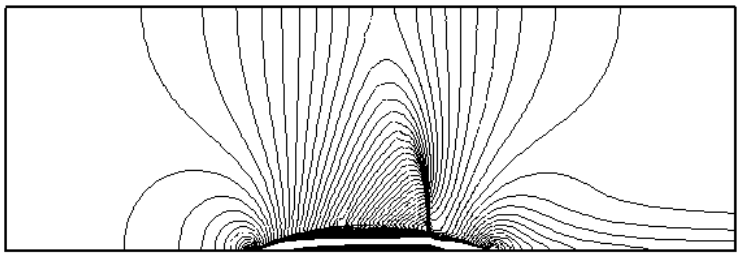
As the bump in the GAMM channel is relative thin, to be efficient the uniform coarse mesh is defined in the rectangular domain and 4 levels of mesh refinement are adapted to the bump wall boundary and 2 levels of solution adaption are used in the domain to enhance the mesh resolution for the predication of the shock wave presented after the throat. The Mach number contours predicted by the two methods are plotted in Figure 5.10 (a) and (b), respectively. It is noticed that the Mach number distribution computed via LDFD method is comparable to other numerical results obtained by Morton and Paisley [54], Luo et al. [45] and Lee et al. [50]. However, the Mach number distribution computed via LDFD-IBM is different compared to those numerical results and not reasonable. The streamlines plotted in Figure 5.10 (d) show that recirculation flow pattern is observed behind the channel throat, although no streamlines are noticed cross the bottom boundary of the channel.

This means that the no-penetration boundary condition is satisfied. However, the artificial viscosity introduced by LDFD-IBM implementation has caused unphysical phenomena. The computed Mach number profile along the bottom channel wall via LDFD method is plotted in Figure 5.10 (e) and is compared with the numerical results obtained on body-fitted grid by Luo et al. [45]. The square solids represent Luo's result, and the solid line is the current result. The dashed line is the pressure profile along the bottom channel wall. It is noted from the plot that the maximal Mach number and the shock wave position are in good agreement.

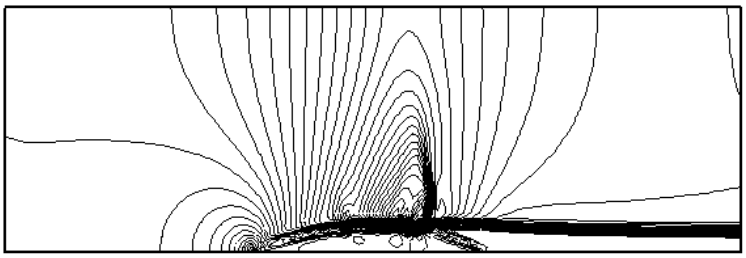
In LDFD-IBM, the flow variables at cells on both sides of the boundary will be corrected in order to enforce the boundary condition. Hence artificial viscous effect will be introduced near the boundary, as analyzed in Section 4.3.3. As the pressure profile in Figure 5.10 (e) shows that low reverse pressure gradient is experienced in the region behind the throat, the unphysical flow phenomena will be induced as the contribution from the numerical viscous effect to the numerical solution increases.

The study of this test case demonstrates that the robustness of the LDFD method is better than the LDFD-IBM, in particular, the LDFD method can give accurate solution when low reverse pressure gradient is experienced near the boundary.

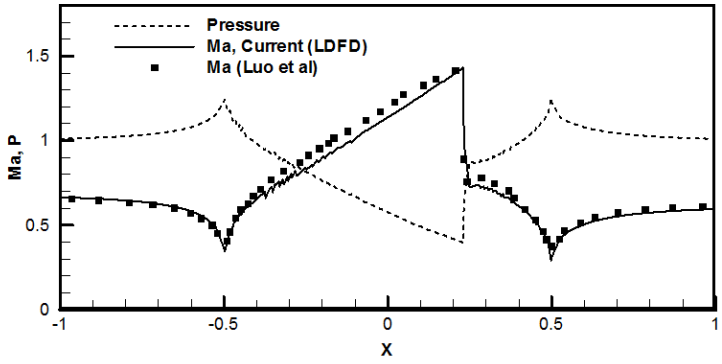
(a)
Mach Contour,
by LDFD



(b)
Mach Contour,
by LDFD-IBM



(c) Streamlines, by LDFD (d) Streamlines, by LDFD-IBM



(e) Mach number profile on the lower wall of the channel, by LDFD

Figure 5.10 Comparison of results obtained by LDFD method and LDFD-IBM for transonic flow in GAMM channel

5.4 More numerical examples and discussions

From the validation and comparison for the LDFD method and LDFD-IBM in the previous section, it shows that LDFD method is more robust and is able to predict more accurate solution for compressible inviscid flows under various complex conditions. Thus, more numerical studies will be carried out by the LDFD method to test the robustness and performance of the LDFD method integrated with the current adaptive solver. It should be noted that the basic concept for the implementation of the LDFD method and the LDFD-IBM is similar as the boundary conditions are enforced in X and Y direction instead of in normal condition. The major difference is the that boundary conditions are enforced by correcting the functional values at the solid DFD cells from the fluid DFD cells for LDFD method; while for LDFD-IBM, the boundary conditions are enforced by mirrored correction between the DFD cells on both sides of the boundary.

5.4.1 Transonic flow over a NACA0012 airfoil

The transonic flow over a NACA0012 airfoil at $M_\infty = 0.799$ and the angle of attack at 2.8° is computed by the LDFD method. As the airfoil has thin geometry near the trailing edge, the local DFD implementation needs to consider the special cases similar as the implementation of ghost-cell method as presented in subsection 3.2.3 and Figure 3.8 (a) for a RAE2822 airfoil. Just to re-cap that when a solid DFD cell has more than one fluid DFD cells in the X direction or in the Y direction, two sets of values are stored and linked to the corresponding boundary. As shown in Figure 5.11, one set of functional

values at solid DFD cell (S) can be corrected from the fluid DFD cell (S_1) opposite the boundary $\overline{AC}$; and the other set of functional values at (S) can be corrected from the fluid DFD cell (S_2) opposite the boundary $\overline{BC}$.

The coarse uniform mesh defined in the domain is at the size of 2. Finer meshes are clustered around the airfoil boundary by 6-level refinement. The solution is adapted by 5-level refinement, or at the mesh size of 1/16, to improve the resolution economically in the regions near the shock wave and high flow gradient zones near the airfoil head. The computed Mach number contours, streamlines, pressure coefficient profile on the airfoil surface and the solution adaptive mesh are plotted in Figure 5.12. The computed pressure coefficient profile is compared with the experimental data and numerical results obtained on body-fitted triangular mesh by Liu and Li [55]. The pressure coefficient profile agrees relatively well with the numerical data obtained by Liu and Li except slightly difference on the shock wave position. Nevertheless, the shock wave prediction is a bit off from the experimental data.

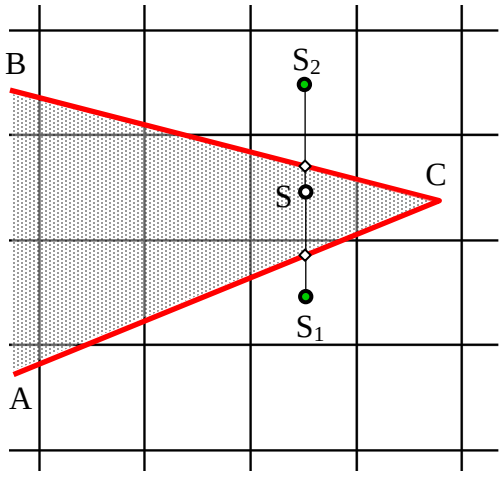
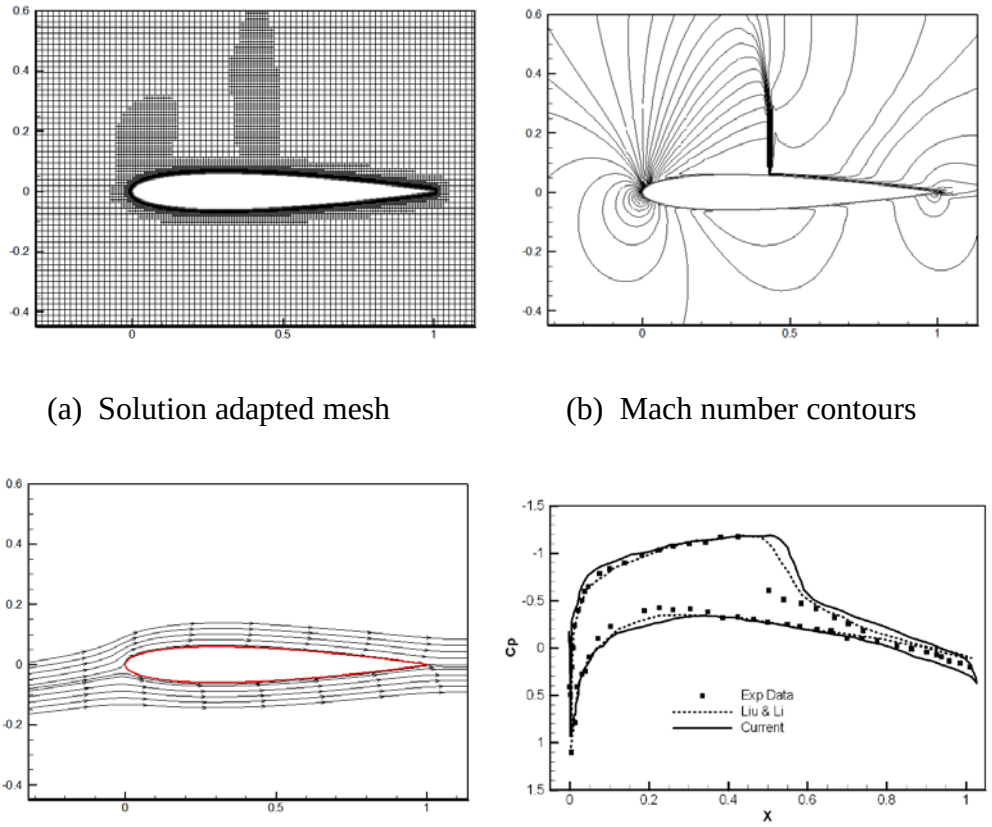


Figure 5.11 Special case for LDFD method near the thin boundary region



(a) Solution adapted mesh (b) Mach number contours

(c) Streamlines around the airfoil (d) Pressure coefficient profile

Figure 5.12 Numerical results for transonic flow over a NACA0012 airfoil ($M_\infty = 0.799, \quad AoA = 2.8°$)

5.4.2 High lift two-element airfoil - NLR 7301

The two-element airfoil NLR 7301 can generate high lift force at large angle of attack without causing airfoil stall. This behavior is commonly used in passenger aircrafts during takeoff or landing when aircrafts fly at low speed and large angle of attack. The airfoil is also used in racing cars to generate down-force which is important in enhancing the car steering at high speed [56]. The configuration of the main airfoil and the flap is found in the AGARD-AR-303 report [57]. The chord of the flap is 32% of the main airfoil chord; the flap is deflected at 20°; and the gap between the main airfoil and the flap is 2.6% of the main airfoil chord. The airfoil geometry is shown in Figure 5.13. The

numerical experiment is done at the condition of the free-stream flow is at Mach number 0.185 under angle of attack at 6° and 13.1°. This test case is chosen to demonstrate the robustness and efficiency of the current LDFD method and the adaptive solver.

Due to the sharp and thin geometry of the flap and the small gap between the main airfoil and the flap, fine meshes with 7-level adaption are clustered around the airfoils based on the coarse uniform mesh at $dh = 2$. The meshes around the modeled NLR 7301 airfoil and the zoom-in view of the meshes in the gap are plotted in Figure 5.14. The computed results for AoA=6° and 13.1° are plotted as Mach number contours, streamlines and pressure coefficient profile in Figure 5.15 and Figure 5.16, respectively. The Mach number contours are compared to the numerical results obtained by Liang and Yang [12]. The streamlines show that no-penetration boundary condition is satisfied for both airfoils. The pressure coefficient profile computed on the main airfoil and the flap for both conditions agrees closely with the experimental data published online [56], as shown in Figure 5.15 (c) and Figure 5.16 (c). The converged solution is obtained on 60,379 mesh cells, with approximate 82% of cells are adapted in the local region near the airfoil geometry and the gap.

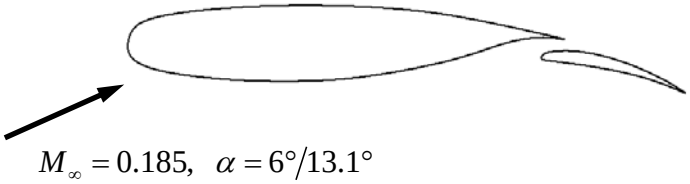


Figure 5.13 Configuration of NLR 7301 two-element airfoil

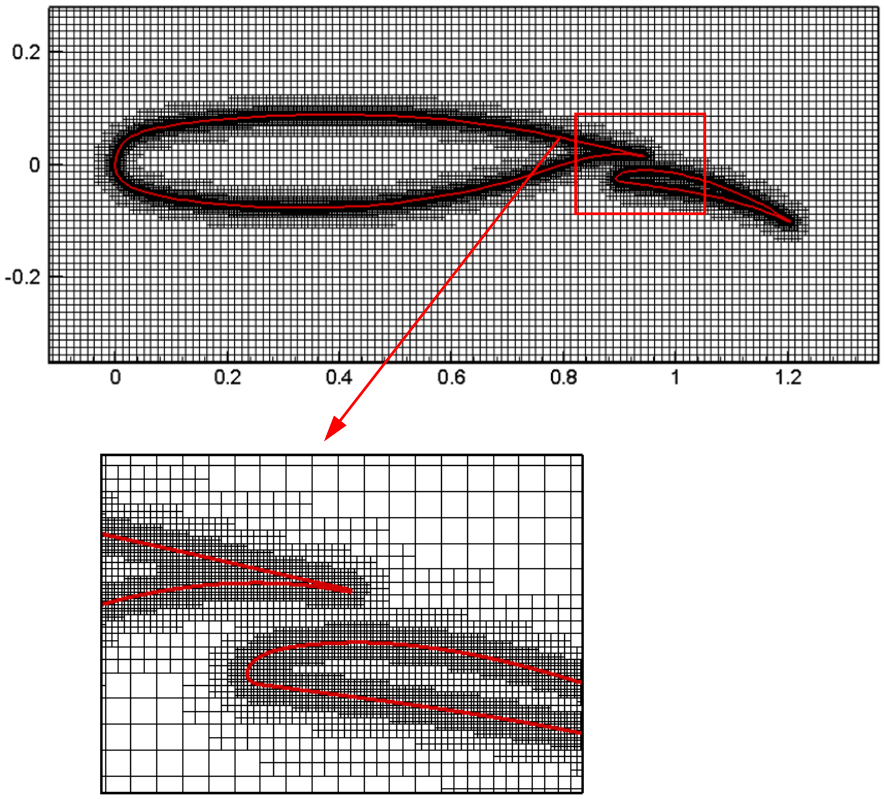
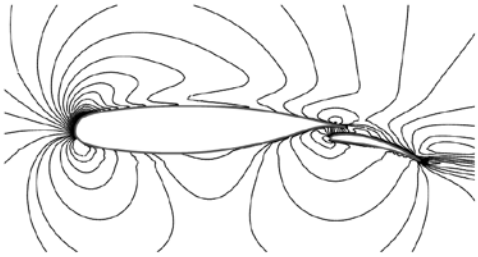
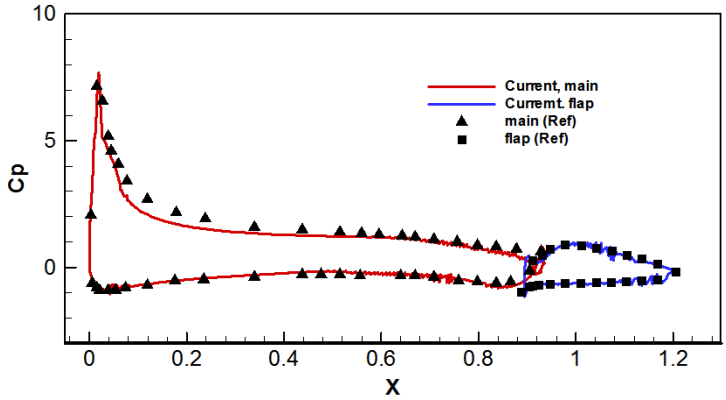


Figure 5.14 Initial mesh adaption near the NLR 7301 two-element airfoil



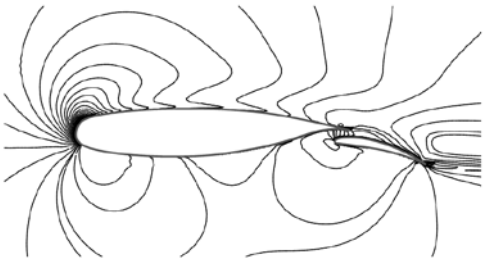
(a) Mach number contours

(b) Streamlines

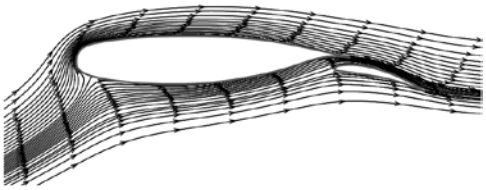


(c) Pressure coefficient profile and comparison

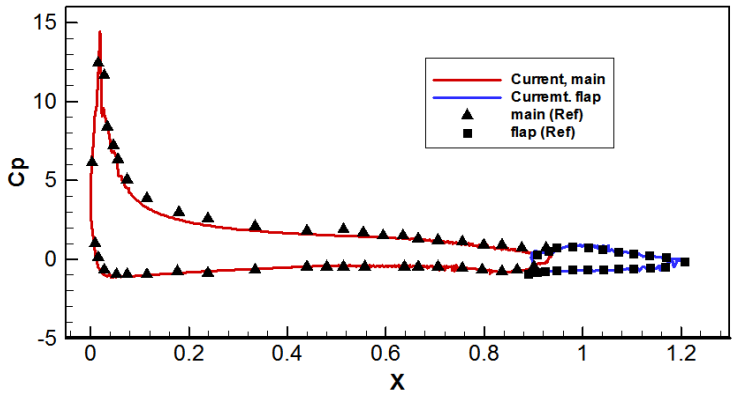
Figure 5.15 Numerical results for NLR7301 ($M_\infty = 0.185, \quad AoA = 6°$)



(a) Mach number contours



(b) Streamlines



(c) Pressure coefficient profile and comparison

Figure 5.16 Numerical results for NLR7301 ($M_\infty = 0.185, \quad AoA = 13.1°$)

5.4.3 Transonic flow over SKF1.1 two-element airfoil

The study of the NLR 7301 two-element airfoil is under subsonic flow condition. In this subsection, the transonic flow over another two-element airfoil SKF1.1 is simulated to demonstrate the performance of the LDFD method. The flow condition over the SKF1.1 airfoil is defined with the free-stream Mach number at 0.65 and the angle of attack at 2.06°. Under this condition, a shock wave will be generated above the main airfoil. Therefore, this test case is more challenging than the previous numerical experiment of the subsonic flow on the NLR 7301 two-element airfoil.

The configuration of the main airfoil and the flap of SKF1.1 airfoil used for this test case is the configuration case 5 documented in the AGARD-AR-138 report [58]. The chord of the flap is $1/4$ of the main airfoil chord; the flap is deflected at 10°; and the gap between the main airfoil and the flap is only 1.55% of the main airfoil chord. The airfoil geometry is shown in Figure 5.17. Finest meshes with 7-level adaption are clustered around the airfoils based on the coarse uniform mesh at $dh = 2$. This allows for sufficient Cartesian mesh cells generated in the gap region, thus the boundary condition can be implemented accurately. The meshes around the studied SKF1.1 airfoil and the zoom-in view of the meshes in the gap are plotted in Figure 5.18.

The computed results of pressure contours, solution adaptive meshes, streamlines and pressure coefficient profile are presented in Figure 5.19. The pressure contour distribution, plotted in Figure 5.19 (a) is agreed with the numerical results obtained by Jahangirian and Hashemi [13]. The streamlines

plotted in Figure 5.19 (c) and (d) show that no-penetration boundary condition is satisfied for both airfoils. The converged solution is obtained on 61,072 cells, with approximate 82% of cells are adapted in the local region near the airfoil geometry, the gap and the scene of the shock wave via solution adaption. The pressure coefficient profile computed on the main airfoil and the flap is compared with the numerical data computed by Jahangirian and Hashemi [13] using an unstructured body-fitted grid solver. As shown in Figure 5.19 (e), good agreement is achieved not only in the magnitude of the pressure coefficient computed on both airfoils, but also in the position of the shock wave above the upper surface of the main airfoil.

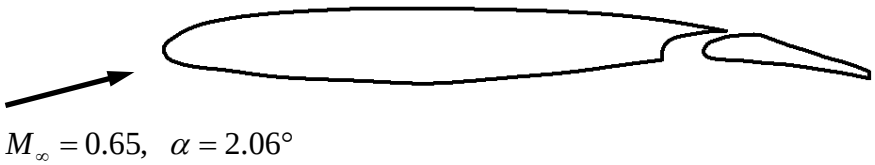


Figure 5.17 Configuration of SKF1.1 two-element airfoil

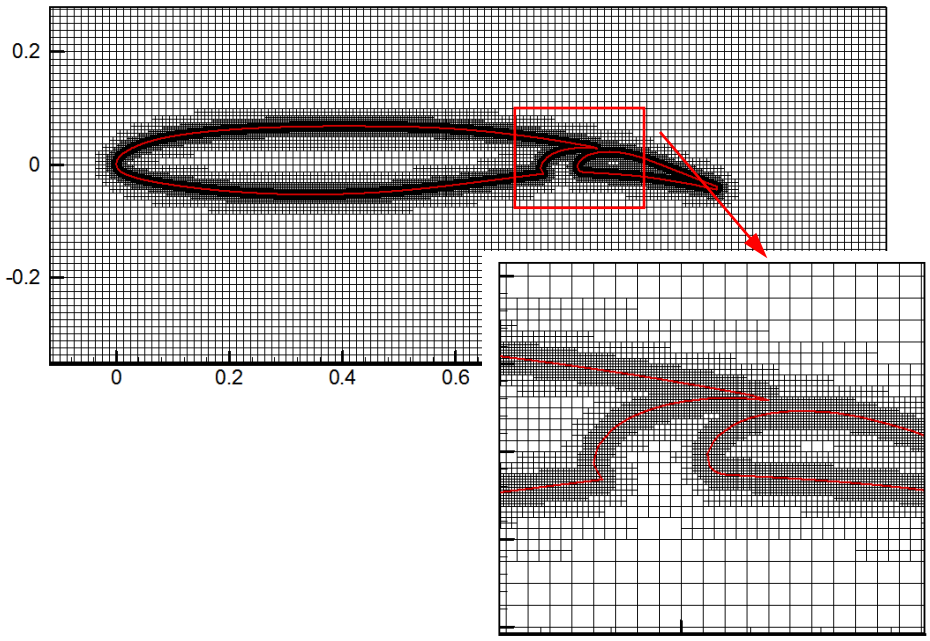
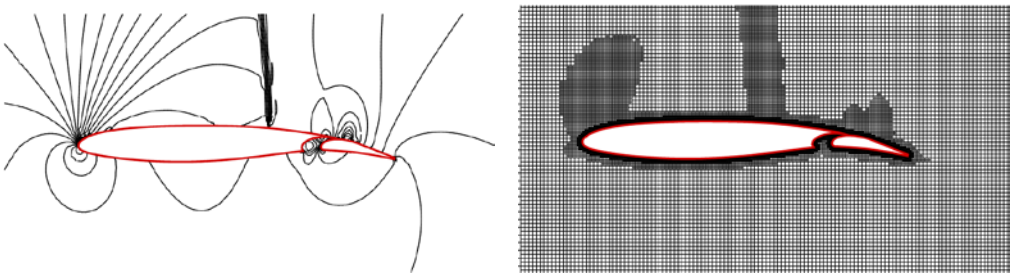
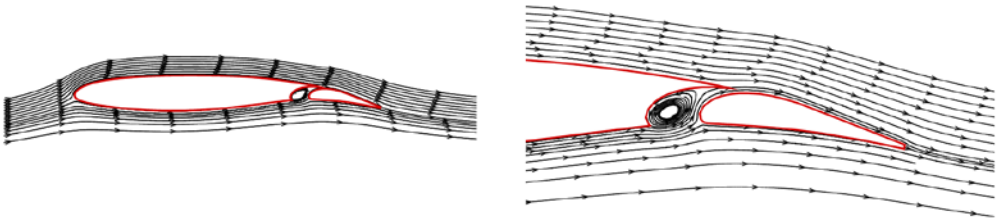


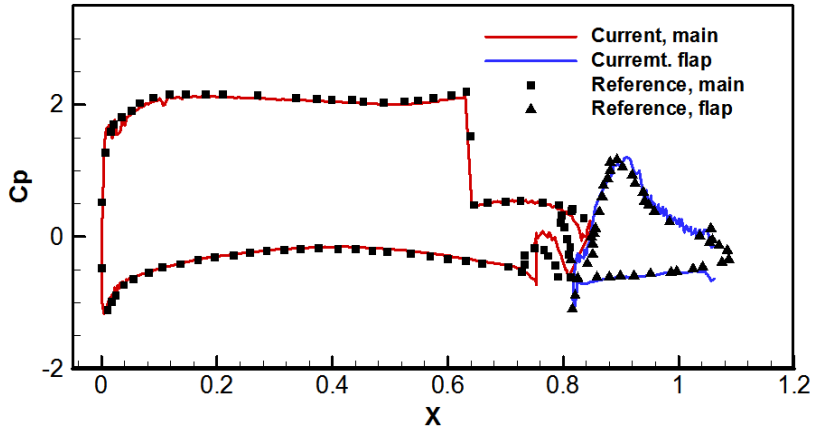
Figure 5.18 Initial mesh adaption near the SKF1.1 two-element airfoil



(a) Pressure contours (b) Solution adapted mesh



(c) Streamlines over SKF1.1 airfoil (d) Zoom-in view near the gap and flap



(e) Pressure coefficient profile and comparison

Figure 5.19 Numerical results for SKF1.1 airfoil ($M_\infty = 0.65,\ \alpha = 2.06°$)

5.5 Conclusions

A local domain-free discretization (LDFD) method is presented in this chapter for compressible inviscid flows. The concept of LDFD for 2D flows is to enforce the boundary condition by correcting the flow information on the solid cells (DFD cells) next to the boundary in X direction and Y direction separately. As the correction is performed either horizontally or vertically, it is simple to implement. A LDFD immersed boundary method (LDFD-IBM) is also presented to make the implementation simpler by avoiding the need to identify the solid DFD cells and fluid DFD cells.

Both methods are implemented in the developed adaptive Euler solver. To validate and compare the two methods, supersonic flows over a circular cylinder and a wedge, hypersonic and supersonic flows over a double-ellipse structure, transonic flow in a channel with bump are simulated. The results obtained from the two methods are comparable and agree well with the available data in the literature. The LDFD method is demonstrated to be more robust and accurate. The challenging problems like the subsonic flow over the high lift two-element airfoil NLR 7301 and the transonic flow over SKF1.1 two-element airfoil are studied by the LDFD method. The computed results are in good agreement with the reference data. The integration of the LDFD method and the developed adaptive Euler solver demonstrates good potential to simulate compressible inviscid flows with complex geometry.

Chapter 6 3D Adaptive Euler Solver Implemented with FC-IBM and LDFD Method

The 2D adaptive Euler solver is discussed and benchmarked in Chapter 2. The proposed new FC-IBM, LDFD method and LDFD-IBM are also implemented in the 2D adaptive solver successfully. Validations and numerical studies demonstrate that the methods are good potential tools to simulate compressible inviscid flows with complex boundaries using the present 2D adaptive Euler solver on Cartesian grids. In this chapter, the development of 3D adaptive Euler solver and the implementation of the proposed immersed boundary methods are discussed.

6.1 Methodology for 3D adaptive Euler solver

The three-dimensional compressible inviscid solver is developed on the basis of the two-dimensional solver. The 3D compressible Euler equations in conservative form are given as:

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}+\frac{\partial H}{\partial z}=0, \tag{6.1}$$

$$U=\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ pw\\ E\end{bmatrix},\quad F=\begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ u\left(E+p\right)\end{bmatrix},\quad G=\begin{bmatrix}\rho v\\ \rho vu\\ \rho v^2+p\\ \rho vw\\ v\left(E+p\right)\end{bmatrix},\quad H=\begin{bmatrix}\rho w\\ \rho uw\\ \rho vw\\ \rho w^2+p\\ w\left(E+p\right)\end{bmatrix}. \tag{6.2}$$

The equation of the state for ideal gas in 3D flow is:

$$p = (\gamma - 1)\left[E - \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right) \right]. \tag{6.3}$$

The numerical method, flux calculation, and boundary conditions for 3D solver are similar to those for 2D solver. The control cells for 3D solver are hexahedral cells in the Cartesian coordinate system. A hexahedral cell consists of six surface interfaces, twelve edges and eight end points. The flux calculation is performed on six surfaces of a hexahedral control cell, and the solution is stored at the cell center of the hexahedral cell. The data structure for the objects of 3D control cell, side surfaces, edges and end nodes are defined as below in Fig. 6.1.

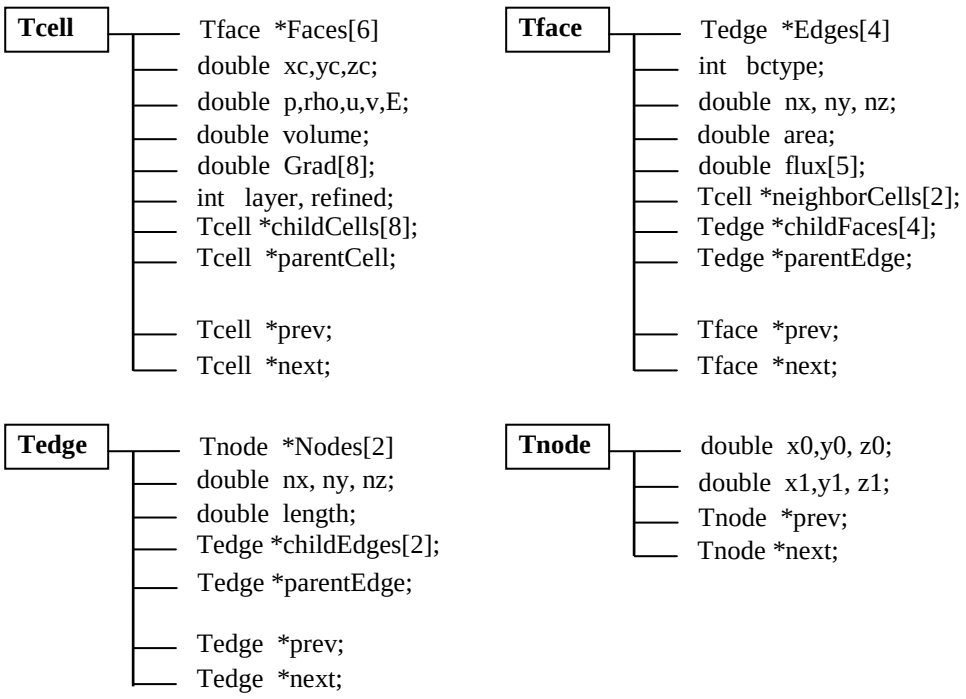


Figure 6.1 Data structure for the objects of cell, face, edge and node for 3D solver

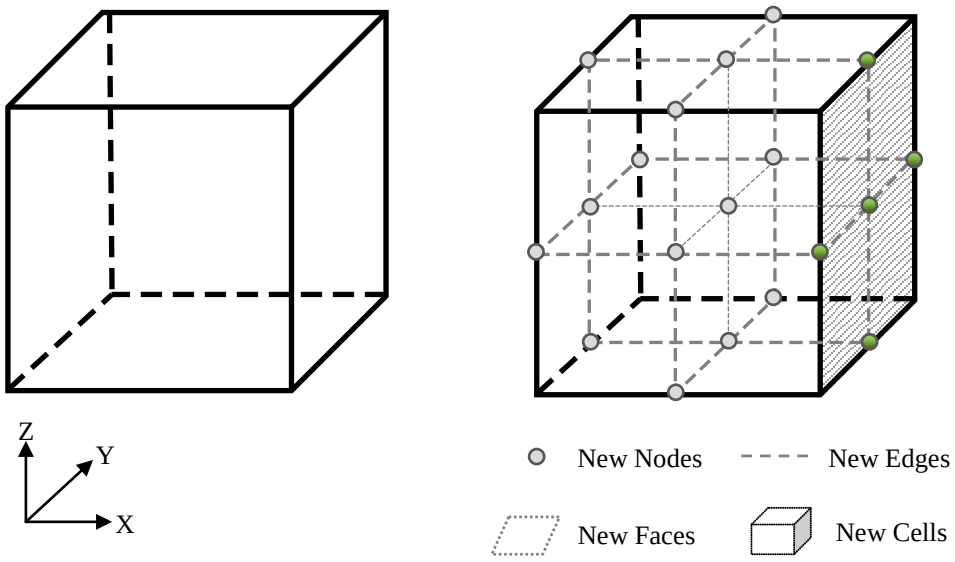


Figure 6.2 Refinement of a 3D Cartesian cell

The adaption of 3D hexahedral cells follows the tree structure as that implemented in the 2D adaptive solver. When a 3D cell is to be refined, it will be split into eight children cells, as shown in Figure 6.2. To form the eight children cells, there will be 19 new nodes, 24 children edges, 22 new edges, 24 children faces and 8 new faces created for this parent cell to be refined. Since there are so many new objects (nodes, edges, faces and cells) to be created in refining a 3D cell, the creation and recording of those new objects and their relationships must be handled carefully, otherwise the adaption will not be able to perform properly. The adaption state for faces and edges of a cell must be recorded too, because this will be relevant to determine the adaption for the neighbor cell. In the condition when any of the six neighbor cells next to the six faces is adapted, the interface between the two cells such as the shaded +X face in Figure 6.2 does not need to be refined again. The associated five new nodes, four new edges, eight new children edges and four new children faces will be re-used by setting the correction relationship. During the coarsening

adaption, the eight children cells, the six edges inside the cell and the node located in the center of the cell can be removed in a straightforward manner. To remove the children faces on the six side faces of the cell and the associated edges and nodes, the dependency has to be verified on the adaption state of the neighbor cells. If the neighbor cell is not refined, then the children faces, edges and nodes on this interface can be removed.

6.2 FC-IBM and LDFD implementation in 3D solver

The concept of IBM implementation in 3D solver is similar to that for the 2D implementation. The key difference is the definition of the immersed wall boundary. To be specific, for 2D implementation, the immersed wall boundary is actually 2D edge and is defined by line segments; while for 3D implementation, the immersed wall boundary is 3D surface and is represented by multiple small surface patches. The 3D boundary surface can be defined by surface meshes, either with triangles or quads. Since a 3D surface can be fully determined by three points in the domain, the triangular meshes are used to represent the 3D boundaries in the present study. ANSYS's Gambit meshing tool is used to generate the triangular surface meshes that represent the wall boundary.

To implement FC-IBM in 3D solver, the zero normal flux conditions are enforced by correcting the conservative terms U(1, 5) in equation (6.2) and the no-penetration condition is enforced by correcting velocity components (u, v, w) in equation (6.2). Because the wall boundary is formed by triangular

bounded surfaces, the normal velocity of a triangular surface at its center needs to be evaluated in order to conduct the flux correction and velocity correction. The delta-function interpolation is used to compute the functional values on each triangular surface. The shadow cells around the center of the triangular surface within double mesh spacing are identified and used for both flux and velocity corrections. There is no need to compute the boundary curvature and to identify whether the cells are in fluid domain or solid domain. The implementation procedure is similar to that for the implementation of 2D case, as discussed in subsections 4.1.1 and 4.1.2. However, it shall be noted that the normal velocity correction on the boundary is re-distributed back to shadow cells to correct the velocity components in X, Y and Z directions for 3D cases.

In 2D implementation of LDFD method, the relationship between DFD cells and the wall boundary needs to be identified. This can be done easily through the checking of intersection status of two line segments, one representing the wall boundary segment and the other representing the line segment linking two cell centers. When 3D implementation is considered, this process becomes a bit complex and tedious because the relationship between cells and boundary is determined by the intersection status of a line segment and a bounded triangular surface. In addition, the velocity transformation between Cartesian coordinate system and the normal-tangential coordinate system in 2D case is obvious and simple; in 3D case, the velocity transformation process is more complex and requires more care in implementation.

The details on the definition of wall boundary and its normal direction, identification of local DFD cells, velocity transformation on wall boundary are explained in the following sections, respectively.

6.2.1 Wall boundary surface and normal direction

The immersed wall boundary in 2D solver is formed by edges, which are defined as line segments. For 3D solver, the immersed wall boundary is formed by faces, which are usually defined by triangular faces or quadrangular faces. In the current development, the immersed wall boundary is defined by triangular faces. Figure 6.3 shows an immersed spherical wall boundary which is formed by many triangular faces. Each of the triangular faces is formed by a triangle ΔABC. In 3D space, the triangular face ΔABC is given as

$$f = ax + by + cz + d = 0. \tag{6.4}$$

As the coordinates of the three endpoints A, B and C are known, so ***a***, ***b***, ***c*** and ***d*** in the equation can be determined by

$$\begin{aligned} a &= \begin{vmatrix} 1 & y_A & z_A \\ 1 & y_B & z_B \\ 1 & y_C & z_C \end{vmatrix}, \qquad b = \begin{vmatrix} x_A & 1 & z_A \\ x_B & 1 & z_B \\ x_C & 1 & z_C \end{vmatrix}, \\ c &= \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix}, \quad d = -\begin{vmatrix} x_A & y_A & z_A \\ x_B & y_B & z_B \\ x_C & y_C & z_C \end{vmatrix}. \end{aligned} \tag{6.5}$$

Once ***a***, ***b***, ***c*** and ***d*** are determined, the normal vector of the triangular face is known as

$$\vec{n} = \left[n_x, \quad n_y, \quad n_z \right] = \left[a, \quad b, \quad c \right]. \tag{6.6}$$

To normalize the normal vector, a radial length *r* is introduced.

$$r = \sqrt{a^2 + b^2 + c^2}. \tag{6.7}$$

And then the normal vector is normalized as

$$\vec{n} = \left[\hat{n}_x, \quad \hat{n}_y, \quad \hat{n}_z \right] = \left[a/r, \quad b/r, \quad c/r \right]. \tag{6.8}$$

For the convenience in description, the normalized normal vector is still written in the format of $\left[n_x, \quad n_y, \quad n_z \right]$ in the remaining of this section.

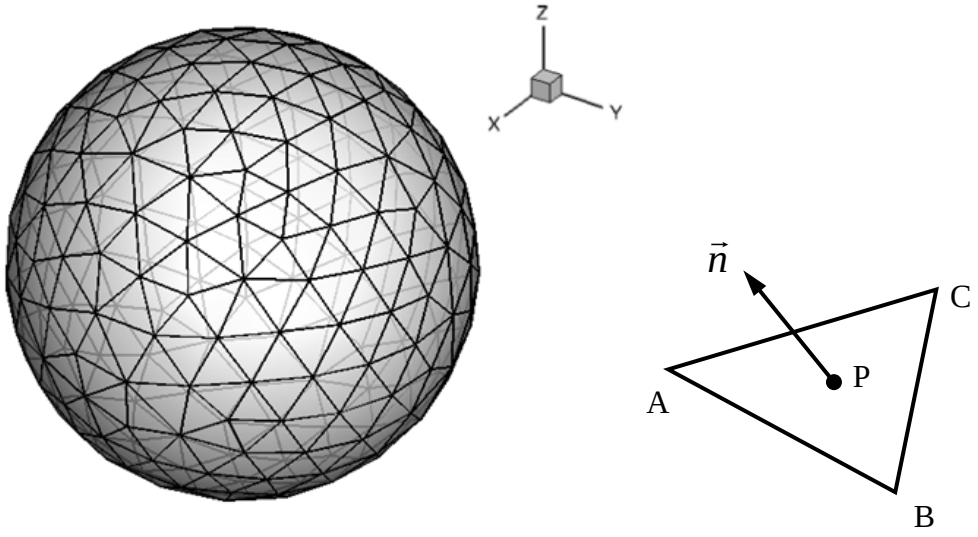
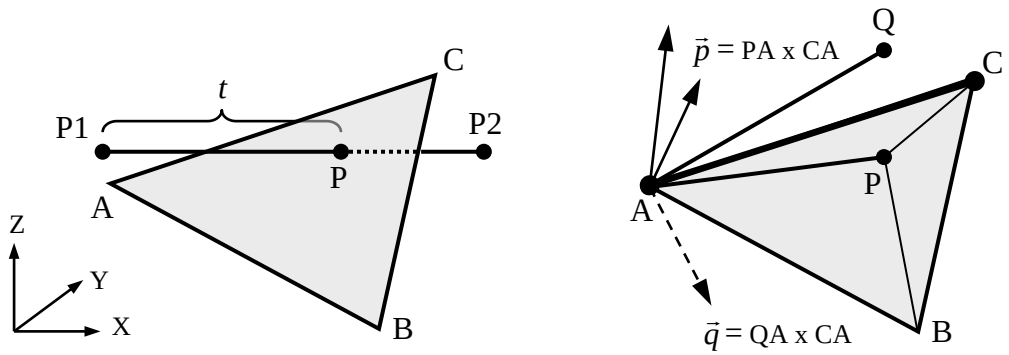


Figure 6.3 3D wall boundary surface and basic triangular face

6.2.2 Identification of local DFD cells

As the LDFD method presented in the previous chapter demonstrates better accuracy in dealing with the immersed wall boundary condition, this method is implemented in the 3D solver for further study. To implement the LDFD

method, the cells near the wall boundary have to be identified first. As a local DFD cell next to the wall boundary definitely has a neighbor cell on the opposite side of the wall boundary, so the major task to identify the pair of the local DFD cells next to the wall boundary is to determine whether the line segment formed by the two cell centers intersects with any bounded triangular face of the wall boundary.



(a) Point P is between P1 and P2. (b) Point P is inside ΔABC.

Figure 6.4 Intersection relationship between a triangular face and a line segment

Taking the X direction for example, let points P1 and P2 in Figure 6.4 (a) represent the centers of the two neighbor cells in X direction, ΔABC represents a triangular face of the wall boundary. To determine whether the line segment $P1 \rightarrow P2$ intersects with the bounded triangular face ΔABC, the procedures are as below.

1) Check if the line determined by $P1 \rightarrow P2$ intersects with the unbounded face on ΔABC,
2) Check if the intersection point ***P*** is on the line segment $P1 \rightarrow P2$, and find out the point,
3) Check if the intersection point ***P*** is inside ΔABC.

To start the determination, first it is needed to check if line determined by $P1 \rightarrow P2$ intersects with the unbound face on ΔABC. Assume that they intersect at point ***P***, then the coordinates of point $P(x, y, z)$ can be given as

$$P(x, y, z) = P1(x, y, z) + t \cdot \left[P2(x, y, z) - P1(x, y, z) \right]. \tag{6.9}$$

The above equation can be rewritten as

$$\begin{aligned} x &= x_1 + t(x_2 - x_1), \\ y &= y_1 + t(y_2 - y_1), \\ z &= z_1 + t(z_2 - z_1). \end{aligned} \tag{6.10}$$

In addition, as point ***P*** falls on the plane determined by the triangular face ΔABC, so the coordinate ***P***(x, y, z) must satisfy the 3D plane equation (6.4) derived on face ΔABC.

$$a\left[x_1 + t \cdot (x_2 - x_1) \right] + b\left[y_1 + t \cdot (y_2 - y_1) \right] + c\left[z_1 + t \cdot (z_2 - z_1) \right] + d = 0. \tag{6.11}$$

The factor *t* is thus obtained by

$$t = -\frac{ax_1 + by_1 + cz_1 + d}{a(x_2 - x_1) + b(y_2 - y_1) + c(z_2 - z_1)}. \tag{6.12}$$

Only when the factor *t* is in the range of 0 to 1, the intersection point ***P*** falls between P1 and P2. When $t \rightarrow 0$ the point ***P*** is closer to P1; and when $t \rightarrow 1$ the point ***P*** is closer to P2.

Next step is to verify if the intersection point ***P*** is inside the triangular face ΔABC. As shown in Figure 6.4 (b), if point ***P*** is inside the triangular face

ΔABC the following relationship always stands truly.

$$S_{\Delta APB} + S_{\Delta BPC} + S_{\Delta APC} = S_{\Delta ABC}. \tag{6.13}$$

The S_{Δ} in the equation represents the area of the triangle. Let ***a***, ***b***, and ***c*** represents the length of the three sides of a triangle, the area of the triangle can be calculated using the Heron's Formula:

$$S_{\Delta} = \frac{1}{4}\sqrt{\left(a^2+b^2+c^2\right)^2 - 2\left(a^4+b^4+c^4\right)}. \tag{6.14}$$

Theoretically, the judgment method is obvious and easy in implementation. However, numerical experiments show that it may give wrong results due to the numerical errors which are possibly caused by the power and square root calculation in equation (6.14). In addition, it is tricky to verify the equal relationship for two floating point numbers in computer platform and this may also introduce unexpected errors.

Another way to verify whether the intersection point ***P*** is inside the triangular face ΔABC is same-sided phenomena. As shown in Figure 6.4 (b), if point ***P*** is inside the triangular face ΔABC, then when walking from any of the three endpoints to another endpoint, point ***P*** is always at the same side of the third endpoint of the triangle. For example, when walking from A→C direction, both points ***P*** and ***B*** are at the right side of the edge $\overrightarrow{AC}$ as point ***P*** is inside the triangle; but point ***Q*** is at the left side of the edge $\overrightarrow{AC}$ as point ***Q*** is outside the triangle. To verify this, the following two cross products are calculated first

$$\begin{aligned}\vec{b} &= \overrightarrow{AB} \times \overrightarrow{AC}, \\ \vec{p} &= \overrightarrow{AP} \times \overrightarrow{AC}.\end{aligned} \tag{6.15}$$

If points ***P*** and ***B*** are at the same side when looking from A→C, then the dot product of $\vec{b}$ and $\vec{p}$ should be positive or no-negative, as the internal angle between the two vectors is not more than 90-degree.

$$\vec{b} \cdot \vec{p} = \left(\overrightarrow{AB} \times \overrightarrow{AC}\right) \cdot \left(\overrightarrow{AP} \times \overrightarrow{AC}\right) \geq 0. \tag{6.16}$$

Otherwise, as point ***Q***, which is outside the triangle face ΔABC, shown in the figure, the dot product of $\vec{b}$ and $\vec{q}$ will be negative, as the internal angle between the two vectors is more than 90-degree.

$$\vec{b} \cdot \vec{q} = \left(\overrightarrow{AB} \times \overrightarrow{AC}\right) \cdot \left(\overrightarrow{AQ} \times \overrightarrow{AC}\right) < 0. \tag{6.17}$$

Similarly, point ***P*** must also satisfy the following two conditions for edges $\overrightarrow{CB}$ and $\overrightarrow{BA}$, then it can be confirmed that the point ***P*** is inside the triangle face ΔABC.

$$\begin{aligned}\left(\overrightarrow{CA} \times \overrightarrow{CB}\right) \cdot \left(\overrightarrow{CP} \times \overrightarrow{CB}\right) \geq 0, \\ \left(\overrightarrow{BC} \times \overrightarrow{BA}\right) \cdot \left(\overrightarrow{BP} \times \overrightarrow{BA}\right) \geq 0.\end{aligned} \tag{6.18}$$

If any of the conditions in equations (6.16) and (6.18) is not satisfied, then point ***P*** will be marked outside the triangle face ΔABC.

As described above, this method uses the cross product and dot product for vectors in the verification and avoids the need for the calculation of power,

square root and equality checking of two floating point numbers, so the intersection relationship between the line segment P1-P2 and the triangular face ΔABC can be determined accurately.

Because the relationship of cells and the wall boundary is unknown at the initial state, the above verification process has to be carried out for every triangular face of the wall boundary against all the cells. To be effective in finding all the local DFD cells in the initialization step, the process described in equations (6.12), (6.16) and (6.18) will be performed only when the cell is near to the triangular wall face ΔABC; otherwise it will be skipped. Once again, use the X direction as an example. If a cell satisfies the condition in equation (6.19) it will be skipped for the verification. The Y and Z coordinate values for P1 and P2 are the same, so the coordinate for P1 is sufficient for the shortlisting.

$$\begin{cases} x_{P1} < \max\left(x_A, x_B, x_C\right) \quad AND \quad x_{P2} > \min\left(x_A, x_B, x_C\right), \\ y_{P1} \in \left[\min\left(y_A, y_B, y_C\right), \max\left(y_A, y_B, y_C\right)\right], \\ z_{P1} \in \left[\min\left(z_A, z_B, z_C\right), \max\left(z_A, z_B, z_C\right)\right]. \end{cases} \tag{6.19}$$

The location of the local DFD cells in either the fluid domain or the solid domain can be determined following the similar methods as described for 2D implementation.

6.2.3 Boundary condition for immersed wall

Using the same boundary condition implementation for the immersed wall boundary in 2D LDFD method, the no-penetration condition, slip condition

and zero normal derivative approximation for pressure and density are applicable.

$$v_n = 0, \quad \frac{\partial v_t}{\partial n} = 0, \quad \frac{\partial p}{\partial n} = 0, \quad \frac{\partial \rho}{\partial n} = 0. \tag{6.20}$$

To enforce the no-penetration and slip wall boundary condition, the local normal velocity vector and the tangential velocity components need to be found out. Both normal velocity vector and tangential velocity components must be obtained in spherical coordinate system. So it is necessary to transfer the velocity vector in the Cartesian coordinate system into the corresponding spherical coordinate system.

Figure 6.5 shows the conversion between a Cartesian coordinate system and its corresponding spherical coordinate system. The normalized vector $\vec{n} = \left[n_x, \quad n_y, \quad n_z \right]$ represents the local normal direction on the wall boundary. To convert any interested vector into the spherical coordinate system, the radial distance *r*, polar angle θ and azimuthal angle ϕ are determined by:

$$\begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{n_x^2 + n_y^2 + n_z^2} \\ \arccos\left(n_z / r\right) \\ \arctan\left(n_y / n_x\right) \end{bmatrix}. \tag{6.21}$$

In this context, for a known local velocity vector $\left[u, v, w\right]$ on the wall boundary in X-Y-Z coordinate system, it will be transformed into $r - \theta - \phi$ coordinate system as

$$\begin{bmatrix} V_r \\ V_\theta \\ V_\phi \end{bmatrix} = \begin{bmatrix} n_x & n_y & n_z \\ n_x n_z/s & n_y n_z/s & -s \\ -n_y/s & n_x/s & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} V_r \\ V_\theta \\ V_\phi \end{bmatrix} = [A] \begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{6.22}$$

Here $s = \sqrt{n_x^{\,2} + n_y^{\,2}}$.

The velocity vector in $r-\theta-\phi$ coordinate system can be converted back to X-Y-Z coordinate system by

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = [A]^T \begin{bmatrix} V_r \\ V_\theta \\ V_\phi \end{bmatrix} = \begin{bmatrix} n_x & n_x n_z/s & -n_y/s \\ n_y & n_y n_z/s & n_x/s \\ n_z & -s & 0 \end{bmatrix} \begin{bmatrix} V_r \\ V_\theta \\ V_\phi \end{bmatrix}. \tag{6.23}$$

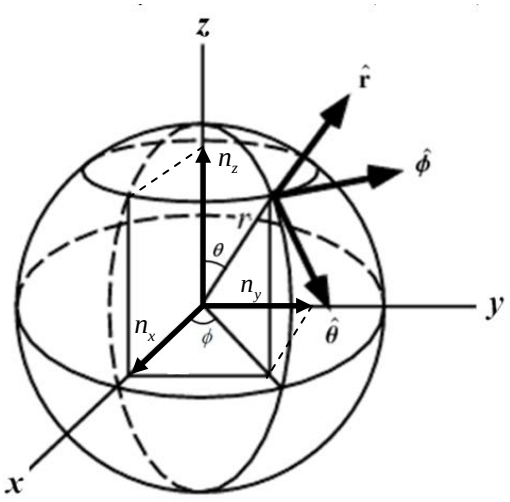


Figure 6.5 Velocity transformation between Cartesian coordinate system and Spherical coordinate system

The following steps summarize the procedure in enforcing the boundary condition (equation (6.20)) for the LDFD method.

1) For every pair of DFD cells, calculate the normal velocity and tangential velocity components using equation (6.22) on the fluid

DFD cell.

2) To enforce the no-penetration condition, the normal velocity V_r at the solid DFD cell can be calculated by equation (5.3).
3) To enforce the slip condition, the two tangential velocity components V_θ and V_ϕ at the solid DFD cell can be set by equation (5.4).
4) The corrected velocity component $\left[V_r, V_\theta, V_\phi\right]$ in the spherical coordinate system is then converted back to Cartesian coordinate system as [u, v, w] by equation (6.23).
5) The pressure and density at the solid DFD cell are set the same as those at the fluid DFD cell.
6) The energy at the solid DFD cell is updated by the equation of state for ideal gas.
7) All the conservative terms in Euler equations are finally updated with the new values of the flow variables.

Similar to the LDFD implementation for 2D flows, some solid DFD cells could be paired with fluid DFD cells in more than one direction or all the three directions of X, Y and Z. For those cells, the average weighting factors will be used to take into account the contribution of the boundary condition correction performed from multiple directions.

6.3 Validation for the 3D adaptive solver

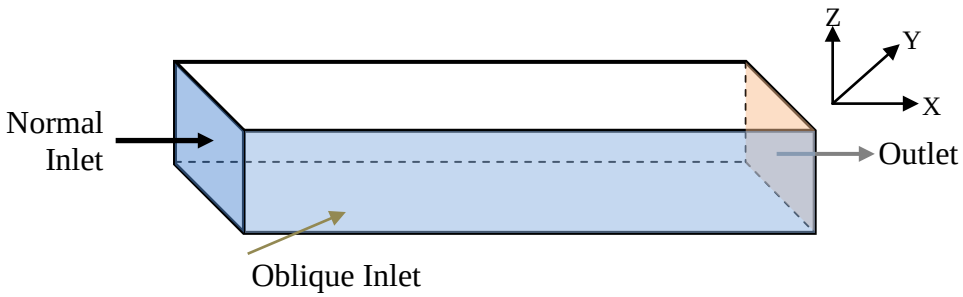
In this subsection, the performance of the 3D adaptive solver will be studied by the oblique shock problem and a 3D shock explosion problem. The two problems are chosen to test and benchmark the present 3D adaptive solver without the consideration of immersed wall boundaries. The 3D flows with immersed wall boundaries will be simulated to demonstrate the 3D IBM implementation by FC-IBM as well as LDFD method.

6.3.1 Oblique shock problem in 3D

The 2D oblique shock problem is studied in 3D domain to demonstrate the performance and accuracy of the current 3D adaptive Euler solver. The computational domain is chosen as 4×1×0.5. The normal inlet, oblique inlet and outlet boundaries are defined as shown in Figure 6.6. The boundary conditions for the normal inlet and oblique inlet are the same as those for 2D study in Chapter 2. Besides the two inlets and one outlet, the remaining three boundaries are defined as wall boundaries.

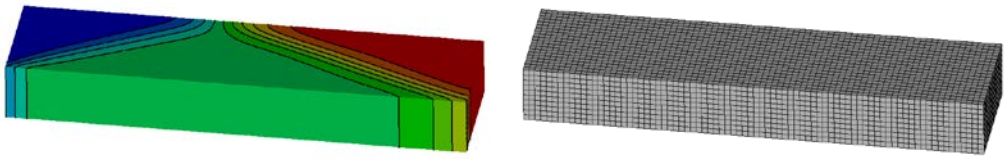
The solver is first run on uniform coarse mesh with size of $dh = 0.5$, and then the adaptive solution-based 1-level refinement and 2-level refinement are conducted. The density contours on the domain boundaries and the solution adaptive meshes are presented in Figure 6.7. From the contour plots, it is noticed that the shock wave captured is much sharper with two levels of solution adaption. The number of mesh cells for uniform coarse mesh, 1-level and 2-level adaption cases are 16,000, 56,670, and 264,920, respectively. The number of mesh cells for 1-level and 2-level adaption is 56% and 74% less

respectively as compared with that needed for the uniform mesh with the same finer mesh spacing. The remarkable reduction in cell numbers will save the demands for large memory/space and long computing time. The computing times for the solutions based on three different mesh resolutions are 94 seconds, 285 seconds and 2617 seconds respectively, as shown in Table 6.1. In comparison, the computing time needed for uniform mesh solution with size of $dh = 0.25$ is 2050 seconds, and 38,860 seconds for $dh = 0.125$. This is approximately 7 times and 14 times longer than that by the adaptive solver with the same finer mesh resolution. The numerical experiments demonstrate that the developed 3D adaptive solver is very efficient comparing to solve the flows on uniform mesh especially when higher level of adaption is used.

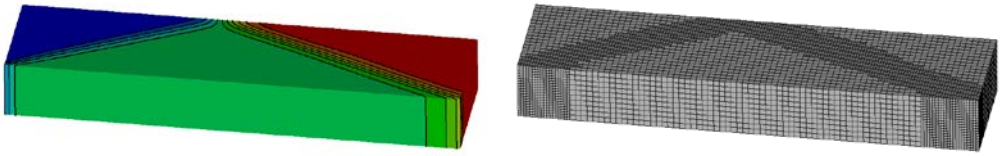


Inlet	ρ	u	v	w	p
Normal	1.0	2.9	0	0	0.7143
Oblique	1.69997	2.61934	0.50632	0	1.5282

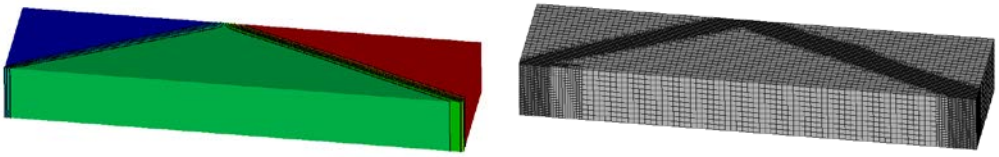
Figure 6.6 Oblique shock problem in 3D domain



(a) Density contours (10 intervals) and uniform meshes



(b) Density contours (10 intervals) and 1-level adaptive meshes



(c) Density contours (10 intervals) and 2-level adaptive meshes

Figure 6.7 Adaptive solution for the oblique shock problem in 3D domain

Mesh Size	Uniform Finer Mesh		Adaptive Mesh		
dh (h)	**No. of cells**	**CPU time (s)**	**No. of cells**	**CPU time (s)**	**Speedup**
0.5	16,000	94	-	-	-
0.25	128,000	2,050	56,670	285	7.2
0.125	1,024,000	38,860	264,920	2,617	14.8

Table 6.1 Computational time comparison for uniform mesh and adaptive mesh for 3D oblique shock problem

6.3.2 Three dimensional shock explosion

To further validate the current 3D adaptive solver, the 3D shock explosion problem is simulated. In this problem, the still air at high pressure and density is confined in a spherical volume. The computational domain for the explosion

is $[-0.5,\ 0.5]\times[-0.5,\ 0.5]\times[-0.5,\ 0.5]$. The sphere volume is defined as $r \le 0.2$ and the initial conditions for the problem are listed in the table below:

	ρ	u	v	w	p
$r \le 0.2$:	1.0	0	0	0	1
$r > 0.2$:	0.125	0	0	0	0.1

The solution is obtained at dimensionless time $t = 0.1$ to ensure that the explosion waves do not reach the domain boundaries. All the six boundaries of the computational domain are defined as outlets, and the flow information is extrapolated from the interior cells based on zero normal gradients. The solution is computed on three different meshes: coarse uniform size $dh = 0.02$, 1-level adaption with finest mesh size $dh = 0.01$ and 2-level adaption with finest mesh size $dh = 0.005$. The density gradient is used as the indicator for mesh adaption, as it reflects all the three types of waves (rarefaction wave, contact discontinuity wave and shock wave) during the wave propagation. Because the spherical waves travel inside the domain, the mesh cells in $(x, y, z) > 0$ octant are output for the plot of the results to achieve better result visualization. Figure 6.9 shows the present results of density contours, solution adapted meshes. The results clearly show that the finest meshes are adapted to the regions of the wave shape. The density profile in the radial direction is plotted along the line from (0, 0, 0) to (0.5, 0, 0), as shown in Figure 6.8. It is observed that the present results agree well with the analytical data obtained by Lahooti and Pishevar [59]. Apart from the accuracy of the solver, the efficiency improvement is also examined. Comparison of the computational time required for the solver on uniform mesh and adaptive solver shows that

the speedup rate is about 7 times and 14 times when the finest meshes' size are $dh = 0.01$ and $dh = 0.005$, respectively.

Mesh Size	Uniform Finer Mesh		Adaptive Mesh		
dh (h)	**No. of cells**	**CPU time (s)**	**No. of cells**	**CPU time (s)**	**Speedup**
0.02	125,000	31	-	-	-
0.01	1,000,000	866	374,816	119	7.3
0.005	8,000,000	15,120	2,016,477	1,090	13.9

Table 6.2 Computational time comparison for uniform mesh and adaptive mesh for 3D shock explosion problem

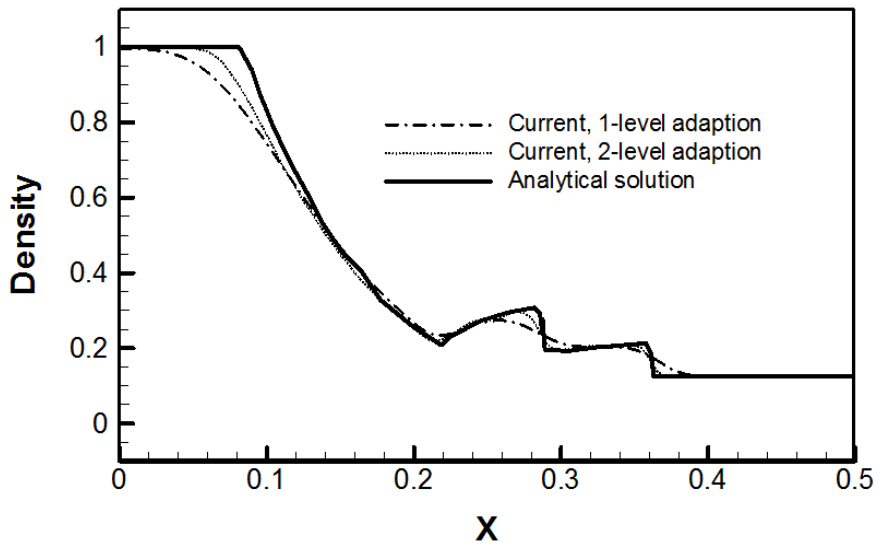
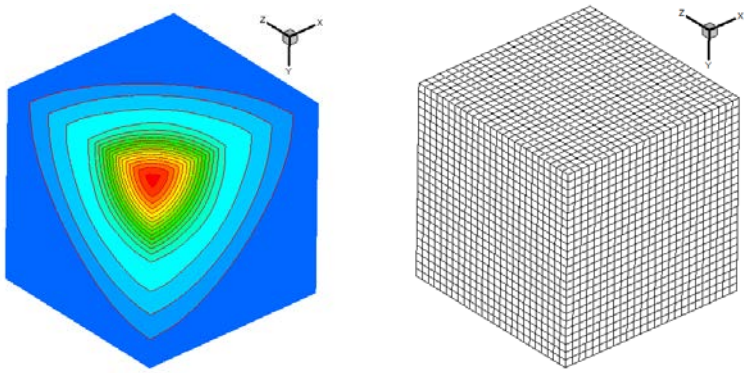
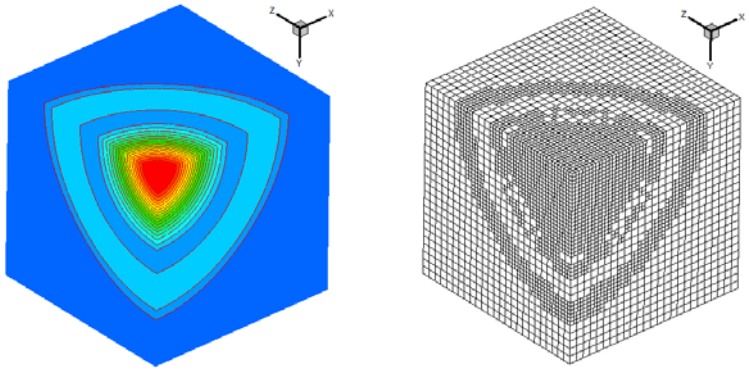


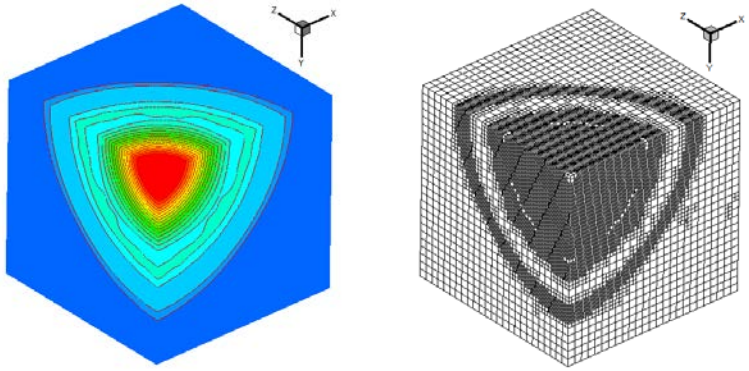
Figure 6.8 Density profile in radial direction for the 3D shock explosion problem



(a) Density contours (20 intervals) and uniform meshes



(b) Density contours (20 intervals) and 1-level adaptive meshes



(c) Density contours (20 intervals) and 2-level adaptive meshes

Figure 6.9 Adaptive results for the 3D shock explosion problem

6.4 Numerical examples for 3D IBM methods

In this subsection, the FC-IBM and LDFD method are implemented together with the 3D adaptive Euler solver. To validate the developed method, simulation of Mach 3 flow over a sphere is performed and the results are compared with numerical data available in the literature.

6.4.1 Mach 3 flow over a sphere

As a benchmark case, Mach 3 flow over a sphere is simulated to validate the current adaptive solver with immersed boundary method implemented for 3D problems. In this case the bow shock generated in front of the sphere can be used for comparison and accuracy analysis.

The radius of the sphere is one unit. Because the free stream flow is in supersonic range and the bow shock is in front of the sphere, only half of the sphere is chosen for the simulation, as shown in Figure 6.10. The computed results are plotted in Figure 6.11 and Figure 6.12. To compare the results quantitatively, the Mach number distribution along the central line from (-2,0,0) to (-1,0,0), as indicated by the solid line $\overline{AB}$ in Figure 6.10, is plotted in Figure 6.11 and is compared with the numerical results obtained on a structured mesh solver by Rispoli et al. [60].

It is noticed that the Mach number profiles computed by FC-IBM and LDFD have very minor difference, and both are comparable to the numerical results obtained on structured grid by Rispoli et al. [60]. In order to visualize the

results near the immersed body clearly, the Mach number contours are only plotted when the mesh cells are in the region of $(y,z)\geq 0$.

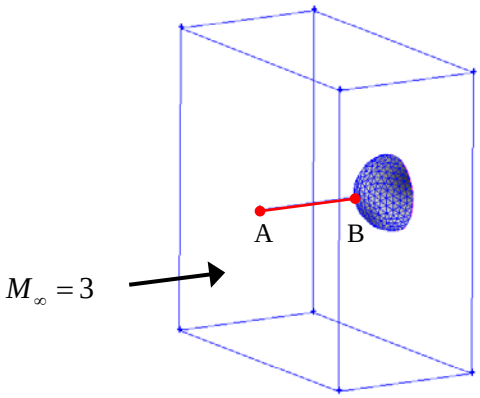


Figure 6.10 Computational domain for Mach 3 flow over a sphere

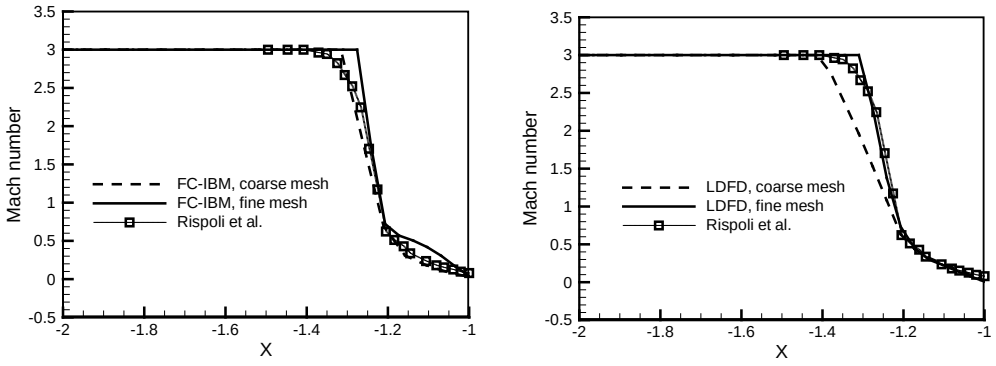
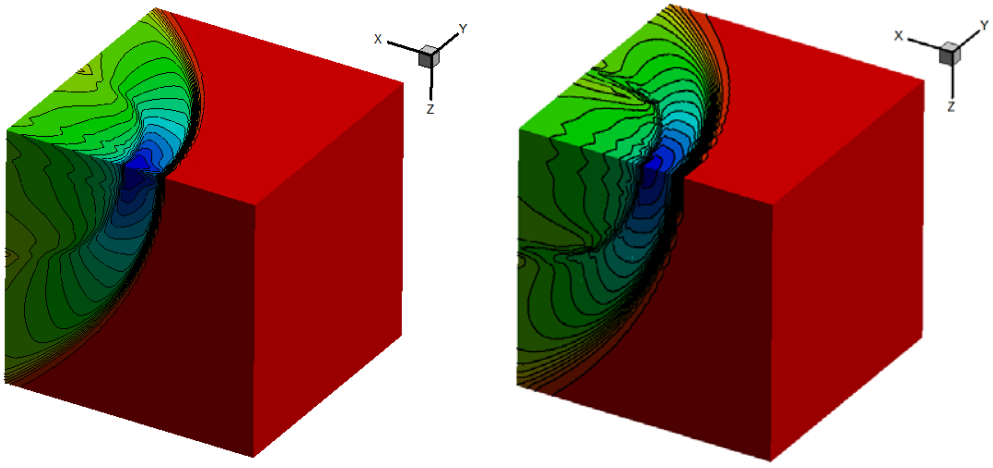


Figure 6.11 Mach number profile for Mach 3 flow over a sphere



(a) Computed by FC-IBM (b) Computed by LDFD method

Figure 6.12 Mach number contours for Mach 3 flow over a sphere

6.4.2 Supersonic flow over 3D objects

In the previous subsection, shock propagation over a simple spherical surface is simulated. To demonstrate the capability of the developed adaptive-immersed boundary solver in handling 3D complex boundaries, supersonic flows over a 3D aircraft head and an Apollo-shaped re-entry vehicle are simulated using the adaptive LDFD method, as this method demonstrates to be more robust and accurate. The geometry and the surface meshes of the aircraft head and the re-entry vehicle are plotted in Figure 6.13 (the outer surfaces of the aircraft head and the re-entry vehicle are represented by triangular mesh cells generated in Gambit.) The computational domain is 3D box which covers the boundary surfaces completely.

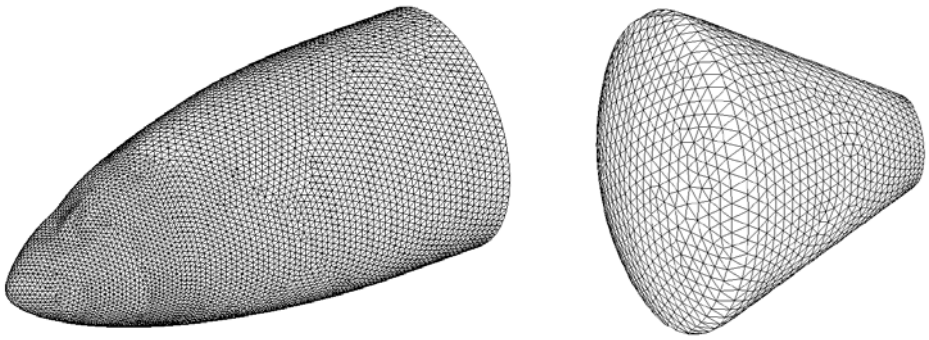
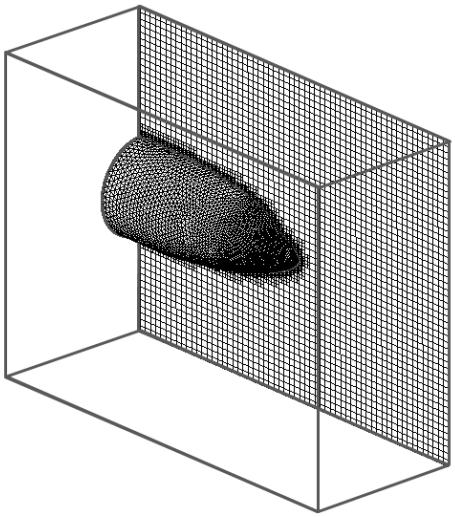


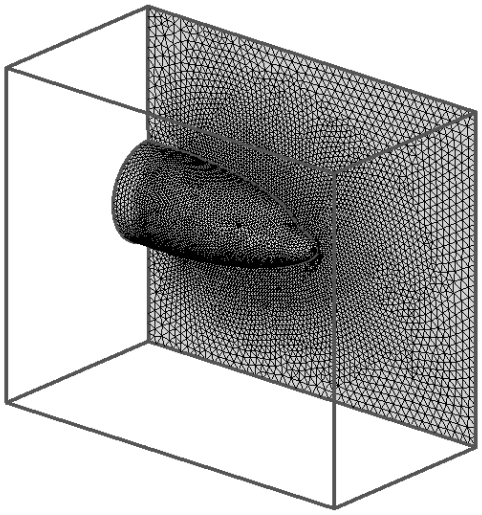
Figure 6.13 Geometry and surface meshes on 3D aircraft head and an Apollo-shaped re-entry vehicle

Firstly, supersonic flow at Mach number 2 and the angle of attack AoA=20° over an aircraft head is computed. The case is an extension of the 2D supersonic flow over a double-ellipse structure presented in Chapters 4 and 5. In order to compare and validate the current solver, the supersonic flow with same surface meshes on the aircraft head, same domain size and same flow

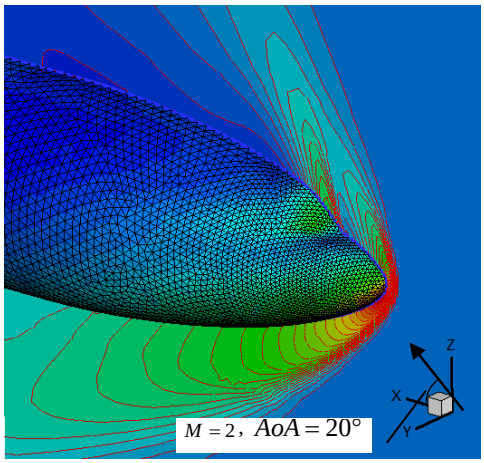
conditions is solved using Fluent solver based on tetrahedral meshes. The symmetrical modeling is applied for this study in reducing the computational demand. The meshes on the aircraft head structure and the symmetrical plane, and the computational domain are plotted in Figure 6.14 (a) and (b) for the current solver and Fluent solver, respectively.



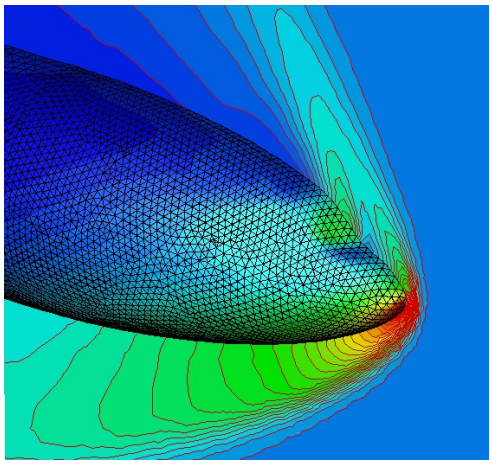
(a) 3D domain and Cartesian mesh for the current solver



(b) 3D domain and tetrahedral mesh for Fluent solver



(c) Pressure contour computed by the current solver



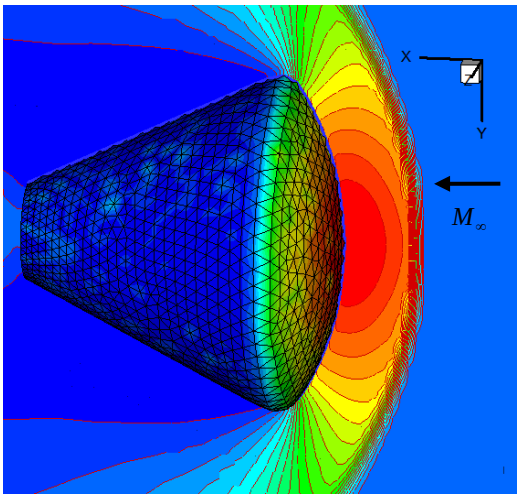
(d) Pressure contour computed by Fluent solver

Figure 6.14 Pressure contours for 3D supersonic flows over an aircraft head with M=2, AoA=20°

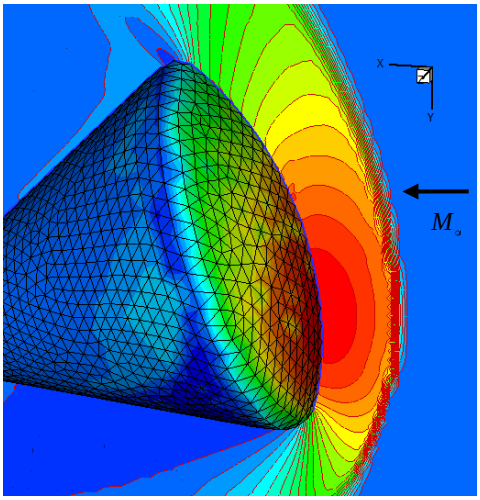
The computed pressure contours are plotted on the central surface of the domain and the aircraft body, as shown in Figure 6.14 (c) and (d) for the current solver and Fluent solver, respectively. The bow shock structure captured by the current solver agrees well with that computed by Fluent solver.

Another test case is to simulate the supersonic flow over an Apollo-shaped re-entry vehicle. The free stream incoming flow is at Mach number 2.5 and with two angle of attack *AoA*=0° and 20°. To demonstrate the capability of the present method in dealing with different immersed wall boundaries, the free stream incoming flow is fixed at $M_\infty = 2.5$ along the positive X direction and the flow condition at *AoA*=20° is achieved by rotating the re-entry vehicle surface by 20 degrees along +Z direction. Because there is no change for the computational domain and the 3D Cartesian meshes, no additional work is required for the solver in simulating the flows at *AoA*=0° and *AoA*=20°. The influence of the re-entry vehicle body introduced into the domain will be enforced according to the actual vehicle surface and its boundary location.

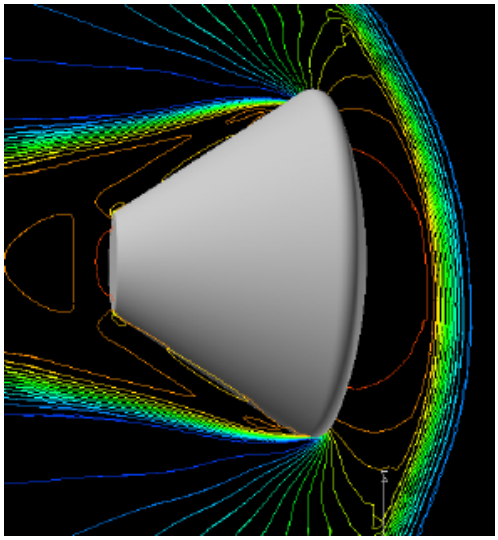
The computed pressure contours are plotted on the central surface of the domain and the re-entry vehicle body, as shown in Figure 6.15 (a) and (b) for AoA=0° and AoA=20°, respectively. It can be seen that the bow shock in front of the vehicle is captured and adapted to the different angle of attack. The shock waves predicted by the present solver are comparable to the numerical results obtained on body-fitted grids by Marcy [61] as shown in Figure 6.15 (c) and (d).



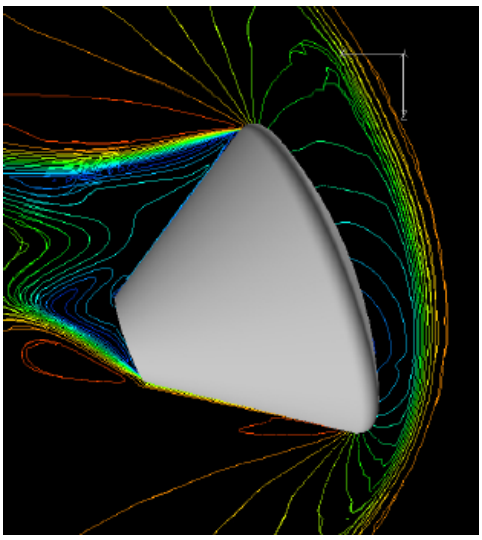
(a) 3D flow over a re-entry vehicle at Mach number 2.5, AoA=0°.



(b) 3D flow over a re-entry vehicle at Mach number 2.5, AoA=20°.



(c) Mach number 2.5, AoA=0° [61].



(d) Mach number 2.5, AoA=20° [61].

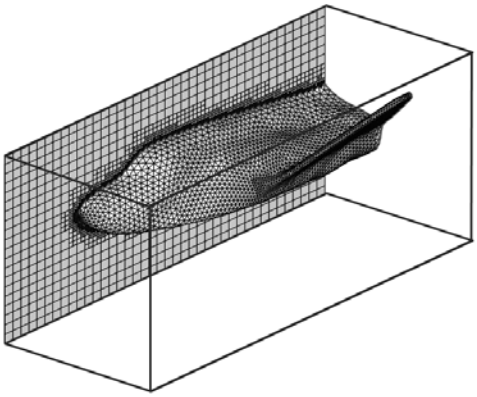
Figure 6.15 Pressure contours for 3D supersonic flow over an Apollo-shaped re-entry vehicle with M=2, AoA=0° and 20°

6.4.3 Supersonic flow over a 3D space vehicle

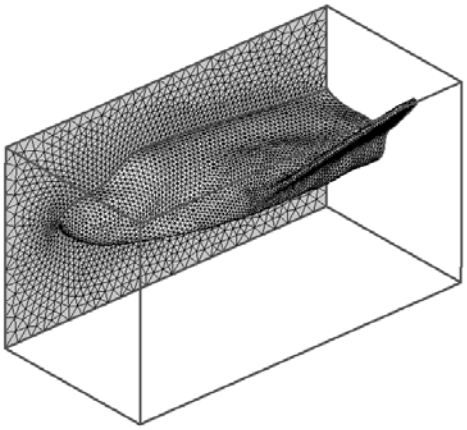
In the previous two numerical test cases, supersonic flows over an aircraft head and an Apollo-shaped re-entry vehicle are computed. The geometry of the aircraft head and the re-entry vehicle is considered to be bluff body, which

is relative simple. In this test case, supersonic flow over a full three-dimensional space vehicle, including a craft body and a pair of wings, will be studied using the current solver. As the thickness of the wings is about 2% of the full dimension of the space vehicle, finer mesh resolution is required to resolve the thin geometry structure of wings accurately. This is a challenging case as the size of the 3D problem will become relatively large. To compare and validate the current solver, an identical numerical model is created with unstructured tetrahedral meshes and the flow is solved by Fluent solver with same flow and boundary conditions.

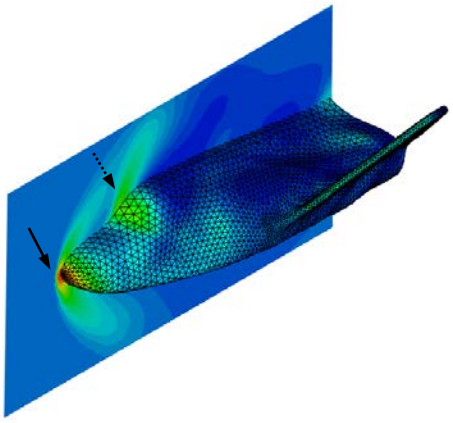
The computational domain and mesh for the current solver and Fluent solver is shown in Figure 6.16 (a) and (b), respectively. The flow condition is defined with the free stream incoming flow at Mach number 2.0 and the angle of attack at zero degree. The computed pressure contours on the surface of the space vehicle and the middle symmetrical plan are plotted in Figure 6.16 (c) and (d) for the current solver and Fluent solver. The plots show that similar flow pattern is obtained from the current solver and Fluent solver, which is illustrated by the bow shock wave (indicated as solid-line arrow) predicted in front of the space vehicle and the weak oblique shock wave (indicated as dotted-line arrow) captured near the cockpit window. The pressure distribution on the space vehicle surface and the middle symmetrical plan is plotted in Figure 6.16 (f). The results also show that the pressure distribution obtained by the current solver and Fluent solver match accurately well. This demonstrates the current solver's capability in solving flows over complex 3D structures.



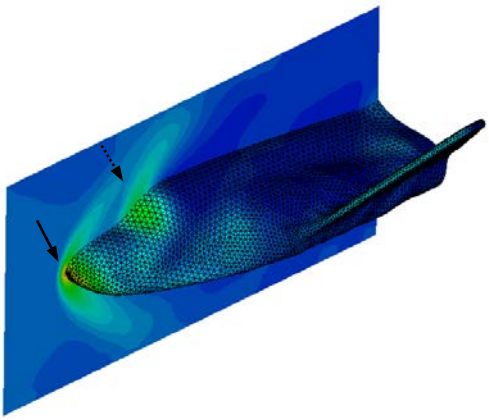
(a) 3D domain and Cartesian mesh for the Current solver



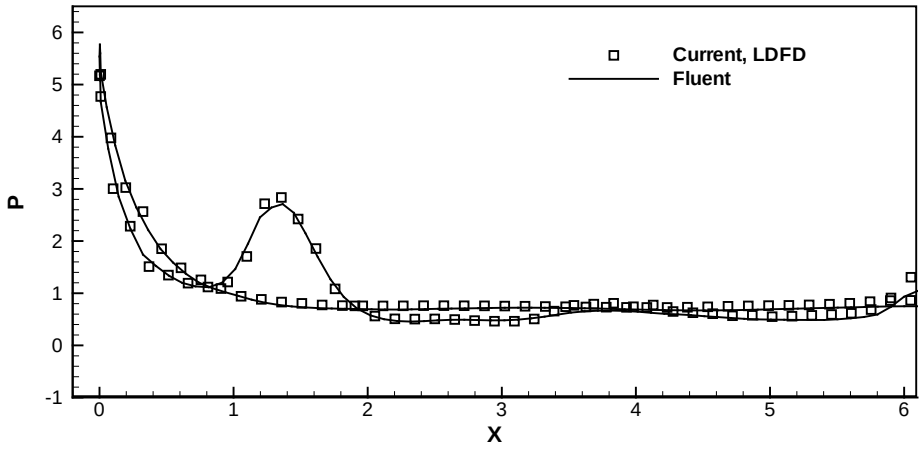
(b) 3D domain and tetrahedral mesh for Fluent solver



(c) Pressure contour computed by the Current solver



(d) Pressure contour computed by Fluent solver



(f) Pressure distribution on the space vehicle surface and the symmetrical plan

Figure 6.16 Computed results for supersonic flow over a 3D space vehicle with M=2, AoA=0°

6.5 Conclusions

The 3D adaptive Euler solver is developed using the same approach as that for the 2D adaptive Euler solver. The FC-IBM and LDFD method proposed in Chapters 4 and 5 are implemented in the 3D adaptive solver to enforce the boundary condition on wall boundary which is represented by 3D triangular surface segments.

The accuracy of the 3D adaptive solver is validated and the performance of the solver is benchmarked. The results show that the adaptive solver is able to achieve finer mesh solution efficiently with better accuracy, in particular when higher level of adaption is used. The FC-IBM and LDFD implementation on the 3D adaptive Euler solver is validated by the 3D supersonic flow over a sphere. The results show good agreement with the reference data. Supersonic flows over an aircraft head, an Apollo-shape re-entry vehicle and a space vehicle with a pair of wings have been simulated by the LDFD method. The results demonstrate that the method is accurate and robust, and is a good potential tool for the application to solve such kind of supersonic flows over complex geometry.

Chapter 7 Development of Adaptive Viscous Solver for Laminar Flows

Since Peskin first introduced the immersed boundary method to simulate the complex geometry-fluid interactions on a fixed Cartesian mesh ([20], [21]), significant efforts have been made to refine this method. Though the current work was initiated with the objective to solve compressible inviscid flows, the developed adaptive solver and the proposed IBM implementation can be easily modified to solve viscous flows in laminar condition with minor effort.

7.1 Laminar viscous flow solver

The dimensionless, two-dimensional compressible Navier-Stokes equations in the conservative form are given as below using the far field reference Mach number and Reynolds number.

$$\frac{\partial U}{\partial t}+\frac{\partial F}{\partial x}+\frac{\partial G}{\partial y}=\frac{\sqrt{\gamma}M_{\infty}}{\mathrm{Re}_{\infty}}\left(\frac{\partial F_v}{\partial x}+\frac{\partial G_v}{\partial x}\right). \tag{7.1}$$

Compared to the Euler equations in Chapter 2, the viscous stresses on the right side of the equation need to be included in the solver for viscous flows. The viscous stresses are given by

$$F_v=\begin{bmatrix}0\\ \tau_{xx}\\ \tau_{xy}\\ u\tau_{xx}+v\tau_{xy}-q_x\end{bmatrix},\qquad G_v=\begin{bmatrix}0\\ \tau_{xy}\\ \tau_{yy}\\ u\tau_{xy}+v\tau_{yy}-q_y\end{bmatrix}. \tag{7.2}$$

With the Stokes hypothesis, $\lambda = -\frac{2}{3}\mu$, the dimensionless stress tensor and heat flux vector q are given by

$$\tau_{xx} = 2\frac{\partial u}{\partial x} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = \frac{2}{3}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right), \tag{7.3}$$

$$\tau_{yy} = 2\frac{\partial v}{\partial y} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = \frac{2}{3}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right), \tag{7.4}$$

$$\tau_{xy} = \tau_{yx} = \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \tag{7.5}$$

$$q_x = \frac{\gamma}{\gamma - 1}\frac{1}{\mathrm{Pr}}\frac{\partial\left(p/\rho\right)}{\partial x}, \qquad q_y = \frac{\gamma}{\gamma - 1}\frac{1}{\mathrm{Pr}}\frac{\partial\left(p/\rho\right)}{\partial y}. \tag{7.6}$$

The Reynolds number (*Re*), Prandtl number (*Pr*) and Mach number (M_∞) used for the dimensionalization of Navier-Stokes equations are defined as

$$\mathrm{Re} = \frac{\rho U_\infty d}{\mu}, \qquad \mathrm{Pr} = \frac{\mu C_P}{K}, \qquad M_\infty = \frac{U_\infty}{c}. \tag{7.7}$$

Here, U_∞ is the far field reference velocity, d is reference length, ρ is density, μ is the molecular viscosity, C_p is the heat capacity, K is the conductivity, and c is the sound speed.

For the viscous solver, the inviscid flux on the cell interface is still calculated following the HLLC scheme as described in Chapter 2. The viscous flux on the cell interface is calculated via the central difference scheme using the first order derivatives computed on the cell centers.

$$f_{viscous} = \frac{1}{2}\left(f_v^L + f_v^R \right),$$

$$f_v^{L,R} = \frac{\sqrt{\gamma} M_\infty}{\mathrm{Re}_\infty} \left\{ n_x \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{bmatrix} + n_y \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y \end{bmatrix} \right\}. \tag{7.8}$$

And the solution evolution in time domain is

$$U_c^{n+1} = U_c^n - \frac{\Delta t}{A_c} \sum \left(f_{invscid} - f_{viscous} \right)_i \vec{l}_i \cdot \vec{n}_i. \tag{7.9}$$

7.2 IBM implementation for viscous flows

The flux correction-based immersed boundary method (FC-IBM) proposed in Chapter 4 for inviscid flows will be modified to enforce wall boundary condition in the current laminar viscous flow solver, due to its simple and ease in the actual implementation and other successful implementation of IBM for incompressible viscous flows. For viscous flows, the no-slip condition may be the most commonly used boundary condition. The no-slip boundary condition indicates that both X and Y velocity components of fluid at the solid surface must be the same as that of the solid boundary. Compared to the no-penetration condition for inviscid flows, there is no need to perform the velocity transformation to obtain the normal velocity on the boundary. The target of the velocity correction is to satisfy both X and Y velocity components on the boundary to be zero for stationary boundary, thus the no-slip condition will be enforced and satisfied.

In the current method, the velocity correction is performed via an iterative method similar to that described in Chapter 4. Because the implementation process of the no-slip condition for X direction velocity component (u) and Y direction velocity component (v) is the same, the detailed process is only illustrated for the velocity (u) in X direction. Assuming that the boundary velocity is known as (u_w^0, v_w^0), the velocity correction procedures are:

1) Using the delta function to compute the velocity on the boundary from the nearby shadow cells: $u_w^* = \sum\left(u_i \cdot D_{i,j}\right)$.
2) Setting the velocity correction as $\delta u_w = \left(u_w^0 - u_w^*\right)$, as the computed velocity u_w^* may not equal to the velocity (u_w^0) of boundary.
3) Distributing the velocity correction on the boundary back to the nearby shadow cells via the delta function and obtaining the corrected velocity on the shadow cells as: $u_i^* = u_i + \sum\left(\delta u_w \cdot D_{j,i}\right)$.
4) Using the corrected velocity on cells to repeat steps 1) to 3) until the no-slip condition is satisfied when the velocity correction on the boundary is within the required criteria ε, e.g. $\left|\delta u_w\right| = \left|u_w^0 - u_w^*\right| \le \varepsilon$.

When the boundary condition is satisfied through the above iterative procedure, the effective velocity correction on an individual cell near the boundary is

$$\delta u_i = \left(u_i^0 - u_i^*\right). \tag{7.10}$$

As derived by Wu and Shu [32], the force exerted on the solid body can be computed via the integration of the velocity correction δu_i on all cells, of which the velocity is corrected.

$$F_{Solid} = \sum_{i}^{\Omega}\left[\left(\rho_i \frac{\delta u_i}{\delta t}\right)\times\left(dx\cdot dy\right)\right],\ \ \Omega \in \{\text{all the shadow cells}\}. \tag{7.11}$$

As the current solver is a compressible flow solver, so the fluid density at the cell center is used in the force computation. Although the force calculated by equation (7.11) is the force exerted on the fluid, it is equivalent of the force exerted onto the boundary of the immersed solid in the opposite direction due to Newton's third law of motion.

7.3 Numerical tests

The flow past a circular cylinder is chosen as the validation case for the current laminar flow solver. As a classic problem this flow has been studied extensively, and there are many theoretical, experimental and numerical data available in the literature for comparison. Another validation case to be simulated in this subsection is the laminar flow over a NACA0012 airfoil. The pressure distribution on the airfoil surface will be compared for accuracy analysis.

7.3.1 Flow over a circular cylinder

Reynolds (Re) number is typically used to classify the flow behaviors for numerical studies of flow over a circular cylinder. The *Re* in this analysis is

determined by the reference free stream velocity (U_∞), the diameter (D) of the cylinder and the dynamic viscosity (ν) of the fluid by

$$\mathrm{Re} = \frac{U_\infty D}{\nu}. \tag{7.12}$$

The laminar flows are studied numerically at four different Reynolds numbers: 20, 40, 100 and 200. The flow characteristic at *Re* of 20 and 40 is dominated by a pair of symmetrical and stationary recirculation vortices behind the cylinder. At the relative higher *Re* of 100 and 200, the flow characteristic is dominated by a repeating pattern of swirling vortices downstream of the cylinder, which is well-known as the Kármán vortex street.

The flow conditions for this study are

$$\rho = 1.4, \quad p = 1, \quad U_\infty = 0.1, \quad M_\infty = 0.1, \quad D = 1. \tag{7.13}$$

The computational domain is 40D×40D with coarse uniform mesh at 200×200, and the flow is from left to right in X direction. The finest meshes are clustered around the cylinder via the adaption technique developed in the current solver to increase the mesh resolution locally. Adaptive mesh refinement is performed based on the magnitude of the vorticity in order to improve the solution accuracy in the recirculation region behind the cylinder and the swirling vortices downstream.

The drag force introduced by the cylinder can be calculated based on the velocity correction on all shadow cells in X direction.

$$F_D = \sum_i^{\Omega}\left[\left(\rho_i \frac{\delta u_i}{\delta t}\right)\times\left(dx\cdot dy\right)\right], \quad \Omega \in \{\text{all the shadow cells}\}. \tag{7.14}$$

And then the drag coefficient (C_d) is derived as

$$C_d = \frac{F_D}{(1/2)\rho_\infty U_\infty^2 D}. \tag{7.15}$$

Similarly, the lift force (F_L) introduced by the cylinder and the lift force coefficient (C_l) can be calculated based on the velocity correction on all shadow cells in Y direction.

$$F_L = \sum_i^{\Omega}\left[\left(\rho_i \frac{\delta v_i}{\delta t}\right)\times\left(dx\cdot dy\right)\right], \quad \Omega \in \{\text{all the shadow cells}\} \tag{7.16}$$

$$C_l = \frac{F_L}{(1/2)\rho_\infty U_\infty^2 D} \tag{7.17}$$

Besides the drag coefficient (C_d), the recirculation length (L) and the flow separation angle (θ), as shown in Figure 7.1, are often used for comparison and accuracy analyses.

As the Mach number for this test case is $M_\infty = 0.1$, air can be considered as incompressible fluid. Hence the results can be compared and validated by the numerical results obtained via those incompressible flow solvers. At low Re numbers of 20 and 40, the drag coefficient (C_d), the recirculation length (L) and the flow separation angle (θ) computed by the present solver are compared with the data available in the literature. The details are tabulated in

Table 7.1. It can be seen that the current results agree well with the numerical results obtained from other IBM implementations for this problem. At the condition of *Re*=20, the drag coefficient obtained by the current solver is 2.11~2.12, which matches well to other numerical results in the range of 2.0~2.152. At the condition of *Re*=40, the drag coefficient obtained by the current solver is about 1.58~1.59, which matches well to other numerical results in the range of 1.498~1.59 as well. In general, the recirculation length and the separation angle also match well with other numerical results as well for both *Re* numbers of 20 and 40.

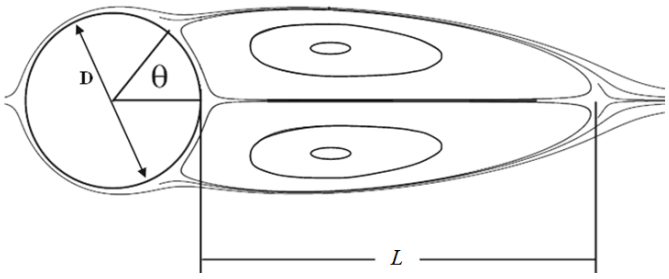
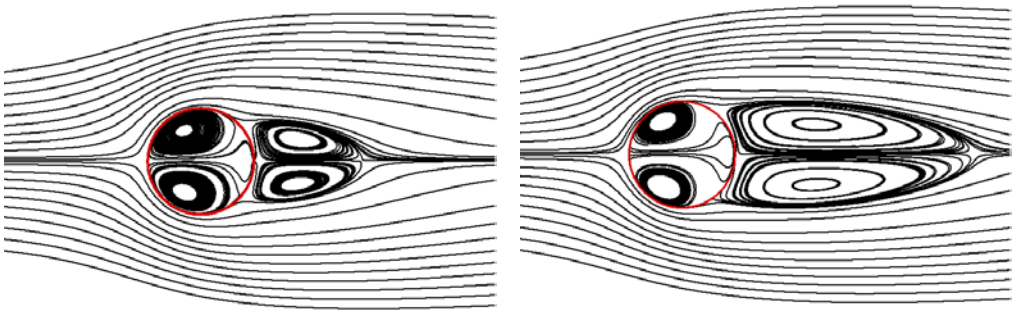


Figure 7.1 Flow characteristics of the flow over a circular cylinder

Streamlines obtained around the circular cylinder at Re=20 and 40 are plotted in Figure 7.2 (a) and (b), respectively. No-penetration of streamlines is observed in the fluid domain outside the cylinder and also inside the cylinder, which indicates the no-penetration boundary condition is well satisfied. It agrees well with the results obtained by Wu and Shu [32] using the IVC-IBM for incompressible flows.

Case	References	Cd	2L/D	θ
Re=20	Wu & Shu [32]	2.091	1.86	-
	Taira et al. [62]	2.06–2.07	1.88-1.94	43.3°-44.1°
	Others [22], [68]	2.0–2.152	1.82-1.88	43.3°-45.0°
	Present			
	coarse mesh (1/40)	2.12	1.98	43.2°
	fine mesh (1/80)	2.11	1.857	43.3°
Re=40	Wu & Shu [32]	1.565	4.62	-
	Taira et al. [62]	1.54-1.55	4.6-4.66	53.7°-54.1°
	Others [22], [68]	1.498–1.59	4.26-4.69	53.6°-54.1°
	Present			
	coarse mesh (1/40)	1.58	4.47	53.2°
	fine mesh (1/80)	1.59	4.49	53.1°

Table 7.1 Comparison of drag coefficient C_d, recirculation length L and flow separation angle θ for steady-state laminar viscous flow over a circular cylinder at *Re*=20 and 40.



(a) Re=20 (b) Re=40

Figure 7.2 Computed streamlines for the flow over a circular cylinder

When Re increases, a repeating flow pattern of swirling vortices downstream of the cylinder is developed. This is the well-known Kármán vortex street. The flow pattern under this condition is asymmetrical and repeated periodically.

Table 7.2 presents the current results obtained at Re of 100 and 200 with the drag coefficient, lift coefficient and Strouhal number (*St*) and the comparison with other numerical results. The Strouhal number is a dimensionless number describing oscillating flow pattern such as the Kármán vortex street for this study and is defined as

$$St = \frac{f_T D}{U_\infty}. \tag{7.18}$$

The f_T in the above equation is the frequency of the periodic flow pattern.

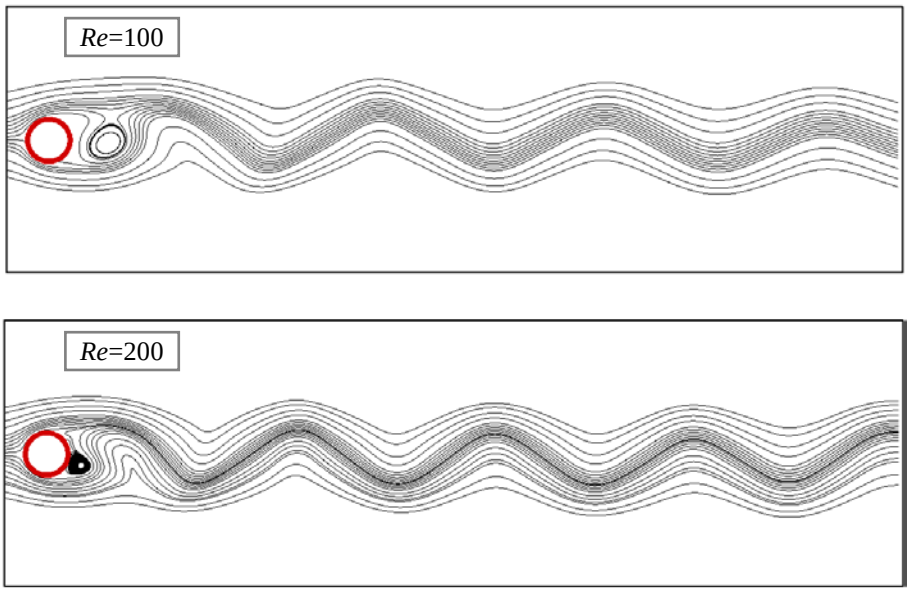
From the results listed in Table 7.2, it is observed that the average drag coefficient computed via the current compressible flow solver in general agrees well with other numerical predictions ([24], [32], [69]-[71]). The present average drag coefficient is 1.38 for *Re*=100 and 1.36 for *Re*=200. The values are in the high band among the published numerical results of 1.329~1.384 for *Re*=100 and 1.19~1.4 for *Re*=200. The computed magnitude of the lift coefficient has the similar tendency and is also in the high band among the published numerical data. The computed Strouhal number is 0.161 for *Re*=100 and 0.187 for *Re*=200, which are in good agreement with published results, ranging from 0.142~0.18 for *Re*=100 and 0.163~0.211 for *Re*=200. The comparison implies that the current solver can predict the unsteady-state periodic flows accurately.

The computed streamlines, drag coefficient and lift coefficient versus time are plotted in Figure 7.3. The contours of vorticity and the solution-adapted

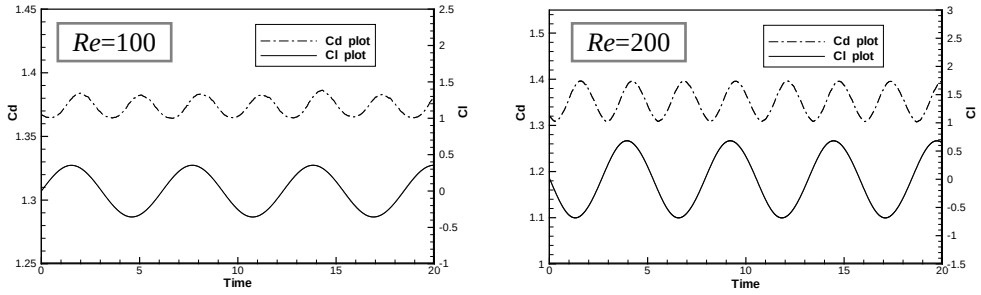
meshes are plotted in Figure 7.4. The plots demonstrate clear periodicity of the flow pattern of the Kármán vortex street downstream of the cylinder and the benefit of using solution adaptation solver for such kind of unsteady-state flow study. In the plots of vorticity contours, Figure 7.4 (a), the dotted-lines and the solid-lines denote the negative and positive levels of vorticity, respectively. The corresponding solution adapted meshes are plotted in Figure 7.4 (b). It is observed that finer meshes are adapted to the vortices closely.

Case	**References**	**Cd (average)**	**Cl**	**St**
Re=100	Wu & Shu [32]	1.364	±0.35	0.163
	Lai & Peskin [24]	1.447, 1.463	±0.33	0.144, 0.165
	Others [69]-[71]	1.325~1.384	±(0.25~0.339)	0.142~0.18
	Present	1.38	±0.354	0.161
Re=200	Wu & Shu [32]	1.349	±0.72	0.193
	Lai & Peskin [24]	-	-	0.163, 0.190
	Liu et al. [71]	1.31	±0.69	0.192
	Others [69]-[71]	1.19~1.4	±(0.5~0.69)	0.163~0.211
	Present	1.36	±0.688	0.187

Table 7.2 Comparison of drag coefficient C_d , lift coefficient C_l and Strouhal number *St* for unsteady-state laminar viscous flow over a circular cylinder at *Re*=100 and 200.

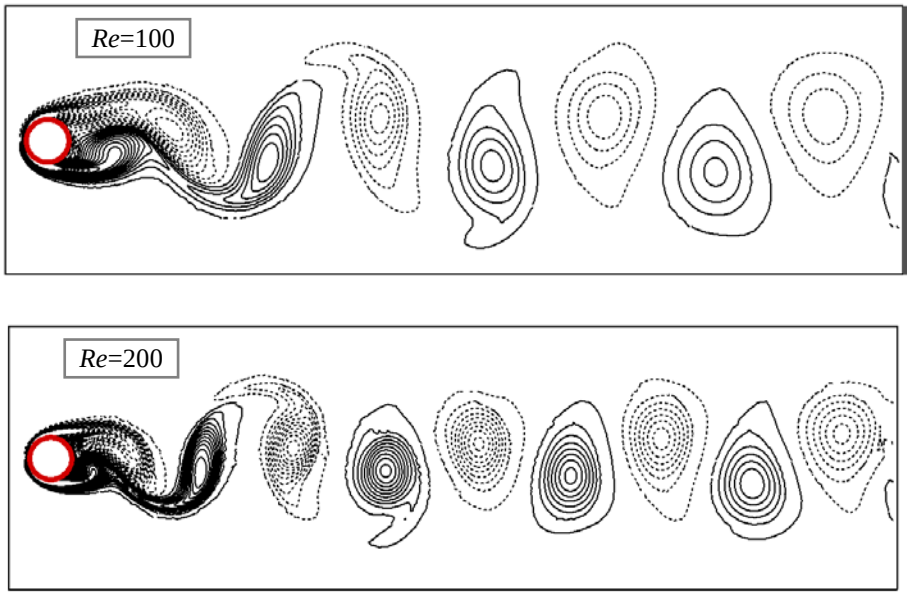


(a) Streamlines

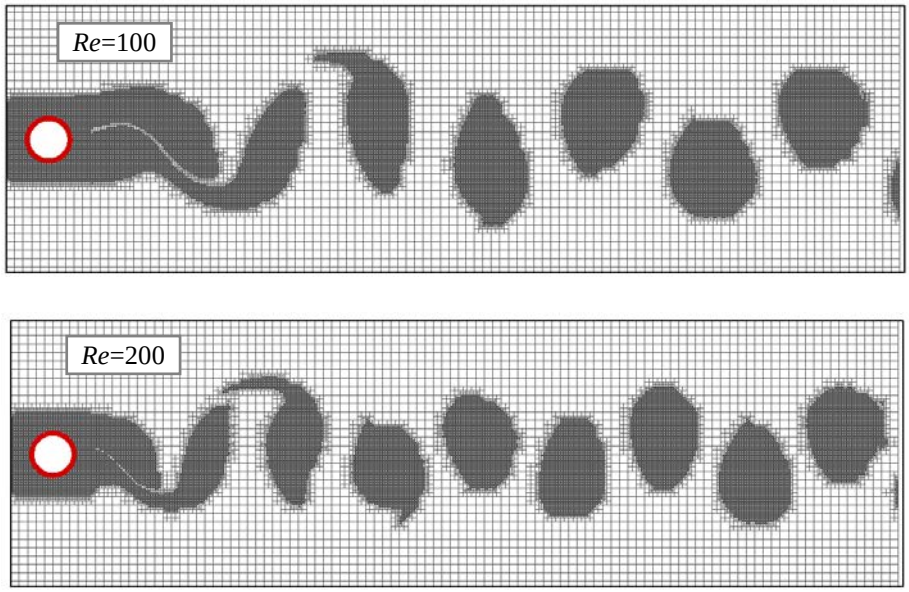


(b) Drag and lift coefficients versus time

Figure 7.3 Computed results of streamlines, drag and lift coefficient profile for the flow over a circular cylinder at *Re*=100 and 200



(a) Contours of vorticity



(b) Solution-adapted meshes.

Figure 7.4 Computed results of vorticity contours and adapted meshes for the flow over a circular cylinder at *Re*=100 and 200

7.3.2 Flow over a NACA0012 airfoil

The second validation case for the current viscous adaptive solver is to simulate the flow over a NACA0012 airfoil. Flow structures in two different conditions are computed. The first case is defined with the Reynolds number at Re=500, the free steam velocity U_∞ =0.1 and the reference Mach number M_∞=0.1, and the angle of attack *AoA*=0°. The flow condition for the second case is the Reynolds number at Re=500, the free steam velocity U_∞ =0.8 and the reference Mach number M_∞=0.8, and the angle of attack *AoA*=10°. The chord size of the airfoil is c=1 and the head is located at the center of the domain. The computational domain is 40c×40c with coarse uniform mesh of 200×200. To resolve the thin geometry of the airfoil accurately, finer meshes are clustered around the airfoil surface. Figure 7.5 shows the configuration of the computational domain for this study and the clustered mesh distribution near the airfoil. Because the changes of flow are not so great on the cells far away from the airfoil, lower level adaption is used to refine those mesh cells based on the vorticity of flow.

For the first case study with the flow condition of $\mathrm{Re} = 500$, $U_\infty = 0.1$, $M_\infty = 0.1$, and $AoA = 0°$, the computed results of the pressure coefficient on the airfoil surface and the streamlines superimposed on the pressure contours are presented in Figure 7.6. The results compare well with the numerical results obtained via the velocity correction-based IBM for incompressible flows by Wu and Shu [32]. To be specific, the drag coefficient computed by the current solver is $C_d = 0.1761$, which agrees closely with the drag

coefficient $C_d = 0.1759$ predicted by Wu and Shu [32].

For the second case study with the flow condition of $\mathrm{Re} = 500$, $U_\infty = 0.8$, $M_\infty = 0.8$, and $AoA = 10°$, the computed results of the pressure coefficient on the airfoil surface and the streamlines superimposed on the pressure contours are presented in Figure 7.7. The current result is compared with the published numerical results obtained by Jahangirian and Hashemi [13]. The pressure coefficient profile predicted on the airfoil surface agrees well with the numerical results. To be specific, the current drag force coefficient predicted is 0.479, which matches fairly to their result of 0.475. The separation point for this flow above the airfoil is at about 37.5% in the chord-wise direction, which is also in good agreement with their results of 37% with maximum error of 1%.

The results computed for both cases of flow over a NACA0012 airfoil demonstrate that the current adaptive viscous solver is accurate and efficient for flow simulations over streamlined geometry like airfoil in different flow conditions. Other flow conditions, such as various Reynolds numbers, different angle of attack, multiple element airfoils, and moving airfoil can be studied easily using the current viscous solver with its specific features of solution adaption and ability in handling immersed wall boundary.

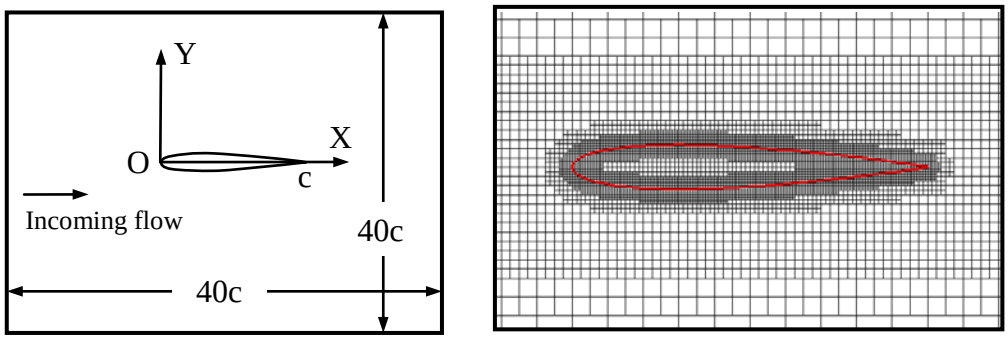


Figure 7.5 Configuration of flow over NACA0012 airfoil and clustered mesh near the airfoil

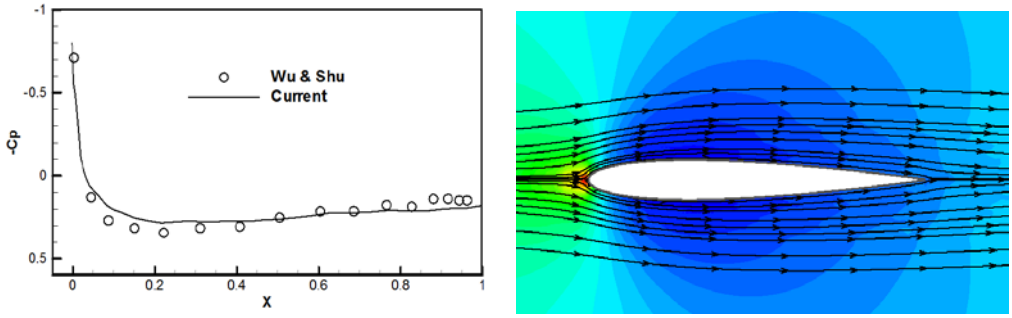


Figure 7.6 Computed pressure coefficient, pressure contours and streamlines for NACA 0012 airfoil at $M_\infty = 0.1$, $\mathrm{Re} = 500$, $AoA = 0°$

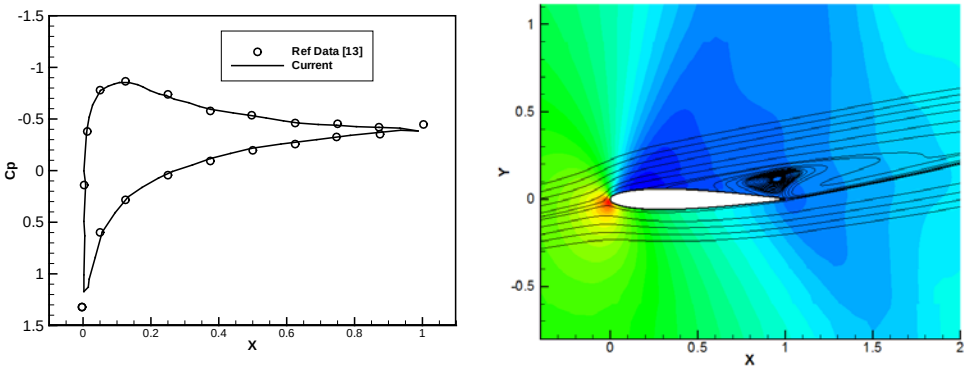


Figure 7.7 Computed pressure coefficient, pressure contours and streamlines for NACA 0012 airfoil at $M_\infty = 0.8$, $\mathrm{Re} = 500$, $AoA = 10°$

7.4 Conclusions

With the viscous stresses considered, the present 2D adaptive Euler solver is extended to solve laminar viscous flows easily. The fluxes contributed by the viscous stresses are computed using central difference scheme. Because no-slip condition is defined for wall boundary in viscous flows, the IBM implementation is achieved easily by correcting the velocity field near the boundary to satisfy zero velocity condition for stationary boundary.

The laminar viscous solver is validated by simulating laminar flow over a circular cylinder in steady-state condition at Re=20 and 40, and also unsteady-state condition with Re=100 and 200. Another test case is to simulate the flow over a NACA0012 airfoil at Re=500 with different Mach number and angle of attack. Current results agree fairly well with other numerical results. The validation shows that the current viscous solver can be used for simulation of laminar viscous flows under various wall boundaries and flow conditions.

Chapter 8
Conclusions and Recommendations

8.1 Conclusions

In this thesis, novel simulation techniques, namely a flux correction-based immersed boundary method (FC-IBM), a modified local domain-free discretization method (LDFD) and a local DFD-based immersed boundary method (LDFD-IBM), were introduced and investigated to simulate compressible inviscid flows in a Cartesian grid-based adaptive Euler solver. For the first time, we are able to implement the widely adopted IBM concept for incompressible flows to solve compressible inviscid flows. The three new methods have been validated and tested comprehensively in the present 2D adaptive Euler solver. The FC-IBM and LDFD method have been investigated in the extended 3D adaptive Euler solver as well. Validation and numerical test results clearly demonstrate the good potential in practical application to solve 3D compressible inviscid flows with complex boundaries. Some conclusions are drawn from the present study.

8.1.1 Development of adaptive Euler solver

An adaptive Euler solver was developed with the finite-volume discretization on Cartesian grid. The HLLC scheme is used to compute the flux on cell interface and six second-order flux schemes were implemented and studied. The adaptive mesh refinement (AMR) technique is an important and useful

tool implemented on the present Euler solver. The ghost-cell method was adopted and integrated with the developed adaptive Euler solver to validate and benchmark against the adaptive solver. Through the validation studies, it has been demonstrated that the present adaptive Euler solver is accurate, efficient and robust in simulating compressible flows with strong and weak shock waves. The adaptive solver has built a concrete platform to this study.

8.1.2 Implementation of FC-IBM

A novel FC-IBM was proposed based on the notion that a wall boundary shall satisfy the conditions of no-penetration, zero mass flux and zero energy flux. In the FC-IBM implementation, the first condition is enforced by velocity correction while the remaining two conditions by a flux correction method. The advantage of the FC-IBM is that it avoids the tedious process to compute the boundary curvature and to identify the cells are in either fluid domain or solid domain. This makes the method be unique and simple in the implementation.

The FC-IBM has been implemented and integrated with the developed 2D adaptive Euler solver. It has been validated and studied through the simulations of supersonic flows over a circular cylinder, a wedge and a double-ellipse structure, transonic and subsonic flows over a NACA0012 airfoil. The results obtained from the new method agree well with the available literature data; however, some numerical viscosity was observed and may affect the eventual solution, especially in the presences of weak convection and small reverse pressure gradient.

8.1.3 Implementation of LDFD and LDFD-IBM

A modified local LDFD (LDFD) method was implemented in the developed adaptive Euler solver. The concept of LDFD for 2D flows is that the wall boundary condition can be enforced by correcting the flow information on the solid cells (DFD cells) next to the boundary in X direction and Y direction directly for 2D flows. Because the DFD cells near the boundary can be identified easily in X and Y directions, the method is simple to implement. The LDFD-based immersed boundary method (LDFD-IBM) was proposed to make the implementation simpler by avoiding the need to identify the solid DFD cells and fluid DFD cells.

Validation and numerical tests for supersonic, hypersonic and transonic flows proved that both methods can handle simulations of various compressible inviscid flows. In particular, the LDFD has been demonstrated to be more accurate and robust. The successful numerical tests for subsonic flow and transonic flow over two different two-element airfoils by LDFD method clearly demonstrated the potential of this tool for the simulation of compressible inviscid flows with complex boundaries.

8.1.4 Development of 3D adaptive Euler solver

The 3D adaptive Euler solver was successfully developed in the platform of 2D adaptive solver. The proposed immersed boundary methods, FC-IBM and LDFD, were implemented and integrated with the 3D adaptive solver. Because of the high demand for longer computing time and the increasing complexity for boundary recognition and implementation, the present 3D adaptive solver

with FC-IBM and LDFD have been validated and benchmarked primarily with supersonic flows over bluff body wall boundaries and complex 3D space vehicle consisting of curved body and wings. The results have been compared and validated with Fluent solver. The validation results and the benchmark performance demonstrated to certain extent the viability of the solver and the proposed methods for compressible inviscid flows.

8.1.5 Application for laminar viscous flow

The adaptive Euler solver has been extended to simulate laminar viscous flows simply by considering the flux contribution from the viscous stresses. Minor effort is required to convert the inviscid Euler solver to a laminar viscous flow solver. The FC-IBM was simply modified to enforce the no-slip boundary condition for viscous flows. Simulation of laminar viscous flows over a circular cylinder and a NACA0012 airfoil gave rise to an accurate derivation of the viscous solver in simulating steady-state and unsteady-state flows.

8.2 Recommendation for future works

Further studies and possible improvements under consideration on the proposed immersed boundary methods, FC-IBM, LDFD and LDFD-IBM, for compressible inviscid flow simulation using the developed Cartesian-grid based adaptive solver include:

- The study of the numerical viscous effect introduced during the implementation of FC-IBM. Because the delta function interpolation

only offers the first-order accuracy, there is a possibility that the numerical viscous effect may be introduced during the interpolation process for the computation of the flux and normal velocity on the wall boundary. Interesting conclusions may surface from higher-order interpolation methods.

- The implementation and study of other boundary conditions and their impact. In compressible inviscid flows, the choice of conditions for wall boundary is not limited to those used in this study. Other conditions such as constant enthalpy and adiabatic wall can be implemented as well for comparative studies.
- The improvement and enhancement of the implementation of the FC-IBM and LDFD method in 3D solver. The 3D adaptive solver can be extended to simulate more challenging 3D flows with complex boundaries, such as that of an airfoil wing, a space shuttle and even an entire aircraft.
- The improvement and enhancement of the adaptive viscous flow solver for the application of unsteady state flows with moving boundaries.
- The parallelization of the 3D solver by making use of the multiple core computing facilities. Naturally, the availability of multiple-core computers can greatly enhance the performance and efficiency of the 3D adaptive solver introduced here.

Bibliography

[1]. M.J. Aftosmis. Solution adaptive Cartesian grid methods for aerodynamics flows with complex geometries. von Karman Institute for Fluid Dynamics, March 1997, Lecture series, 1997-02.

[2]. W.J. Coirier and K.G. Powell. An accuracy assessment of Cartesian-Mesh approaches for the Euler equations. *Journal of Computational Physics*, Volume 117, (1995), 121-131.

[3]. R.B. Pember, J.B. Bell, P. Colella, W.Y. Crutcheield and M. L. Welcome. An adaptive Cartesian grid method for unsteady compressible flow in irregular regions. *Journal of Computational Physics*, Volume 120, (1995), 278-304.

[4]. D.D. Zeeuw and K.G. Powell. An adaptively refined Cartesian mesh solver for the Euler equations. *Journal of Computational Physics*, Volume 104, (1993), 58-68.

[5]. H. Forrer and R. Jeltsch. A higher-order boundary treatment for Cartesian-grid methods. *Journal of Computational Physics*, Volume 140, (1998), 259-277.

[6]. A. Dadone and B. Grossman. Ghost-cell method for inviscid two-dimensional flows on Cartesian grids. *AIAA Journal*, Volume 42, (2004), 2499-2507.

[7]. A. Dadone and B. Grossman. Ghost-cell method for analysis of inviscid three-dimensional flows on Cartesian-grids. *Computers & Fluids*, Volume 36, (2007), 1513-1528.

[8]. A. Dadone and B. Grossman. Ghost-cell method with far field coarsening and mesh adaptation for Cartesian grids. *Computers & Fluids*, Volume 35, (2006), 676-687.

[9]. X. Jiang, Z. Chen and H. Li. Numerical investigation on the interaction of cylinder and shockwave based on the immersed boundary method. *Modern Physics Letters B*. Volume 23, (2009), 317-320.

[10]. J. Liu, N. Zhao and O. Hu. The ghost cell method for inviscid compressible flow on adaptive tree Cartesian grids. *AIP Conference Proceedings*. The 2nd International ISCM Symposium and the 12th International EPMESC Conference, (2010), 759-763.

[11]. H. Luo, J.D. Baum and R. Löhner. A hybrid Cartesian grid and gridless method for compressible flows. *Journal of Computational Physics*, Volume 214, (2006), 618-632.

[12]. X. Liang and G. Yang. Euler solution using adaptive Cartesian grid with a gridless boundary treatment. *Acta Mechanica Sinica*. Volume 25, (2009), 187-196.

[13]. A. Jahangirian and M.Y. Hashemi. Adaptive Cartesian Grid with mesh-less zones for compressible flow calculations. *Computers & Fluids*, Volume 54, (2012), 10-17.

[14]. C. Shu and L.F. Fan. A new discretization method and its application to solve incompressible Navier-Stokes equation. *Computational Mechanics*, Volume 27, (2001), 292-301.

[15]. C. Shu and Y.L. Wu. Domain-free discretization method for doubly connected domain and its application to simulate natural convection in eccentric annuli. *Computer Methods in Applied Mechanics and*

Engineering. Volume 191, (2002), 1827-1841.

[16]. C. Shu and Y.L. Wu. Adaptive mesh refinement-enhanced local DFD method and its application to solve Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, Volume 51, (2006), 897-912.

[17]. Y.L. Wu and C. Shu. Application of local DFD method to simulate unsteady flows around an oscillating circular cylinder. *International Journal for Numerical Methods in Fluids*, Volume 58, (2008), 1223-1236.

[18]. C.H. Zhou, C. Shu and Y.Z. Wu. Extension of domain-free discretization method to simulate compressible flows over fixed and moving bodies. *International Journal for Numerical Methods in Fluids*. Volume 53, (2007), 175-199.

[19]. C.H. Zhou and C. Shu. A local domain-free discretization method to simulate three-dimensional compressible inviscid flows. *International Journal for Numerical Methods in Fluids*. Volume 61, (2009), 970-986.

[20]. C.S. Peskin. Flow patterns around heart valves: a numerical method. *Journal of Computational Physics*, Volume 10, (1972), 252-271.

[21]. C.S. Peskin. A numerical analysis of blood flow in the heart. *Journal of Computational Physics*, Volume 25, (1977), 220-252.

[22]. C. Shu, N. Liu and Y.T. Chew. A novel immersed boundary velocity correction-lattice Boltzmann method and its application to simulate flow past a circular cylinder. *Journal of Computation Physics*, Volume 226, (2007), 1607-1622.

[23]. D. Goldstein, R. Hadler and L. Sirovich. Modeling a no-slip flow boundary with an external force field. *Journal of Computational Physics*. Volume 105, (1993), 354-366.

[24]. M. Lai and C.S. Peskin. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Journal of Computational Physics*. Volume 160, (2000), 705-719.

[25]. M.N. Linnick and H.F. Fasel. A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains. *Journal of Computational Physics*. Volume 204, (2005), 157-192.

[26]. J. Deng, X.M. Shao and A.L. Ren. A new modification of the immersed-boundary method for simulating flows with complex moving boundaries. *International Journal for Numerical Methods in Fluids*, Volume 52, (2006), 1195–1213.

[27]. Y. Sui, Y.T. Chew, P. Roy and H.T. Low. A hybrid immersed-boundary and multi-block lattice Boltzmann method for simulating fluid and moving-boundaries interactions. *International Journal for Numerical Methods in Fluids*, Volume 53, (2007), 1727-1754.

[28]. X. Shi and S. P. Lim. A LBM-DLM/FD method for 3D fluid-structure interactions. *Journal of Computational Physics*, Volume 226, (2007), 2028-2043.

[29]. I. Borazjani, L. Ge and F. Sotiropoulos. Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies. *Journal of Computational Physics*, Volume 227, (2008), 7587–7620.

[30]. D. Pan and T.T. Shen. Computation of incompressible flows with

immersed bodies by a simple ghost cell method. *International Journal for Numerical Methods in Fluids*, Volume 60, (2008), 1378-1401.

[31]. J. Wu, C. Shu and Y. H. Zhang. Simulation of incompressible viscous flows around moving objects by a variant of immersed boundary-lattice Boltzmann method. *International Journal for Numerical Methods in Fluids*, Volume 62, (2009), 327-354.

[32]. J. Wu and C. Shu. Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its application. *Journal of Computational Physics*. Volume 228, (2009), 1963-1973.

[33]. C. Shu, N. Liu, Y.T. Chew and Z. Lu. Numerical simulation of fish motion by using Lattice Boltzmann-immersed boundary velocity correction method. *Journal of Mechanical Science and Technology*, Volume 21, (2007), 1352-1358.

[34]. J. Wu and C. Shu. Numerical study of flow characteristics behind a stationary circular cylinder with a flapping plate. *Physics of Fluids*, Volume 23, (2011).

[35]. E.F. Toro. Riemann Solvers and Numerical Methods for Fluid Dynamics – A Practical Introduction. 2nd Edition. Springer, (1999), 249-264.

[36]. H. Jun, S. Guo and Z. Yao. Solution to Euler equations by high-resolution upwind compact scheme based on flux splitting. *International Journal for Numerical Methods in Fluids*. Volume 56, (2008), 2139-2150.

[37]. H.W. Zheng, C. Shu and Y.T. Chew. An object-oriented and quadrilateral-mesh based solution adaptive algorithm for compressible

multi-fluid flows. *Journal of Computational Physics*, 227, (2008), 6895-6921.

[38]. C. Sun. Simulations of compressible flows with strong shocks by an adaptive Lattice Boltzmann Model. *Journal of Computational Physics*. Volume 161, (2000), 70-84.

[39]. Z.J. Wang and Y. Liu. The spectral difference method for the 2D Euler equations on unstructured grids. *AIAA paper 2005-5112*, (2005).

[40]. P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics*. Volume 54, (1984), 115-173.

[41]. R. Hillier. Computation of shock wave diffraction at a ninety degrees convex edge. *Shock Waves*, Volume 1, (1991), 89-98.

[42]. G.H. Tu and X.J. Yuan. A characteristic-based shock-capturing scheme for hyperbolic problems. *Journal of Computational Physics*. Volume 225, (2007), 2083-2097.

[43]. K. Qu. Development of Lattice Boltzmann method for compressible flows. *Ph.D Thesis*, National University of Singapore. (2008).

[44]. M.R. Visbal and D.V. Gaitonde. Shock capturing using compact-differencing-based methods. *AIAA paper 2005-1265, 43rd AIAA Aerospace Sciences Meeting and Exhibit*, 10-13, January 2005, Reno, Nevada, (2005).

[45]. H. Luo, J.D. Baum and R. Löhner. A fast, matrix-free implicit method for compressible flows on unstructured grids. *Journal of Computational Physics*, Volume 146, (1998), 664-690.

[46]. B. Sjögreen and N.A. Petersson. A Cartesian embedded boundary

method for hyperbolic conservation laws. *Communications in Computational Physics*. Volume 2, (2007), 1199-1219.

[47]. C. Shu, H. Ding, H.Q. Chen and T.G. Wang. An upwind local RBF-DQ method for simulation of inviscid compressible flows. *Computer Methods in Applied Mechanics and Engineering*, Volume 194, (2005), 2001-2017.

[48]. D.W. Levy, K.G. Powell and B.V. Leer. Use of a rotated Riemann solver for the two-dimensional Euler equations. *Journal of Computational Physics*, Volume 106, (1993), 201-214.

[49]. P. Arminjon, M.C. Viallon and A. Madrane. A finite volume extension of the Lax-Friedrichs and Nessyahu-Tadmor schemes for conservation laws on unstructured grids. *International Journal of Computational Fluid Dynamics*. Volume 9, (1997), 1-22.

[50]. S.J. Lee, B.G. Cho and I. Lee. Two-dimensional unsteady aerodynamics analysis based on a multiphase perspective. Computers & Fluids, Volume 53, (2012), 105-116.

[51]. H. Yoshihara and P. Sacher. Test cases for inviscid flow field methods. *AGARD AR-211*, 1986.

[52]. B. Fiorina and S. K. Lele. An artificial nonlinear diffusivity method for supersonic reacting flows with shocks. *Journal of Computational Physics*, Volume 222, (2007), 246-264.

[53]. F. Bramkamp, Ph. Lamby and S. Müller. An adaptive multiscale finite volume solver for unsteady and steady state flow computations. *Journal of Computational Physics*. Volume 197, (2004), 460-490.

[54]. K.W. Morton and M.F. Paisley. A finite volume scheme with shock

fitting for the steady Euler equations. *Journal of Computational Physics*. Volume 80, (1989), 168-203.

[55]. C.H. Liu and Y. Li. Turbulence modeling for computing viscous high-Reynolds-number flows on unstructured meshes. *Computer Methods in Applied Mechanics and Engineering*, Volume 190, (2001), 5325-5339.

[56]. URL online webpage: NLR 7301 Multi-Element Airfoil Calculation. *http://www.symscape.com/node/318#.UIqmDYbpUmc*, last valid in Oct 2012.

[57]. AGARD-AR-303, A selection of experimental test cases for the validation of CFD codes. NATO Advisory Group for Aerospace Research and Development. 1994.

[58]. AGARD-AR-138, Experimental data base for computer program assessment. NATO Advisory Group for Aerospace Research and Development. 1979.

[59]. M. Lahooti and A. Pishevar. A new fourth order central WENO method for 3D hyperbolic conservation laws. *Applied Mathemaitcs and Computation*. Volume 218, (2012), 10258-10270. .

[60]. F. Rispoli, R. Saavedra, F. Menichini and T.E. Tezduyar. Computation of inviscid supersonic flows around cylinders and spheres with the V-SGS stabilization and YZβ shock-capturing. *Journal of Applied Mechanics*. Volume 76, (2009), 021209-1~6.

[61]. J. Marcy. Computational fluid dynamics analysis of flow Over a re-Entry vehicle. *46th AIAA Aerospace Sciences Meeting and Exhibit*, (2008), eSIBN: 978-1-62410-128-1.

[62]. K. Taira and T. Colonius. The immersed boundary method: A

projection approach. *Journal of Computational Physics*. Volume 225, (2007), 2118-2137.

[63]. N. Ganesh, N.V. Shende and N. Balakrishnan. R-parameter: A local truncation error based adaptive framework for finite volume compressible flow solvers. *Computers & Fluids*, Volume 38, (2009), 1799-1822.

[64]. M.D. de Tullio, P. de Palma, G. Iaccarino, G. Pascazio and M. Napolitano. An immersed boundary method for compressible flows using local grid refinement. *Journal of Computational Physics*. Volume 225, (2007), 2098-2117.

[65]. S.J. Lee, B.G. Cho and I. Lee. Two-dimensional unsteady aerodynamics analysis based on a multiphase perspective. *Computers & Fluids*. Volume 53, (2012), 105-116.

[66]. X. Du, C. Corre and A. Lerat. A third-order finite-volume residual-based scheme for the 2D Euler equations on unstructured grids. *Journal of Computational Physics*. Volume 230, (2011), 4201-4215.

[67]. K. Karagiozis, R. Kamakoti and C. Pantano. A low numerical dissipation immersed interface method for the compressible Navier-Stokes equations. *Journal of Computational Physics*. Volume 229, (2010), 701-727.

[68]. X. He and G. Doolen. Lattice Boltzmann method on curvilinear coordinates system: flow around a circular cylinder. *Journal of Computational Physics*. Volume 124, (1997), 306-315.

[69]. P.M. Gresho, R. Chan and C. Upson and R. Lee. A modified finite element method for solving the time-dependent incompressible Navier-

Stokes equations: part 2: applications. *International Journal for Numerical Methods in Fluids,* Volume 4, (1984), 619-640.

[70]. E.M. Saiki and S. Biringen. Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method. *Journal of Computational Physics*. Volume 123, (1996), 450-465.

[71]. C. Liu, X. Zheng and C.H. Sung. Preconditioned multigrid methods for unsteady incompressible flows. *Journal of Computational Physics*. Volume 139, (1998), 35-57.

[72]. M. Nemec and M. J. Aftosmis. Adjoint error estimation and adaptive refinement for embedded-boundary Cartesian meshes. *18th AIAA Computational Fluid Dynamics Conference, June 25-28,* (2007), AIAA paper 2007-4187.

[73]. M. Nemec and M. J. Aftosmis. Adjoint sensitivity computations for an embedded-boundary Cartesian mesh method. *Journal of Computational Physics,* Volume 227, (2008), 2724-2742.

[74]. M. F. Barad, P. Colella and S. G. Schladow. An adaptive cut-cell method for environmental fluid mechanics. *International Journal for Numerical Methods in Fluids,* Volume 60, (2009), 473-514.

[75]. M. Vanella, P. Rabenold and E. Balaras. A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid-structure interaction problems. *Journal of Computational Physics,* Volume 229, (2010), 6427-6449.