LOCAL DOMAIN-FREE DISCRETIZATION METHOD AND ITS COMBINATION WITH IMMERSED BOUNDARY METHOD FOR SIMULATION OF FLUID FLOWS

WU YANLING

NATIONAL UNIVERSITY OF SINGAPORE

2012

LOCAL DOMAIN-FREE DISCRETIZATION METHOD AND ITS COMBINATION WITH IMMERSED BOUNDARY METHOD FOR SIMULATION OF FLUID FLOWS

WU YANLING

(*B.Eng,NUAA, M.Eng, NUS*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2012

Declaration

I hereby declare that this 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.

Wu Yanling

25, July,2012

Acknowledgements

I would like to express my deepest gratitude and thank to my supervisor, Professor Shu Chang, for his invaluable guidance, constant encouragement and great patience throughout this study.

I am extremely grateful to my husband, my son, and my whole family, their support and encouragement made it possible for me to complete the study.

Thanks also go to the staff of Department of Mechanical Engineering for their excellent service and help.

Finally, I would like to thank all my friends who have helped me in different ways during my whole period of study in NUS.

Wu Yanling

Table of Contents

Chapter 6 Extension of LDFD-IBM to Simulate Three-dimensional Flows with Complex Boundary

Summary

Numerical simulation of flows with complex geometries and/or moving boundaries is one of the most challenging problems in Computational Fluid Dynamics (CFD). In this thesis, two new non-conforming-mesh methods, Local Domain-Free Discretization (LDFD) method and hybrid LDFD and Immersed Boundary Method (LDFD-IBM), are proposed to solve this problem.

The concept of LDFD method is based on the mathematical fact that the solution inside the domain can be extended locally to the outside of the domain. With this idea, we can solve the problems with complex geometries on a non-conforming structured mesh. The boundary conditions of the embedded boundaries are enforced by a local extrapolation process, which determines the values of flow variables at the mesh nodes that are adjacent to embedded boundaries but locate on the outside of flow field. This treatment allows the LDFD method to flexibly handle flow problems with complex geometry.

LDFD-IBM is a delicate combination of LDFD method and Immersed Boundary Method (IBM), and enjoys the merits of both methods. For example, the penetration of streamlines into solid objects in the conventional IBM, due to inaccurate satisfaction of no-slip boundary conditions, can be avoided by using the LDFD method. On the other hand, the treatment of boundary condition for pressure at the solid boundary in the LDFD method, which is not a trivial task, is no longer necessary after introducing IBM. Through various numerical tests, LDFD-IBM is shown to be a simple and accurate solver. In this work, the LDFD-IBM is restricted to incompressible flow simulations while the LDFD method is applied to both the incompressible and compressible flow simulations.

Capability of handling moving boundary problems is also an important feature of LDFD and LDFD-IBM. In this thesis, we present different strategies of LDFD and LDFD-IBM for moving boundary problems. The performance of both methods has been carefully checked by numerical experiments. Comparison against the results available in the literature shows that both methods are able to solve moving boundary problems accurately and efficiently.

In principle, the two methods can be applied to any type of mesh. The mesh strategies are closely related to the computational efficiency. Uniform Cartesian mesh is simple and straightforward. But it is not computationally efficient, particularly for three-dimensional cases. Different mesh strategies such as Adaptive Mesh Refinement (AMR) (for 2D simulations), non-uniform mesh (for 3D simulations), and combination of AMR mesh and uniform mesh (for 3D simulations) are presented and they appear to work well with the two methods.

A variety of flow problems have been solved using the two methods, including incompressible and compressible flows with single or multiple bodies either in rest or in motion, with or without heat transfer. Numerical experiments show that the LDFD method and LDFD-IBM are effective tools for the computation of flow problems.

List of Tables

List of Figures

Nomenclature

a	Speed of sound
$\mathbf{A}$	Jacobian matrix
$\widetilde{\mathbf{A}}$	Approximate Jacobian matrix by Roe-average variables
C_D	Drag coefficient
C_L	Lift coefficient
C_p	Pressure coefficient
d	Dimension
D	Diameter of the cylinder
e	Specific internal energy
E	total energy
F	Flux
F_D	Drag force
F_L	Lift force
G	New form of the flux
g	Gravitational acceleration
h	Mesh spacing on the uniform mesh
I	Unit tensor
k	Thermal conductivity
$\overline{k}_{eqi}$, $\overline{k}_{eqo}$	Average equivalent conductivity for inner cylinder and outer cylinder
$\mathbf{L}$	Left eigenvalue of $\widetilde{\mathbf{A}}$
M	Mach number
N	Total number of nodes in the domain

$\vec{n}$	Normal direction to the solid wall
$\overline{Nu}_i, \overline{Nu}_o$	Average Nusselt number on the inner cylinder and outer cylinder
p	Static pressure
q	Local heat transfer rate
Q	Net flux out of a cell
R	Radius of the cylinder
R	Right eigenvalue of $\tilde{\mathbf{A}}$
Pr	Prandtl number, $\mathrm{Pr} = \mu C_p / k$
rr	Radius ratio, $rr = \overline{R}_o / \overline{R}_i$
$\overline{R}_i, R_i$	Dimensional and non-dimensional radii of inner cylinder
$\overline{R}_o, R_o$	Dimensional and non-dimensional radii of outer cylinder
Ra	Rayleigh number, $Ra = C_p \rho_0 g \beta L^3 (T_i - T_o) / kv$
Re	Reynolds number $\mathrm{Re} = \dfrac{\rho UL}{\mu}$
T	Temperature
T	Transverse gap between the two cylinders
T_i, T_o	Dimensionless temperature on the inner and outer cylinder
S	Area of interface of the control volume
St	Strouhal number
t	time
$\vec{t}$	tangential direction of solid body

U	Reference velocity
$\mathbf{W}$	Vector of conservative variables in Euler equation
$\widetilde{\mathbf{W}}$	Roe-average variables
x	Cartesian coordinate or global vector of unknowns
y	Cartesian coordinate of global vector of unknowns
β	Thermal expansion coefficient
Γ	Surface of contour $\partial\Gamma$
γ	Adiabatic exponent or ratio of specific heats
λ	Approximation coefficient
Δt	Time step
ε	Eccentricity
$\|\varepsilon\|$	Error norm
Φ	Dissipation function
τ	Stress tensor
φ_0	Angular position
ψ	Stream function
μ	Viscosity
ρ_0	Reference density
ϕ	Slope limiter
ν	Kinematical viscosity
ω	Vorticity

$\widetilde{\mathbf{\Lambda}}$	Eigenvalue matrix of $\widetilde{\mathbf{A}}$

Superscripts

n	The nth iteration

Subscripts

i	Index of x direction
j	Index of y direction
k	Index of z direction
L	Left side of cell face
R	Right side of cell face

Acronyms

AMR	Adaptive Mesh Refinement
DFD	Domain-Free Discretization method
DQ	Differential Quadrature method
FD	Finite Difference method
FE	Finite Element method
FV	Finite Volume method
LDFD	Local Domain-Free Discretization method
LDFD-IBM	Local Domain-Free Discretization and Immersed Boundary Method combination
RBF	Radial Basis Function

CHAPTER 1

Introduction

1.1 Background

Partial Differential Equations (PDE) have been widely used to mathematically describe phenomena in nature and engineering. Navier-Stokes equations are the set of PDEs to particularly model the dynamics of fluid flows, heat and mass transfer, and are the core of Computational Fluid Dynamics (CFD).

Basically, there are two ways to obtain the solution of PDEs. The first way is analytical method which pursues an analytical expression for the solution, and the obtained solution is exact at any location in the solution domain. The other way is numerical method which seeks an approximate solution (called numerical solution) for a given PDE. It is indicated that the numerical method is usually applied when the analytical solution of a PDE is difficult to obtain. Although the analytical and numerical methods can both give the solution of a PDE, they involve quite different solution procedures.

1.1.1 Analytical method

Analytical solution of PDEs is obtained by mathematical analysis. It satisfies the PDE and the given initial/boundary condition exactly.

To describe how to get the analytical solution, an example is used to demonstrate it in detail. We consider a one-dimensional differential equation in the Cartesian coordinate system

$$\frac{d^2\psi}{dx^2}+\frac{1}{x}\frac{d\psi}{dx}=9x \tag{1.1}$$

The physical domain for equation (1.1) is within $0.5 \le x \le 1$. Eq. (1.1) is subjected to the following boundary conditions

$$\psi\big|_{x=0.5}=-\frac{7}{8}, \qquad \psi\big|_{x=1}=0 \tag{1.2}$$

Firstly, we solve equation (1.1) solely and obtain the general solution of equation (1.1) as

$$\psi = x^3 + C_1 \ln x + C_2 \tag{1.3}$$

Secondly, the two constants in equation (1.3) are determined by substituting equation (1.3) into equation (1.2), which gives

$$C_1 = 0,\ C_2 = -1 \tag{1.4}$$

Thus, the exact solution of the problem can be written as

$$\psi = x^3 - 1 \tag{1.5}$$

Obviously, the solution given by equation (1.5) can be applied in the whole domain $0 \le x \le \infty$, which is much larger than the solution domain $0.5 \le x \le 1$.

It is found that the analytical solution is usually obtained by two steps. In the first step, a general solution is obtained. This process is only based on the given PDE, and neither the domain boundary nor the boundary conditions attached is involved. In the

second step, the expression of the general solution is substituted into the boundary conditions to determine the unknown coefficients in the general solution. Obviously, the first step does not consider the solution domain. The solution domain (geometry of the problem) is only involved in the second step in which the boundary condition is implemented. In general, the analytical method can be equally applied to problems with regular or irregular geometries. Furthermore, the solution obtained can be used to calculate the exact function values of the problem anywhere as long as the PDEs hold, no matter whether the position is inside the domain or not.

1.1.2 Numerical method

The numerical method seeks the approximate solution of the PDEs. The approximations are applied to get values of functions or derivatives at discrete locations in space and time. As a result, a system of algebraic equations or difference equations are then obtained, which can be solved numerically on the computer. In the numerical method, the function or derivative approximation, known as discretization, can take many different forms (named discretization methods). Some commonly used discretization methods in CFD are: Finite Difference (FD) method, Finite Volume (FV) method, Finite Element (FE) method, spectral method, Differential Quadrature (DQ) method. In terms of accuracy, these discretization methods can be classified as low order methods (FD,FV,FE) and high order methods (Spectral method and DQ). A brief review of these methods is given below.

- **Finite Difference (FD) method**

FD method is based on the Taylor series expansion (Strikewerda 2004). In FD method, the derivatives of the PDEs are replaced by the appropriate difference formula, giving an equation that consists solely of the values of variables at the present node and its neighbors. There are many difference formulae available, such as *forward difference*, *backward difference*, and *central difference*. The FD method is very simple and effective on structured grids, on which it can be especially easy to build higher-order schemes. However, it does not adapt well to problems with complex geometry without appropriate coordinate transformation.

- **Finite Element (FE) method**

The FE method is based on the weak form of PDEs (Zinkiewicz,1977). The domain is divided into a set of discrete volumes or finite elements, which are unstructured in general. The ability to handle arbitrary geometries makes FE method very attractive. However, it requires more coding work to maintain detailed nodal and element-based connectivity information of the computational mesh, and more computational operation to solve the relevant "stiffness" matrices of the linearized equations which are not as well structured as those for regular grids.

- **Finite Volume (FV) method**

FV method is based on the integral form of the conservation equations (Ferziger and Peric,2002) . The solution domain is subdivided into so called control volumes (CVs),

and the conservation equations are applied to each CV. Since the conservation properties are satisfied on a discrete level in FV method, it is very desirable for the capture of shock wave and/or other discontinuities. In principle FV methods can be implemented on any type of grid, including unstructured mesh, so it can deal with complex geometries. On the other hand, it suffers the same drawback of FE method on unstructured mesh: high maintenance cost of the unstructured mesh data base. Besides, it is expensive and difficult to develop schemes of order higher than second in accuracy.

- **Spectral method**

Spectral method can be viewed as an extension of the finite element methods (Canuto, 1988). The idea is to consider the solution of the PDE as a sum of certain "basis functions". Spectral method is generally used as a high order method, which means it can generate solutions with high order accuracy. However, it is usually less flexible than the low-order methods and is usually applied to incompressible flow simulations.

- **Differential Quadrature (DQ) method**

DQ method is another high-order method. In DQ method, a partial derivative of a function with respect to a coordinate direction is expressed as a linear weighted sum of the functional values at all mesh nodes along that direction (Bellman et al 1971,1972). DQ method can get accurate numerical solutions by using very few grid

nodes. However, like FD method, the DQ method can only be applied to structured grid.

These numerical methods share one thing in common, i.e. the solution of the PDE is directly coupled with the boundary conditions. In other words, to the contrary of two-steps procedure in seeking analytical solution described in the last section, the numerical solution is obtained in just one step. In this step, the PDE is discretized on the solution domain with proper implementation of the boundary condition. We can see clearly that the numerical discretization of the PDE by a numerical method is problem-dependent. Because even the governing equations are the same, if the boundary conditions or geometries are different, the discretization will give different expressions.

Due to this feature, some numerical methods can only be applied to regular domain problems. Examples are the FD method and the global method of differential quadrature (DQ). These methods discretize the derivatives in a PDE along a mesh line. Thus, they require the computational domain to be regular or be a combination of regular sub-domains. When a problem with complex geometry is considered, the boundary of the problem may not coincide with the mesh line. To apply the finite difference and DQ methods, one has to do the coordinate transformation, which maps the irregular physical domain to a regular domain in the computational space. In the computational space, the FD schemes and the DQ method can be directly applied

since in this space, the solution domain is regular. To do numerical calculation in the computational space, we need to transform the governing PDEs and their boundary conditions into the relevant forms in the computational space. This process is very complicated, and problem-dependent. In addition, it may bring additional errors into the numerical computation. For many years, researchers developed a variety of numerical methods, which attempt to avoid the complicated coordinate transformation process.

Unstructured mesh is one of the remedies for the complex geometries, as being adopted in the FV and FE methods. The elements or control volumes may have any shape, and there is no restriction on the number of neighboring elements or nodes. But the advantage of flexibility is offset by the disadvantage of the irregularity of the data structure. Node locations and neighbor connections need be specified explicitly. The matrix of the algebraic equation system no longer has regular, diagonal structure, and the band width needs to be reduced by reordering of the points. The solvers for the algebraic equation systems are usually slower than those for regular grids. And the grid generation and pre-processing are also very time-consuming. The other drawback of using unstructured mesh in FV method is that high order accuracy is difficult to achieve.

To overcome the drawbacks of conventional numerical methods which strongly couple the PDE with the solution domain, a domain-free discretization method was

proposed by Shu and his co-workers (Shu and Fan 2001, Shu and Wu 2002).

1.2 Domain-Free Discretization (DFD) method

1.2.1 The concept of the Domain-Free Discretization

From Section 1.1.1, the inspiration from analytical method is:

- the PDE and its solution domain can be treated separately;
- the solution obtained satisfies the PDE for both the points inside the domain and the points outside the domain.

Therefore, the basic idea of Domain Free Discretization is:

(1) The governing equations can be applied to anywhere in the flow domain, no matter where the point is located inside the fluid domain or outside of the fluid domain. Since the discretization is just the process of transferring governing equations into discrete form, the discrete form of the given differential equation is irrelevant of solution domain. Therefore there is no need to introduce coordinate transformation.

(2) After discretization, the discrete form of PDE may involve some points outside the given flow domain. The functional values at those points can be obtained by seeking the approximate form of solution from its neighboring nodes.

To demonstrate the above, let us take Eq. (1.1) as an example again, and this time, we use a numerical method. When we apply the second order finite difference scheme to equation (1.1), we can obtain the following discrete form

$$\frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\Delta x^2} + \frac{1}{x_i}\frac{\psi_{i+1} - \psi_{i-1}}{2\Delta x} = 9x_i \tag{1.6}$$

Note that the error between equation (1.6) and equation (1.1) is in the order of $(\Delta x)^2$. In conventional numerical techniques, equation (1.6) only involves functional values at grid points within the solution domain. In other words, when equation (1.6) is applied in the solution domain of $0.5 \le x \le 1$, the difference between its left and right sides is in the order of $(\Delta x)^2$. Now, it is interesting to see that when equation (1.6) is applied at a point outside the domain, $0.5 \le x \le 1$, the difference between its left and right sides is still in the order of $(\Delta x)^2$. Consider a point at $x_i = 2$ and take Δx as 0.1. From equation (1.5), we have,

$$\psi_i = 7, \qquad \psi_{i+1} = 8.261, \qquad \psi_{i-1} = 5.859 \tag{1.7}$$

Substituting equation (1.7) into the left and right sides of equation (1.6) respectively gives

$$\frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\Delta x^2} + \frac{1}{x_i}\frac{\psi_{i+1} - \psi_{i-1}}{2\Delta x} = 18.005 \tag{1.8a}$$

$$9x_i = 18 \tag{1.8b}$$

Equation (1.8) shows that when a discrete form of PDE is applied at a point outside the given physical domain, the truncation error remains the same order as its application at an internal point. This finding states a very important fact: the discrete form of a PDE resulted from numerical discretization is not restricted by the geometry of the problem. In other words, the discrete form of PDEs can involve some points outside the solution domain or some points which do not coincide with grid nodes inside the domain.

In the example shown above, the functional values at outside points are computed from the expression of analytical solution. However, the analytical solution of a problem is not easily obtained. On the other hand, numerical approximation techniques allow us to find the functional values in its approximate form in part of the given domain. At this stage, the boundary condition can be enforced and be used to determine the functional values at the point outside of flow domain.

The essential part of the DFD method is that governing equation and the boundary condition can be treated separately. When discretizing the governing equation, one does not need to consider the geometry of the boundary and the boundary condition attached on it. In general, the Domain-Free Discretization strategy can be used along with any discretization method, such as FD, FV, FE, DQ, etc.

The boundary is considered only when the function approximation is implemented for those discrete points located outside of the computational domain or those which do not coincide with the mesh nodes. The key procedure in the DFD method is how to evaluate the functional values at those points with the boundary condition fitting in.

1.2.2 The procedure of DFD method

The following example demonstrates how DFD method solves a PDE, as shown in Fig.1.1. Suppose that we are seeking the solution $\psi(x, y)$ of a Poisson problem in the Cartesian coordinate system

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = f(x, y) \tag{1.9}$$

First, the computational domain is divided by a set of mesh lines $x_i, i = 1,2,...,N$. Second, mesh nodes are distributed along each x_i line. It should be noted here that the mesh nodes for each x_i line can be arbitrary. Therefore, different number of mesh nodes can be distributed along different mesh lines. For example, M_i nodes along x_i line and M_j nodes along x_j line, as shown in Figure 1.1, could be different.

Then, the derivatives in PDE are discretized by numerical method. For example, the derivative related with x in the given PDE can be discretized by the FD scheme. For instance, the second-order derivative at mesh node A along the mesh line x_j in Fig.1.1 can be discretized by the non-uniform FD scheme as

$$\left.\frac{\partial^2 \psi}{\partial x^2}\right|_A = \frac{(\psi_{A''} - \psi_A)(x_j - x_i) - (\psi_A - \psi_{A'})(x_k - x_j)}{\frac{1}{2}(x_j - x_i)(x_k - x_j)(x_k - x_i)} \tag{1.10}$$

where A' and A'' are the neighbors of A in the x direction. Thus, the given PDE will be reduced to a set of ordinary differential equations (ODEs). The derivatives related with y ,i.e. $\frac{\partial^2 \psi}{\partial y^2}$ remained in the resultant ODEs can be further discretized by a numerical method such as DQ method (Shu, 2000),

$$\left.\frac{\partial^2 \psi}{\partial y^2}\right|_A = \sum_{l=1}^{M_j} br(l, j)\psi_{j,l} \tag{1.11}$$

where $br(l, j)$ are the DQ weighting coefficients of the second order derivatives along the mesh line $x = x_j$ for point A. Hence, the complete discrete form of Eq.(1.9) at mesh node A can be written as

$$\frac{(\psi_{A''}-\psi_A)(x_j-x_i)-(\psi_A-\psi_{A'})(x_k-x_j)}{\frac{1}{2}(x_j-x_i)(x_k-x_j)(x_k-x_i)}+\sum_{l=1}^{M_j} br(l,j)\psi_{j,l}=f(x,y) \tag{1.12}$$

This process can be applied to the all mesh nodes inside the flow domain (represented by black and white circles in Fig 1.1). As a consequence, a set of algebraic equations will be obtained.

The resulting algebraic equation system (1.12) may involve some points at neighboring lines, which may not be the mesh nodes, and can be inside or outside the physical domain (such as A' and A'', represented by black squares in Fig.1.1). The functional values at these points can be computed by one-dimensional interpolation/extrapolation technique. In Cartesian coordinate system, the most convenient way is to find the approximate solution along the mesh line at the specific point. For example, the functional value at the point A' which is outside of the flow domain can be obtained by extrapolation using mesh nodes $i1$,$i2$,$i3$ which locate the same mesh line x_i of A' but inside of the domain. Because node $i1$ is a boundary point, therefore, the boundary condition can be implemented to get the functional value at node $i1$. Only at this stage, the boundary condition comes into the picture. Similarly, the functional value at the point A'' which is inside of the flow domain could be calculated by interpolation with the mesh nodes at mesh line x_k . The point at which the interpolation is implemented is called “interpolation point” (such as A''), or “extrapolation point” (such as A') if the extrapolation is implemented there, as shown in Figure 1.1. Any approximation scheme can be used in DFD method to evaluate the

functional values at interpolation or extrapolation points, such as the Lagrange interpolation/extrapolation scheme (Shu and Fan, 2001), Radial Basis Function approximation (Wu et al, 2004), etc. It is found that although high order Lagrange polynomial-based scheme can give very accurate approximation at interpolation point, its performance at extrapolation points is not very good. This is because for extrapolation, the coefficients of high order Lagrange polynomials become very large and may introduce large numerical errors. Therefore, if polynomial-based collocation approximation scheme is adopted, only low order Lagrange polynomial is used for extrapolation, such as 3-points local Lagrange extrapolation (Shu and Fan, 2001).

1.2.3 The features of DFD method

- **Discretization without coordinate transformation procedure**

The DFD method treats the PDEs and the solution domain separately. Therefore, the coordinate transformation is totally avoided. Much pre-processing work involved in the coordinate transformation in conventional numerical discretization methods (such as FD method) is avoided. This feature of DFD brings great convenience in the practical usage. The derivatives in the PDEs can be discretized by any numerical discretization schemes used in CFD, and different discretization schemes can be used in different coordinate directions. For example, in the Cartesian coordinate system, the derivatives in the x direction can be discretized by the FD method, and the derivatives in the y direction can be discretized by the DQ method (Shu,2000) or the Radial Basis Function-DQ method or RBF-DQ method(Wu and Shu, 2002).

- **One-dimensional interpolation and extrapolation**

The implementation of the DFD method may involve some points which are not mesh nodes, and can be either within the domain or outside the domain. Because the mesh nodes are only distributed along the mesh line, approximation of function values at these "disorder" points is also carried out along the mesh line. Therefore, numerical approximation on these points is actually one-dimensional interpolation or extrapolation. Many numerical interpolation techniques can be used. In DFD method, one can either use global interpolation/extrapolation scheme (i.e., using all the mesh nodes along the mesh line as the supporting points, such as Lagrange polynomial interpolation or Radial Basis function interpolation), or use local interpolation/extrapolation scheme (such as 3-nodes local Lagrange extrapolation).

1.3 Classification of DFD method

According to approximation and discretization methods used, the DFD method can be classified into Global DFD method and Local DFD method.

1.3.1 Overview of Global DFD

As the name indicates, the Global DFD method uses global discretization method (such as DQ method, RBF-DQ method) in one or more coordinate directions to discretize the governing equation. They also use global interpolation/extrapolation scheme (such as Lagrange interpolated polynomial interpolation or global RBF approximation) to get the approximate solution at those points which may not

coincide with the mesh node, inside or outside the domain.

In the work of Shu and Fan (2001), the PDEs on the singly-connected domain are solved by the Global DFD method in the Cartesian coordinate system, and the approximate forms of solution are obtained along each vertical or horizontal line. In the work of Shu and Wu (2002), the PDEs on the doubly-connected domain are solved by the Global DFD method in the cylindrical coordinate system, and the approximate forms of solution are obtained along each radial line. Later, researchers reported that the Global DFD method can also be combined with other advanced numerical techniques to improve its computational efficiency, such as Multi-Grid acceleration (Wu et al, 2003) and Radial Basis Function approximation (Wu et al, 2004).

Due to the decoupling of the solution of governing equations with the boundary conditions, the DFD method can be used on any mesh types. Shu and Fan (2001) and Shu and Wu (2002) found that in DFD method, the number of mesh nodes along a mesh line can be arbitrary, which saves a lot of computational cost when one can distribute few number of mesh nodes in the narrow gap and make the mesh a totally unstructured one. However, unlike the unstructured mesh used in FV and FE methods, here we do not need to store the connectivity of the unstructured mesh. All we need to do is to calculate the functional values by interpolation/extrapolation at the locations where variable values are needed for the discretization.

In the applications (Shu and Fan 2001, Shu and Wu 2002, Wu et al 2003,2004), the Global DFD has demonstrated its high accuracy in solving PDEs such as Navier-Stokes equations on irregular domains. However, the Global DFD encounters two major difficulties:

(1) Global method may generate large numerical errors in some cases or make the iteration unstable. One example is that when the extrapolation point is far away from the boundary, the global form of approximate solution along the respective line would generate large extrapolation coefficients, causing large numerical errors. Another example is that if a global method such as Polynomial-based Differential Quadrature (PDQ) (Shu, 2000) is used in the discretization, the weighting coefficients of the derivatives in PDE become very large, posing a polynomial "snaking" problem which makes the convergence very slow.

(2) For multiply-connected (more than two) domain problems, it is difficult to get approximate form of solution along a line where more than two boundary points are involved by global approximation. Since different boundary conditions are applied to these boundary points, it is difficult to determine automatically which boundary points should be used to evaluate the functional values at the extrapolation points or interpolation points.

To remove these difficulties and make the method more general, the Local DFD approach is developed in this work.

1.3.2 Fundamentals of Local DFD

In the Local DFD method, the numerical work including discretization of derivatives and interpolation/extrapolation for approximate form of solution is made locally by using low order polynomials, so that only few neighboring nodes around the target node are involved in the discretization and the approximation. Similar to Global DFD method, Local DFD method can be used to any coordinate system and any mesh type. Due to the advantages of structured mesh (such as regularity of the data structure, low storage cost of node locations and neighbor connections, the matrix of the algebraic equation system has regular, diagonal structure; the grid generation is easy, etc.), in this thesis, we confine our discussion to the structured mesh. The simplest structured mesh is Cartesian mesh in the Cartesian coordinate system. Therefore, in this sense, the Local DFD method we are talking about in this thesis is a Cartesian mesh-based version, or a kind of Cartesian mesh solver. However, it should be noted that the Local DFD method can be moved beyond the Cartesian mesh, and can generate more different types of Local DFD solver.

1.4 Contributions and Organization of the dissertation

The important contributions of this work are listed as follows:

(1) Two new non-conforming-mesh methods, i.e., Local Domain-Free Discretization (LDFD) method and hybrid LDFD and IBM (LDFD-IBM, IBM means Immersed Boundary Method due to Peskin,1972) are presented. When implemented on Cartesian mesh, the two methods can be considered as Cartesian mesh approaches.

(2) Because in the LDFD and LDFD-IBM, a crucial step is to identify the status of mesh nodes. A fast algorithm of the status identification for complex solid boundaries encountered in engineering practice has been proposed based on the 2D and 3D Cartesian mesh, respectively.

(3) To overcome the drawback of using single uniform mesh, different mesh strategies are introduced in the applications of LDFD and LDFD-IBM to two- and three-dimensional flow problems, such as stencil Adaptive Mesh Refinement (AMR) (for 2D simulations, due to Ding and Shu, 2006), non-uniform mesh (for 3D simulations), and combination of AMR mesh and uniform mesh (for 3D simulations).

(4) The LDFD solver for incompressible flow problems is developed based on central difference scheme under Finite Difference framework, and LDFD Euler solver is developed based on Roe scheme under the Finite Volume framework.

(5) Two strategies of adding the force term of IBM in momentum equations are proposed in LDFD-IBM. In the first strategy, the force term is added implicitly via velocity updating at the dependent points. In the second strategy, a two-steps forcing procedure is presented to model the moving boundary problems.

(6) Due to its easy implementation, the LDFD-IBM is extended to simulate three-dimensional steady/unsteady flows with complex boundary.

(7) A variety of flow problems with dynamic or geometric complexity, have been solved using the two new numerical methods. Numerical results have compared well with the benchmark solutions. The comparative study is carried out in terms of the accuracy and the efficiency.

After the introduction, this thesis is organized as follows:

Chapter 2 gives a detailed description on the LDFD method based on Cartesian mesh. A fast algorithm of identifying the status of the mesh nodes in 2D case is also presented. The LDFD and the proposed numerical technique are validated by simulation of incompressible flows past a NACA0012 airfoil.

In **Chapter 3**, to make LDFD a powerful Cartesian mesh solver with solution adaptive refinement capability, the stencil Adaptive Mesh Refinement (AMR) is introduced into the LDFD. Numerical simulations of natural convection in the concentric annulus between inner circular and outer square cylinders are carried out to validate the AMR-enhanced LDFD method and to demonstrate the efficiency improvement.

In **Chapter 4**, the LDFD-IBM is presented in details. The advantages of this method include simpler and easier implementation than the original LDFD and higher accuracy as compared with conventional IBM, which are demonstrated by numerical tests. The LDFD-IBM is then applied to simulate flows past single/multiple cylinders.

Chapter 5 shows the procedures of the LDFD method and LDFD-IBM in handling the moving boundary problems in detail. For these two methods, the status-changing effect of the moving object is considered and the corresponding strategies are adopted.

The solvers are validated by the numerical simulations of flow past an oscillating cylinder and flow with two cylinders moving to each other.

Chapter 6 demonstrates the applicability of present LDFD-IBM to three-dimensional incompressible flow problems. A fast algorithm of identifying the status of the mesh nodes in 3D case is also presented. We adopt two meshing strategies to avoid huge computational costs on 3D uniform mesh, which are: (1) non-uniform mesh; (2) two-dimensional AMR in x-y plane and uniform mesh in z direction. The former is applied to 3D flow past a sphere and flow over a torus. The latter is used to simulate the 3D flow past a circular cylinder.

Chapter 7 presents the LDFD Euler solver based on Roe scheme under the framework of Finite Volume method. It is applied to simulate the compressible flows with different geometries and flow parameters. The obtained results are compared with available data in the literature and good agreements are observed.

Chapter 8 draws some conclusions and some recommendations are also given.

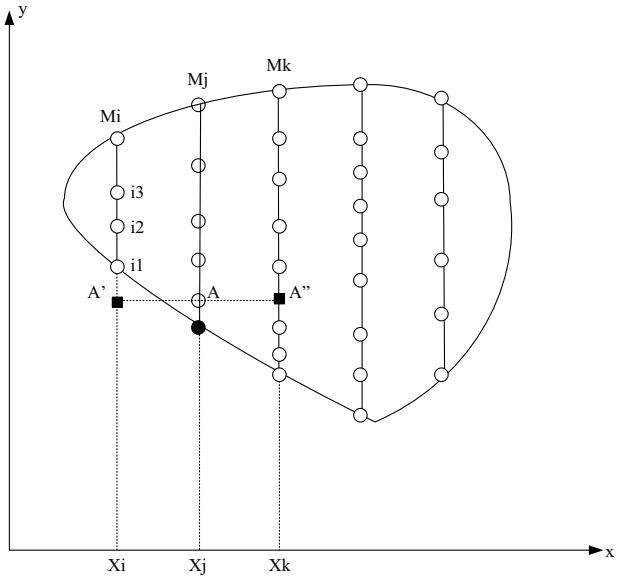


Figure 1.1 Configuration of the DFD method in Cartesian coordinate system with mesh, interpolation and extrapolation points

CHAPTER 2

Local Domain-Free Discretization Method

In this chapter, we present Local Domain-Free Discretization (LDFD) method, in which low-order polynomials are used to construct approximations of the derivatives and function values near the boundaries. Since the method is mainly implemented on a Cartesian mesh, in this sense, it belongs to the family of "Cartesian mesh methods". We firstly review the Cartesian mesh methods developed in the past decades, and compare LDFD with other Cartesian mesh solvers. The implementation procedure of LDFD for simulation of the incompressible flows with irregular geometry is then described in detail, including the fast algorithm of identifying status of mesh nodes and the treatment of various boundary conditions. The method is validated by numerical simulation of flows past an airfoil NACA0012 at different Reynolds number and different attack angles.

2.1 Cartesian mesh methods

The simplest mesh to generate is Cartesian mesh, in which the grid lines just follow the coordinate directions. When the geometry of the physical domain is regular, the spatial discretization of the governing equation is quite straightforward. However, most flows in engineering applications involve irregular geometries, which in general do not conform to the grid lines of the Cartesian mesh. In order to apply such a grid to solution domains with inclined or curved boundaries, a variety of Cartesian mesh methods have been proposed. Basically, they can be divided into the following four categories:

2.1.1 Saw-tooth boundary method

The simplest option for handling an irregular geometry on a Cartesian mesh is to approximate the boundaries by saw-tooth surface or staircase-like steps. This method yields a geometrical description of order one in mesh spacing. Chai and Patankar (1994) used this method to simulate the radiative heat transfer on an irregular geometry and obtained satisfactory results. This method was also adopted by Manhart and Wengle (1994) in their large eddy simulation of flow over a wall-mounted hemisphere. However, the steps at the boundary may bring errors into the solution, especially when the grid is coarse (Ferziger and Peric, 2002).

2.1.2 Immersed Boundary Method

The Immersed Boundary Method (IBM) (Peskin, 1998; Leveque et al 1994; Ye et al 1999; Udaykumar et al 2001,etc) includes an additional body force in the momentum equation to account for the presence of embedded objects. IBM is usually limited to the numerical simulation of incompressible viscous flows. A large number of Cartesian mesh methods relate themselves to IBM method. They have one feature in common, that is, the discretization is performed at every mesh node in the domain, although the particular treatments of the boundary conditions are quite different (Mittal & Iaccarino 2005, Fadlun et al 2000, Udaykumar et al 2001, Ye et al 1999, Niu et al 2006, Shu et al 2007, etc). The conventional IBM suffers from two major drawbacks: One is the streamline penetration to the body and the other is the low order of accuracy near the boundary.

2.1.3 Cut cell method

Another popular Cartesian mesh method involves cell cutting. This approach is normally based on the finite volume discretization on a Cartesian mesh, of which the embedded boundaries are "carved" out. Therefore, the complex geometries are represented by a set of cut cells which are usually in irregular shape. These cut cells require special attention and significantly increase the complexity for coding. For example, some cut cells could be very small. They may cause the computation to suffer from very small time steps, as requested by numerical stability. Many Cartesian mesh methods for compressible Euler equations belong to this type. To name a few, they are Cartesian grid embedded boundary method (Colella, 2006), hybrid cut-cell/ghost cell method(Ji et al 2008), and cut-cell method (Corier and Powell 1995).

2.1.4 Ghost Cell methods

Ghost cell methods employ the mesh cells immediately outside of the computational domain, i.e., “Ghost cells”, to implement the boundary conditions. Examples being Ghost Cell method (GCM) by Dadone and Grossman (2004) and Ghost Fluid method (GFM) proposed by Fedkiw et al (1999). GCM and GFM were originally designed to handle the discontinuities in compressible flow or two-phase flows. Taking the jump condition across the contact discontinuity into account, the pressure and normal velocity of the ghost point can be reasonably set identical to the pressure and normal velocity of their mirror point in real fluid, while the entropy and tangential velocity at the ghost point are extrapolated in the normal direction from the real fluid, which is located on the other side of the interface. Most applications are two- and three-dimensional simulations of compressible inviscid flows (Fedkiw et al 1999, Dadone and Grossman 2004,2007).

2.2 Comparison of LDFD method with other Cartesian mesh methods

As stated in the introduction, LDFD method can essentially be applied to any type of mesh, and Cartesian mesh is a natural choice since it is the simplest one and can save a lot of computational effort for mesh generation. Compared to other Cartesian mesh methods, LDFD method has its own unique features.

(1) In LDFD method, the boundaries are represented by a piecewise line segments linked by the interaction points between the grid lines and the boundary, so the description of the boundaries is smoother than the saw-tooth method. Therefore the boundary condition is implemented more accurately.

(2) In contrast to IBM, the boundary conditions such as no-slip boundary condition, are enforced directly, rather than satisfied approximately. And it discards the mesh nodes inside the solid body which are not adjacent to the boundary in the discretization and computation.

(3) Unlike the cut cell method, the actual boundary does not appear in the LDFD implementation, neither in computational mesh nor the discrete form of PDEs. Hence, LDFD does not involve any irregular cells.

(4) LDFD is very similar to GFM and GCM because they use extrapolation techniques to obtain the function value at a point located at the outside of the domain. But they have different physical backgrounds. The LDFD is based on the concept of extension of the solution inside of the flow domain to the exterior points outside of the flow domain. The discretization of the governing equations and the treatment of boundary conditions are decoupled. The crux of Ghost cell method is the implementation of

jump conditions or symmetry boundary condition along the interface. Therefore, in physics, the definition of flow variables at a ghost point in GCM is different from extension of the solution in LDFD. In LDFD point of view, the “ghost points” referred in GFM and GCM is actually the “real” nodes if you take these nodes as the places where both fluid domain and solid domain are overset. The governing equations do apply to these “real” nodes if you consider them located in the extended fluid domain, so that the value on these nodes are the "real" value in the flow domain. It can be stored in the current time level and used for the discretization of the governing equations in the next time level. In this sense, the LDFD can be applied to solve the moving boundary problem directly and easily, which is demonstrated in Chapter 5.

2.3 Local Domain-Free Discretization method on Cartesian mesh

The idea of Domain-Free Discretization (DFD) method has been introduced in Chapter 1. Basically, the numerical discretization in the LDFD method consists of two parts. The first is the derivative approximation using low order discretization schemes, and the second is the calculation of functional values at the mesh nodes which are outside of the flow domain. With the enforcement of the boundary condition, approximation of the functional values can be done by using low order interpolation/extrapolation schemes.

2.3.1 The procedure of LDFD method

The procedure of the LDFD method on the Cartesian mesh (the present method) is described below, which is illustrated by Fig. 2.1.

For simplicity, we assume that the mesh is uniform, and the mesh spacing in both the x and y directions is the same. Suppose that the solution of a problem is expressed by $u(x, y)$. In Fig.2.1, the mesh nodes inside the physical domain (named **interior node**) are represented by solid circles, while the mesh nodes outside the domain (named **exterior nodes**) are represented by shaded circles. There are two types of mesh nodes in the immediate neighborhood of solid boundary, and we call them **dependent points**. If the dependent point is located in the flow domain, it is termed an **interior dependent point** (represented by empty square) and if the dependent point is located outside of flow domain, it is termed an **exterior dependent point** (represented by empty circle). The small solid squares on the boundary are points intersected by the mesh lines and boundary curve, which are called **intersection points**. If an intersection point coincides with a mesh node, the mesh node is considered as an exterior dependent point whose variable values are determined by the boundary conditions (see Section 2.3.2) rather than by extrapolation scheme.

It should be noticed that in LDFD, the discretization of the PDE is always performed at the interior nodes and interior dependent points. For the exterior nodes, the discretization is skipped. (Later in Chapter 4, we will do the discretization on the whole domain, and term that method as LDFD-IBM). Therefore, the functional values at the nodes outside the domain (such as E and D in Fig. 2.1) are not defined.

In low order schemes, the numerical discretization only involves the functional values at the reference node and its immediate neighbors. For the interior nodes which are not near the boundary such as *F*, the spatial discretization is rather straightforward, e.g., using the

central-difference scheme on the uniform mesh. For the interior dependent points near the boundary such as *A1*, if the central-difference scheme on the uniform mesh is still used, the discrete form of the first and second order derivatives in the *x* direction at the position *A1* can be approximated by

$$\frac{\partial^2 u_{A1}}{\partial x^2} = \frac{1}{h^2}[u_E - 2u_{A1} + u_{B1}] + o(h^2) \tag{2.1}$$

$$\frac{\partial u_{A1}}{\partial x} = \frac{1}{2h}[u_E - u_{B1}] + o(h^2) \tag{2.2}$$

where h is the mesh spacing on the uniform mesh in the x-direction, and $O(h^2)$ denotes the order of truncation errors of the schemes.

In LDFD method, the functional value at node *E* can be calculated by extrapolation through the local approximate form of the solution. In this study, we will use the three local points (*P1*, *A1*, *B1*) to construct a polynomial-based extrapolation at the exterior dependent point *E*. Points *A1* and *B1* are inside the domain and thus their functional values are obtained by solving the governing equation; Point *P1* is the intersection point between the mesh line and the boundary, so its functional value is given from the boundary condition. The extrapolation at node *E* can be written as,

$$u_E = \frac{(x_E - x_{A1})(x_E - x_{B1})}{(x_{P1} - x_{A1})(x_{P1} - x_{B1})} u_{P1} + \frac{(x_E - x_{B1})(x_E - x_{P1})}{(x_{A1} - x_{B1})(x_{A1} - x_{P1})} u_{A1} + \frac{(x_E - x_{P1})(x_E - x_{A1})}{(x_{B1} - x_{P1})(x_{B1} - x_{A1})} u_{B1} \tag{2.3}$$

Substituting equation (2.3) into equations (2.1), (2.2), the discrete form of the x derivatives at node *A1* can be given, which only involves information at interior and boundary points.

Similarly, the discrete form of the first and second order derivatives in the *y* direction at the position *A2* is,

$$\frac{\partial^2 u_{A2}}{\partial y^2} = \frac{1}{h^2}[u_{B2} - 2u_{A2} + u_E] + o(h^2) \tag{2.4}$$

$$\frac{\partial u_{A2}}{\partial y} = \frac{1}{2h}[u_{B2} - u_E] + o(h^2) \tag{2.5}$$

where

$$u_E = \frac{(y_E - y_{A2})(y_E - y_{B2})}{(y_{P2} - y_{A2})(y_{P2} - y_{B2})} u_{P2} + \frac{(y_E - y_{B2})(y_E - y_{P2})}{(y_{A2} - y_{B2})(y_{A2} - y_{P2})} u_{A2} + \frac{(y_E - y_{P2})(y_E - y_{A2})}{(y_{B2} - y_{P2})(y_{B2} - y_{A2})} u_{B2} \tag{2.6}$$

It should be noted that two distinct values are obtained at the node *E*. One is used for the discretization of *x* derivatives at node *A1*, while the other is used for the discretization of *y* derivatives at node *A2*.

2.3.2 Treatment of Boundary Conditions

In the previous section, it is known that the boundary conditions should be implemented at boundary points *P1*, *P2*, and the functional values calculated from the boundary conditions have to be substituted into the discrete form of the PDEs through the process of extrapolation (Equations (2.3),(2.6)). The implementation of the boundary conditions at points *P1*, *P2* can influence the computation significantly. Basically, there are two types of boundary conditions: Dirichlet and Neumann conditions. Most of boundary conditions are the combination of these two basic boundary conditions.

2.3.2.1 Dirichlet boundary condition

When the boundary conditions at points *P1*, *P2* are the Dirichlet type, i.e. the variable values are known, the implementation is straightforward. That is, the functional values can be substituted directly into equations (2.3) and (2.6).

2.3.2.2 Neumann boundary condition

If the gradient in a particular direction (usually normal to the boundary) is given as the boundary condition, it may be necessary to use some close-to-boundary points to approximate the derivatives at boundary. Here, we take the point *P1* as an example. Suppose that the first order derivative is given at the boundary point *P1*. Then the following second order one-sided approximation form can be used,

$$(\frac{\partial u}{\partial x})_{P1} \approx \frac{u_{A1}(x_{B1}-x_{P1})^2 - u_{B1}(x_{A1}-x_{P1})^2 - u_{P1}[(x_{B1}-x_{P1})^2 - (x_{A1}-x_{P1})^2]}{(x_{B1}-x_{P1})(x_{A1}-x_{P1})(x_{B1}-x_{A1})} \tag{2.7}$$

When zero gradient in the x direction is prescribed and the first order approximation is adopted, we have

$$\left(\frac{\partial u}{\partial x}\right)_{P1} = 0 \Rightarrow \frac{u_{A1}-u_{P1}}{x_{A1}-x_{P1}} = 0 \Rightarrow u_{P1} = u_{A1} \tag{2.8}$$

If the Neumann boundary condition at point *P1* is given in the normal direction to the boundary $\vec{n}$, say $\frac{\partial u}{\partial \vec{n}}, \frac{\partial^2 u}{\partial \vec{n}^2}$, the points *aa*, *bb*, *cc* as shown in Fig.2.1(a), which are the intersection points of the perpendicular line through *P1* and the grid lines, are used in the approximation formulae similar to equations (2.7),(2.8) except that the perpendicular distances $|aa-P1|, |bb-P1|, |bb-aa|$ instead of the distances in the *x* direction are used.

The functional values at the points *aa*, *bb*, *cc* can be calculated by interpolation between nodes $A1$ and G (for point *aa*), nodes H and G (for point *bb*), and nodes H and F (for point *cc*).

One can also adopt the simpler strategy shown in Fig.2.1(b), that is, we can find two points *d1,d2* along the normal direction to the boundary through P1 with same spacing say Δ_d. These points should locate at a mesh cell whose 4 vertices are all interior nodes (for example, point *d1* is in the mesh cell $\overline{A1B1HG}$ whose 4 vertices *A1 ,B1 ,H ,G* are all inside of the flow domain). Then the function values at *d1,d2* can be obtained using bilinear interpolation. Thus, $\frac{\partial u}{\partial \vec{n}}, \frac{\partial^2 u}{\partial \vec{n}^2}$ can be written as

$$\frac{\partial u}{\partial \vec{n}} = \frac{4u_{d1} - u_{d2} - 3u_{P1}}{2\Delta_d} \tag{2.9}$$

$$\frac{\partial^2 u}{\partial \vec{n}^2} = \frac{u_{d2} - 2u_{d1} + u_{P1}}{\Delta_d^2} \tag{2.10}$$

2.3.2.3 No-slip boundary condition

No-slip boundary condition is commonly used at solid surfaces in incompressible flows, which could be the combination of Dirichlet and Neumann boundary conditions. In LDFD method, it is used to determine the flow variables at intersection points such as P1. In this thesis, two forms of N-S equations are used. They are the vorticity-stream function formulation and the primitive variable form, therefore no-slip boundary condition can take different expressions.

1) No-slip boundary condition in the vorticity-stream function form

In the vorticity-stream function form, no-slip boundary condition can be written as

$$\psi = u = v = 0, \omega = \frac{\partial^2 \psi}{\partial n^2} \tag{2.11}$$

where ψ is the stream function, ω is the vorticity.

The stream function boundary condition is a Dirichlet condition, which can be enforced directly as

$$\psi_{P1} = 0 \tag{2.12}$$

The vorticity boundary condition is a Neumann type condition. The implementation of vorticity at the boundary is actually the approximation of the second-order derivative of stream-function. Using the method as shown in Fig. 2.1(a), the vorticity at the wall ω_{P1} can be given by the following second order scheme as:

$$\omega_{P1} = \frac{3}{l^2}(\psi_{aa} - \psi_{P1}) - \frac{1}{2}\omega_{aa} \tag{2.13}$$

where l is the perpendicular distance between *aa* and P1. It should be noted that the nearest point such as point *aa* is the first intersection point between the line through P1 along the direction of vector $\vec{n}$ and the grid line.

2) No-slip boundary condition in primitive variable form

In the no-slip boundary condition on the stationary wall, velocity at the boundary is assigned to be zero, i.e.,

$$u_{P1} = v_{P1} = 0, \tag{2.14}$$

For the pressure, one needs to solve local normal momentum equation

$$\left(\frac{\partial p}{\partial n}\right)_{P1} = \frac{1}{\mathrm{Re}}\left[\frac{\partial^2 (u_n)}{\partial n^2}\right]_{P1} + \frac{\rho_{P1}(u_t)^2_{P1}}{R_{P1}} \tag{2.15}$$

where R_{P1} is the local radius of curvature along the solid boundary at the intersection point *P1*, $(u_n)_{P1}, (u_t)_{P1}$ are the normal and tangential components of the velocity at *P1*. Because of zero velocity at point *P1*, $(u_n)_{P1} = (u_t)_{P1} = 0$. Therefore,

$$\left(\frac{\partial p}{\partial n}\right)_{P1} = \frac{1}{\mathrm{Re}}\left[\frac{\partial^2 (u_n)}{\partial n^2}\right]_{P1} \tag{2.16}$$

With the method as shown in Fig. 2.1(b), it can be written as

$$\left(\frac{p_{d1} - p_{P1}}{\Delta_d}\right) = \frac{1}{\mathrm{Re}}\left(\frac{(u_n)_{d2} - 2(u_n)_{d1}}{\Delta_d^2}\right) \Rightarrow$$

$$p_{P1} = p_{d1} - \frac{1}{\mathrm{Re}}\left(\frac{(u_n)_{d2} - 2(u_n)_{d1}}{\Delta_d^2}\right) \tag{2.17}$$

Same process applies for getting the functional values at intersection point *P2*. After that, using the same extrapolation formulae (2.3) and (2.6), we can get two pressure values at exterior dependent point *E*, one is for the discretization of $\left(\frac{\partial p}{\partial x}\right)_{A1}$ at node *A1*, while the other is used for the discretization of *y* derivative $\left(\frac{\partial p}{\partial y}\right)_{A2}$ at node *A2*.

2.4 Status of Mesh Nodes

2.4.1 Classification of Status of mesh nodes

Before we perform any spatial discretization, it is crucial to precisely determine the status of mesh nodes in the domain, such as which mesh node is the interior node where the

governing equation should be discretized, and which mesh node is exterior dependent point where extrapolation should be applied.

There are four statuses that mesh nodes could have. Take the nodes in Fig. 2.1 as an example:

1) STATUS=1: the interior node such as C1;

2) STATUS=2: the interior dependent point such as A1;

3) STATUS=3: the exterior dependent point such as E;

4) STATUS=4: the exterior node such as D.

Note that: the interior dependent point and exterior dependent point always appear in pairs, and they actually depend on each other.

We can decide the action at one node according to its status, e.g., always solving the governing equation at the mesh node with STATUS=1 or STATUS=2; implementing an extrapolation scheme if STATUS=3; and skipping the node from the computation if STATUS=4.

Normally, the status of one node is unique, but with one exception: when the thickness of solid wall is smaller than one grid spacing, i.e., thin object such as the trailing edge of an airfoil as shown in Fig. 2.2. In Figure 2.2, point C1 is an interior dependent point to the upper wall of the thin object, but at the same time it is also the exterior dependent point D2 to the lower wall of the thin object. Similarly, the interior dependent point C2 (to low wall) is also the exterior dependent point D1 (to upper wall). Points C1 and D1 are dependent points pair (to the upper wall), and Points C2 and D2 are dependent points pair

(to the lower wall).Therefore, mesh nodes C1 and C2 have dual statuses. These dual-status points always appear in pairs.

Each flow variable at dual-status point such as C1 has two values: one is computed from the governing equations as an interior dependent point, and the other is obtained from extrapolation as an exterior dependent point.

2.4.2 Fast algorithm of identifying the status of mesh nodes

In practice, most complex solid boundaries in 2D are defined as collections of piecewise line segments. Therefore, in this work, we present a fast algorithm which can identify the status of mesh nodes around the complex solid boundary quickly. The idea is: Firstly, set all mesh nodes with Status=1; Second, locate the surface segments quickly; Then identify the mesh nodes with Status=2 and Status=3 through finding the intersection point between mesh lines and a surface segment; After that, determine the mesh nodes with Status =4 using the filling algorithm from Computer Graphics (Foley, 1996).

The detailed procedure is shown in the following:

(*1) Surface mesh cell location*

Firstly, locate line segments of the surface mesh in the Cartesian mesh. This means that for each segment, find its maximum and minimum coordinates in *x* and *y* directions, i.e., xmax, xmin, ymax, ymin. Suppose that the computational domain starts from $\Omega_{x\min}, \Omega_{y\min}$, then

$$i1 = \mathrm{int}\left(\frac{x\min - \Omega_{x\min}}{dx}\right) \qquad i2 = \mathrm{int}\left(\frac{x\max - \Omega_{x\min}}{dx}\right) + 1$$
$$j1 = \mathrm{int}\left(\frac{y\min - \Omega_{y\min}}{dy}\right) \qquad j2 = \mathrm{int}\left(\frac{y\max - \Omega_{y\min}}{dy}\right) + 1 \tag{2.18}$$

It is found that $i1$, $i2$, $j1$, $j2$ are actually the indexes of mesh nodes. By this way, we can locate the segment in the whole computational domain quickly.

As shown in Fig.2.3, the surface mesh segment of interest $\overline{Q1Q2}$ lies in a mesh box. Index ($i1$, $j1$) represents the minimum index of the mesh box which contains the present segment, Index($i2$, $j2$) represents the maximum index of the mesh box. In this mesh box, there are mesh nodes whose indexes are from ($i1$, $j1$) to ($i2$, $j2$). Each mesh node connects with its 4 neighbors via 4 mesh edges (Figure 2.4). These mesh edges may have intersections with the segment $\overline{Q1Q2}$, such as P1 and P2.

(*2) Finding the intersection point between the segment of surface mesh and the mesh edges*

To find the intersection of mesh edges and segment in surface mesh, we take full advantage of horizontal or vertical direction on which the edges lie in the Cartesian mesh. For example, the mesh edges from (i-1, j) to (i, j) and (i, j) to (i+1, j) are lying on horizontal mesh line of $y = y_j$, while edges from (i, j-1) to (i, j) and (i, j) to (i, j+1) are lying on vertical mesh line of $x = x_i$. As shown in Figure 2.4, suppose that the coordinates of (i, j) is ($x0$, $y0$), the intersection point on the horizontal direction is *P1* ($x1$, $y1$), and the intersection point on the vertical direction is *P2*($x2$,$y2$).

It is clear that for *P1*, we have

$$x1 = x_{Q1} + \frac{x_{Q2} - x_{Q1}}{y_{Q2} - y_{Q1}}(y0 - y_{Q1}) \qquad (2.19)$$
$$y1 = y0$$

If $x1$ and $y1$ satisfy the conditions

$$x_{i-1,j} \le x1 \le x_{i,j} \quad , \quad y_{i-1,j} \le y1 \le y_{i,j} \qquad (2.20)$$

and

$$x\min_{Q1Q2} \le x1 \le x\max_{Q1Q2} \quad , y\min_{Q1Q2} \le y1 \le y\max_{Q1Q2} \qquad (2.21)$$

then the intersection point between the west edge of mesh node (i, j) and segment $\overline{Q1Q2}$ is found.

Similarly, for P2, we have

$$x2 = x0 \qquad (2.22)$$
$$y2 = y_{Q1} + \frac{y_{Q2} - y_{Q1}}{x_{Q2} - x_{Q1}}(x0 - x_{Q1})$$

If $x2$ and $y2$ satisfy the conditions

$$x_{i,j} \le x2 \le x_{i,j+1} \quad , \quad y_{i,j} \le y2 \le y_{i,j+1} \qquad (2.23)$$

and

$$x\min_{Q1Q2} \le x2 \le x\max_{Q1Q2} \quad , y\min_{Q1Q2} \le y2 \le y\max_{Q1Q2} \qquad (2.24)$$

Then the intersection point at the north edge of mesh node (i, j) is found.

This process is carried out for all the mesh nodes in the mesh box containing the segment $\overline{Q1Q2}$, i.e. $\{\text{mesh node}\,(i,j), i = i1, i2; j = j1, j2\}$ until all the intersection points are found.

(3) Identification of interior dependent points and exterior dependent points (STATUS=2 and STATUS=3)

As shown in Fig .2.4, because the intersection point P1($x1$,$y1$) is within the mesh edge from (i-1, j) to (i, j), therefore for two end nodes of this mesh edge, mesh node (i-1, j) and (i, j) are dependent points. The former is the interior dependent point and the latter is the exterior dependent point since the interior dependent point and exterior dependent point always appear in pairs.

The question is which end node is the exterior dependent point? The answer is to use area of a triangle to determine if an end node of the mesh edge is outside or inside of the solid boundary.

Take Figure 2.5 as an example, suppose that the indexes of a piecewise segment $\overline{Q1Q2}$ are arranged in the clock-wise sequence (most CAD software always give this kind of surface coordinate index sequence so that the normal outward direction of the solid body is directed into flow domain), and point $A2(i, j)$ is the given mesh node, we have already found the intersection point $P1$ of the mesh edge $\overline{A1A2}$ with surface segment $\overline{Q1Q2}$, the area of a triangle $\Delta\overline{A2Q1Q2}$ is found by evaluating the determinant

$$area(\Delta\overline{A2Q1Q2}) = \frac{1}{2}\begin{vmatrix} x_{Q1} - x_{Q2} & x_{A2} - x_{Q2} \\ y_{Q1} - y_{Q2} & y_{A2} - y_{Q2} \end{vmatrix} \tag{2.25}$$

If $A2$,$Q1$, and $Q2$ are in anti-clockwise order, the determinant is positive. It is negative if they are clockwise. If the area of triangle $\Delta\overline{A2Q1Q2}$ is negative, then $A2$ is the exterior dependent point, and $A1$ is the interior dependent point, like the case in Fig.2.5.

(*4) Identification of interiors node and exterior nodes (STATUS=1 and STATUS=4)*

In this work, we propose the so called "odd/even parity method" inspired from the scan-line polygon filling algorithm in Computer Graphics (Foley, 1996), as described in the following (see Figure 2.6):

In scan-line polygon filling algorithm, suppose that there is a scan-line $j1$ in the horizontal direction. This line has 4 intersection points with the solid body (denoted by Ω), which are noted as p1, p2, p3, p4. It was found that the odd index of intersection point (such as p1, p3) is always the point where the scan-line moves into the body while the even index of the intersection point (such as p2, p4) is the point where the scan-line moves out of the body. Therefore, we can number the series of intersection points in forward sequence along the scan direction, and mark the mesh nodes between every odd/even parity as the same status.

In our case, things become easier. Because indices of nodes with STATUS=3 have already been known, we just need to number the series of nodes with STATUS=3 on the same mesh line, and mark the indexes of the mesh nodes between odd/even parity as STATUS=4. We call this method "odd/even parity method".

Once the mesh nodes with STATUS=2, 3, 4 are identified, the remaining mesh nodes in the computational domain are with STATUS=1.

In order to identify the mesh node with dual statuses (see Section 2.4.1) , we trace the record of change of status for each node. At first, we set all mesh nodes with STATUS=1. Following the above mentioned step 1 ~ step 4 , some mesh nodes may change their status to STATUS=2~4. If a mesh node changes its status twice, then it is marked as a dual-status point.

2.5 Numerical application of LDFD to simulate incompressible flows

In this section, we apply the LDFD method to simulate 2D incompressible flows. The details of two-dimensional LDFD incompressible solver are shown below.

2.5.1 Governing Equations

The primitive variable form of N-S equations is taken as the governing equation for numerical simulation, which can be written as

Continuity equation:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.26}$$

Momentum equation:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \Delta \mathbf{u} \tag{2.27}$$

Re denotes the Reynolds number, defined as

$$\mathrm{Re} = \frac{UL}{\nu} \tag{2.28}$$

where L is the reference length, U the reference velocity and ν the kinematic viscosity.

In 2D Cartesian coordinate system, Eqs. (2.26) and (2.27) can be written as:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.29}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\nabla^2 u \tag{2.30}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\nabla^2 v \tag{2.31}$$

2.5.2 Numerical discretization

2.5.2.1 Spatial discretization by LDFD method

As shown in the previous section, the spatial derivatives can be discretized by the LDFD method. The second order central difference scheme is applied in both the x and y directions to approximate the spatial derivatives. Equations (2.30)-(2.31) can be discretized at a mesh node (x_i, y_j) as

$$\begin{aligned}&\frac{du_{i,j}}{dt} + u_{i,j}\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + v_{i,j}\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y} = -\frac{p_{i+1,j} - p_{i-1,j}}{2\Delta x} \\ &+ \frac{1}{\mathrm{Re}}\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}\right)\end{aligned} \tag{2.32}$$

$$\frac{dv_{i,j}}{dt} + u_{i,j}\frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} + v_{i,j}\frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} = -\frac{p_{i,j+1} - p_{i,j-1}}{2\Delta y} + \frac{1}{\mathrm{Re}}\left(\frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{\Delta x^2} + \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{\Delta y^2}\right) \tag{2.33}$$

If one or more of the neighboring nodes of mesh node (x_i, y_j), i.e., one of nodes $(i-1, j), (i+1, j), (i, j-1), (i, j+1)$ are located in the solid domain, then the functional values at these nodes can be evaluated by using extrapolation introduced in Section 2.2. In the process of extrapolation, the no-slip boundary condition is applied.

2.5.2.2 Temporal discretization by explicit three-step formulation

The time derivative is approximated using an explicit three-step formulation based on a Taylor series expansion in time (Young et al,2001). From Taylor' series, a function f in time can be written as

$$f(t+\Delta t) = f(t) + \Delta t\frac{\partial f(t)}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 f(t)}{\partial t^2} + \frac{\Delta t^3}{6}\frac{\partial^3 f(t)}{\partial t^3} + O(\Delta t^4) \tag{2.34}$$

where Δt is the time interval. Approximating Eq.(2.34) up to third-order accuracy, the three-step formulation can be written as:

$$\begin{aligned} f(t+\frac{\Delta t}{3}) &= f(t) + \frac{\Delta t}{3}\frac{\partial f(t)}{\partial t} \\ f(t+\frac{\Delta t}{2}) &= f(t) + \frac{\Delta t}{2}\frac{\partial f(t+\Delta t/3)}{\partial t} \\ f(t+\Delta t) &= f(t) + \Delta t\frac{\partial f(t+\Delta t/2)}{\partial t} \end{aligned} \tag{2.35}$$

2.5.2.3 Solving N-S equation by fractional-step method

It is found from above governing equations (2.26)-(2.27) that an independent equation for the pressure is absent. A fractional step method due to Choi and Moin (1994) is employed in the present work. With above discretization schemes for spatial and time derivatives, the fractional step method solves Eqs. (2.26)-(2.27) in the following procedure:

First, solve the momentum equations using the three-step formulation described in Section 2.5.2.2, i.e.,

Step 1:

$$\frac{\tilde{\mathbf{u}}^{n+1/3}-\mathbf{u}^{n}}{\Delta t/3}=-H(\mathbf{u}^{n})-G(p^{n})+\frac{1}{\mathrm{Re}}L(\mathbf{u}^{n}) \tag{2.36}$$

Step 2:

$$\frac{\tilde{\mathbf{u}}^{n+1/2}-\mathbf{u}^{n}}{\Delta t/2}=-H(\tilde{\mathbf{u}}^{n+1/3})-G(p^{n})+\frac{1}{\mathrm{Re}}L(\tilde{\mathbf{u}}^{n+1/3}) \tag{2.37}$$

Step 3:

$$\frac{\tilde{\mathbf{u}}^{n+1}-\mathbf{u}^{n}}{\Delta t}=-H(\tilde{\mathbf{u}}^{n+1/2})-G(p^{n})+\frac{1}{\mathrm{Re}}L(\tilde{\mathbf{u}}^{n+1/2}) \tag{2.38}$$

where H denotes the discrete advection operator, G is the discrete gradient operator, L the discrete Laplace operator. Superscript n, n+1/3, n+1/2, and n+1 denote the time levels.

Since $\tilde{\mathbf{u}}^{n+1}$ has been calculated by equation (2.38), we can use following equation (2.39) to compute the intermediate velocity $\mathbf{u}^{*}$.

$$\frac{\mathbf{u}^* - \tilde{\mathbf{u}}^{n+1}}{\Delta t} = Gp^n \tag{2.39}$$

With $\mathbf{u}^*$, the complete velocity $\mathbf{u}$ at t^{n+1} is corrected by including the pressure field, given by

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -Gp^{n+1} \tag{2.40}$$

where p^{n+1} is the pressure at time level t^{n+1}. Apart from the momentum equation, the continuity equation (2.26) should also be satisfied at time level t^{n+1}, that is,

$$D\mathbf{u}^{n+1} = 0 \tag{2.41}$$

where D is the divergence operator. Taking divergence of equation (2.40) and using Eq.(2.41), we have

$$Lp^{n+1} = \frac{1}{\Delta t}(D\,\mathbf{u}^*) \tag{2.42}$$

Equation (2.42) is the Poisson equation for pressure p^{n+1}. It should be noted that we need sub-iteration for solving the pressure Poisson equation. Normally we need around $O(10^2)$ iterations to get the converged solution for the pressure Poisson equation.

Once the pressure p^{n+1} is calculated by equation (2.42), the velocity $\mathbf{u}^{n+1}$ can be given by

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \Delta t G(p^{n+1}) \tag{2.43}$$

2.5.3 Numerical validation: incompressible flows over a NACA0012 airfoil

To validate our incompressible flow solver with LDFD method in solving the fluid flow with complex geometry, and test our fast algorithm for mesh node status identification for practical applications, we choose the incompressible flow over a NACA0012 airfoil at different Re numbers and different angles of attack (AOA) as test problems. The Reynolds number is based on the chord length of the airfoil and the free-stream velocity.

In these test problems, the uniform flow of $U_\infty = 1$ is given at the in-flow boundary; zero-gradient condition is imposed at far field boundaries of computational domain. No-slip boundary condition is applied at airfoil surface. The airfoil surface is represented by 200 piecewise segments which consist of 201 discrete points.

1) Steady flow at Re=500 and zero angle of attack (AOA)

The airfoil with unit chord length is located at the origin. The inflow boundary is located at 3 chord lengths upstream of the airfoil. The upper and lower boundaries are located at 5 chord lengths from the airfoil. The outflow boundary is located 7 chord lengths downstream of the airfoil.

The pressure coefficient is taken as a characteristic parameter for comparison, which is defined as,

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho U_\infty^2}, \tag{2.44}$$

where p and p_∞ are the pressure on the airfoil surface and far field, respectively.

To demonstrate that the results are grid-independent, numerical simulation for several grids with different sizes were carried out. The minimum value of the pressure coefficient along the airfoil surface is used as an indicator for this study. It is obvious that for an airfoil, a fine mesh resolution is needed in the y direction. Therefore, both uniform meshes with $\Delta y = \Delta x$ and $\Delta y = \frac{1}{2}\Delta x$ are adopted in this study. From the results listed in Table 2.1, the grid resolution of $\Delta y = \frac{1}{2}\Delta x = 0.005$ is sufficient to capture the important characteristics.

The results based on the grid resolution of $\Delta y = \frac{1}{2}\Delta x = 0.005$ are shown in Figure 2.7, in terms of pressure contours and streamlines. Both of them qualitatively appear to be in good agreement with the results obtained by using N-S solver of CFL3D (Imamura et al. 2004). Figure 2.8 shows the distribution of pressure coefficient along the airfoil surface. It is compared against numerical results of CFL3D, and a quantitative agreement has been achieved.

2) Unsteady flow at Re=1000 and AOA=10°

The airfoil with unit chord length is again located at the origin. The inflow boundary is located at 5 chord lengths upstream of the airfoil. The upper and lower boundaries are located at 5 chord lengths from the airfoil. The outflow boundary is located 15 chord

lengths downstream of the airfoil. A uniform mesh with $\Delta y = \Delta x = 0.005$ is adopted in this study.

For the unsteady case, the Strouhal number is defined as

$$St = \frac{fL}{U_\infty}, \tag{2.45}$$

where f is the frequency of vortex shedding, L is the characteristic length which is the chord length in this case. In our study, the frequency f is obtained by measuring the oscillation period of the lift coefficient. To demonstrate that the results are time step independent for this unsteady case, numerical simulation for several time intervals were carried out. The Strouhal number is used as an indicator for this study. It is found that the simulations with $\Delta t = 0.0005$ and $\Delta t = 0.00025$ give the same Strouhal number, which indicates that time step independence has been achieved.

Numerical results show that the flow eventually reaches a periodic state. The instantaneous vorticity and streamlines in one complete cycle are plotted in Figure 2.9. The current results compare well with those of Wu and Shu (2009) who solved the problem using implicit velocity correction-based immersed boundary-lattice Boltzmann method.

In the present simulation, the calculated Strouhal number is 0.86, while the value given by Johnson and Tezduyar (1994) is 0.86 and the value of Mittal and Tezduyar(1994) is 0.862. Therefore, the present result compares well with the data in the literature.

2.6 Concluding remarks

In this Chapter, the Local Domain-Free Discretization method (LDFD) is presented in details. The discretization can be carried out directly on the Cartesian mesh. The flow variables at exterior dependent points are obtained by extrapolation. The approximate schemes with second order accuracy are used for derivative discretization and functional value extrapolation. A fast algorithm to determine the status of mesh nodes is also presented for arbitrary complex geometries embedded into a structured mesh which are generally defined by the piecewise line segments in two-dimensions.

The solution procedure of LDFD method for incompressible flow is described in detail. Numerical experiments of flows past a NACA0012 airfoil, are carried out to validate the present LDFD method in simulating two-dimensional incompressible flows. The good agreements with published data show the applicability of LDFD method to the flow problems with complex geometries.

Table 2.1 Grid independent study for flow past NACA0012 airfoil

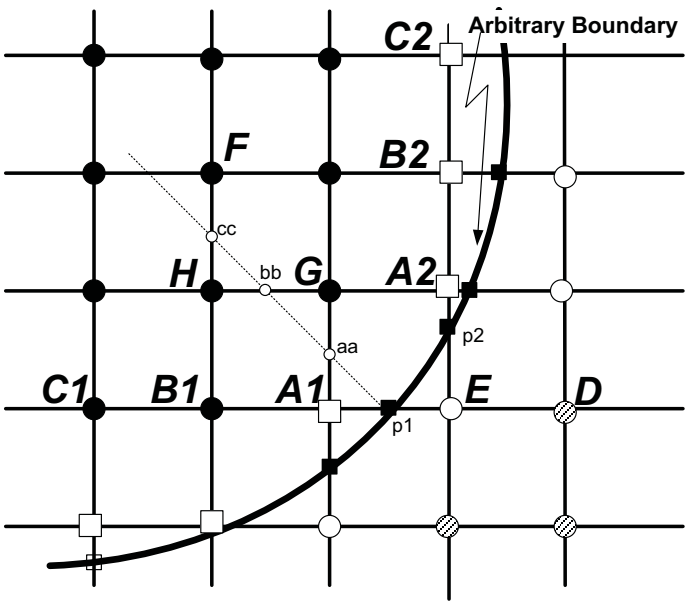
(Re=500, AOA=0)

Grid resolution	$C_{p\min}$
$\Delta x = 0.0125, \Delta y = 0.00625$	-0.3644
$\Delta x = 0.01, \Delta y = 0.005$	-0.3519
$\Delta x = 0.005, \Delta y = 0.005$	-0.3587
$\Delta x = 0.0025, \Delta y = 0.0025$	-0.3552

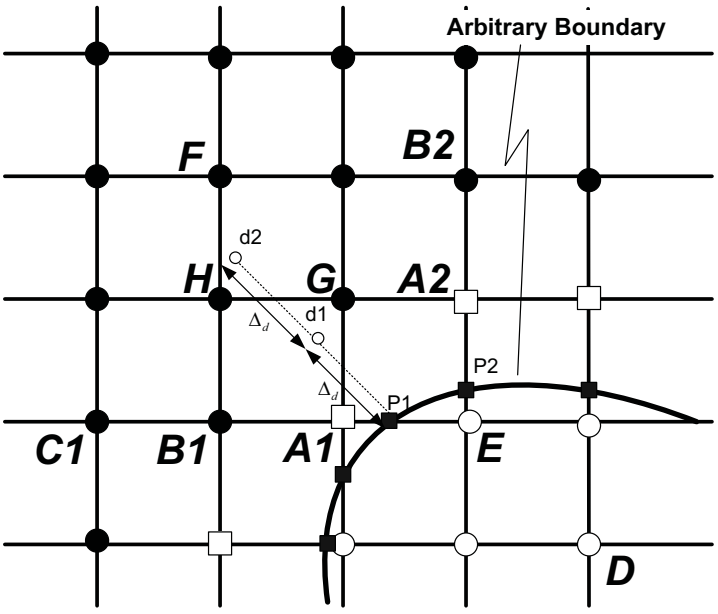
Table 2.2 Time step independent study for flow past NACA0012 airfoil

(Re=1000,AOA=10°)

Δt	St
$\Delta t = 0.001$	0.88
$\Delta t = 0.0005$	0.86
$\Delta t = 0.00025$	0.86



(a)



(b)

Figure 2.1 Configuration of the LDFD-Cartesian mesh method with mesh and extrapolation points

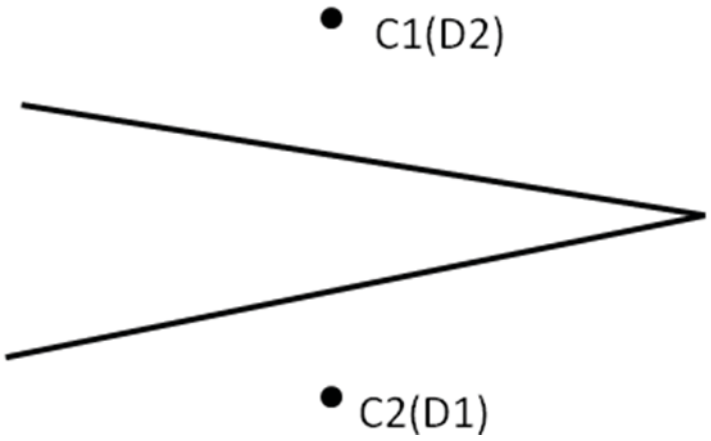


Figure 2.2 Dual statuses node associated with thin object

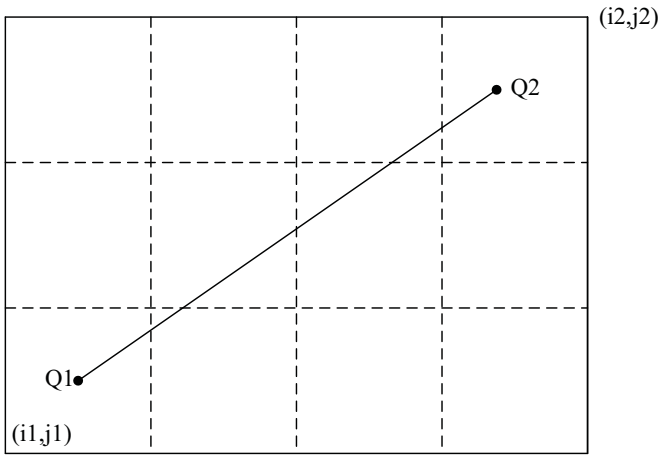


Figure 2.3 Mesh box containing a segment of surface mesh

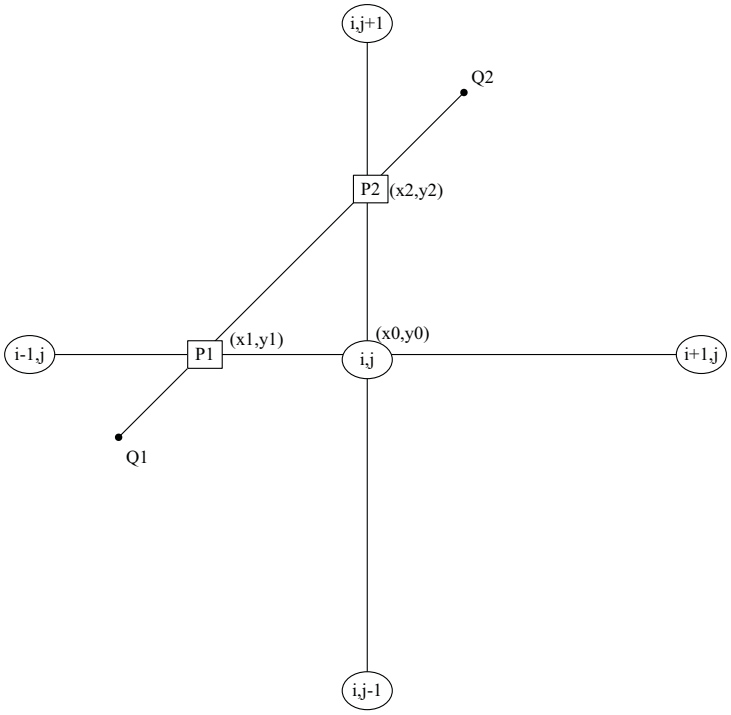


Figure 2.4 Finding the intersection point between the mesh edges and the segment

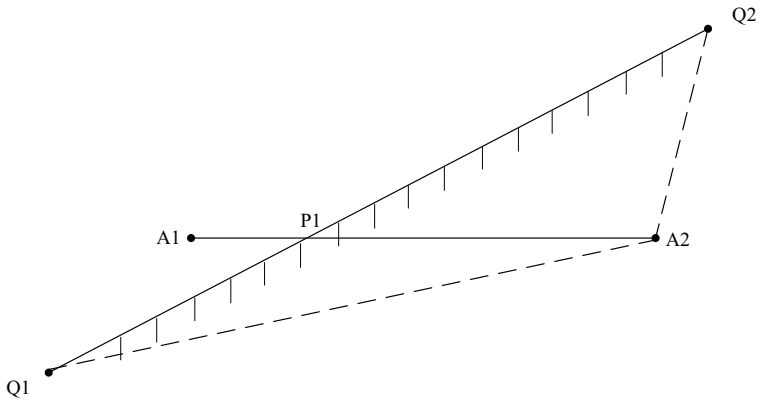


Figure 2.5 Determination of status of the dependent points

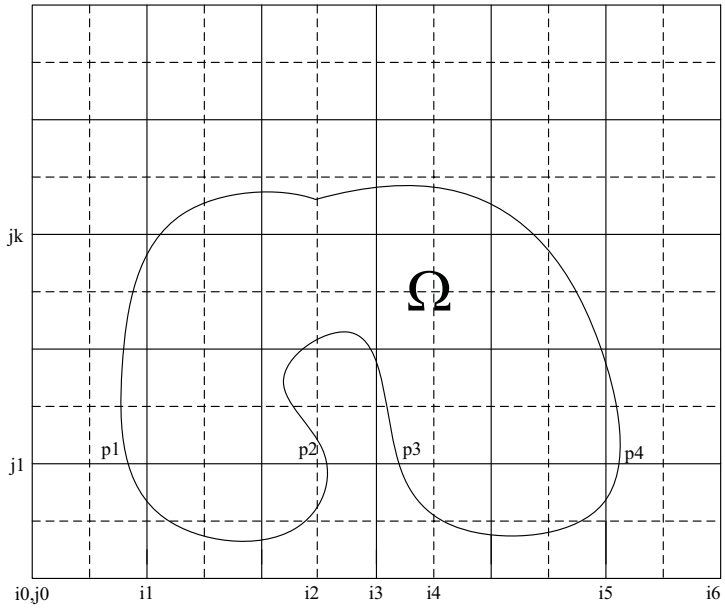


Figure 2.6 Illustration of “odd/even parity method”

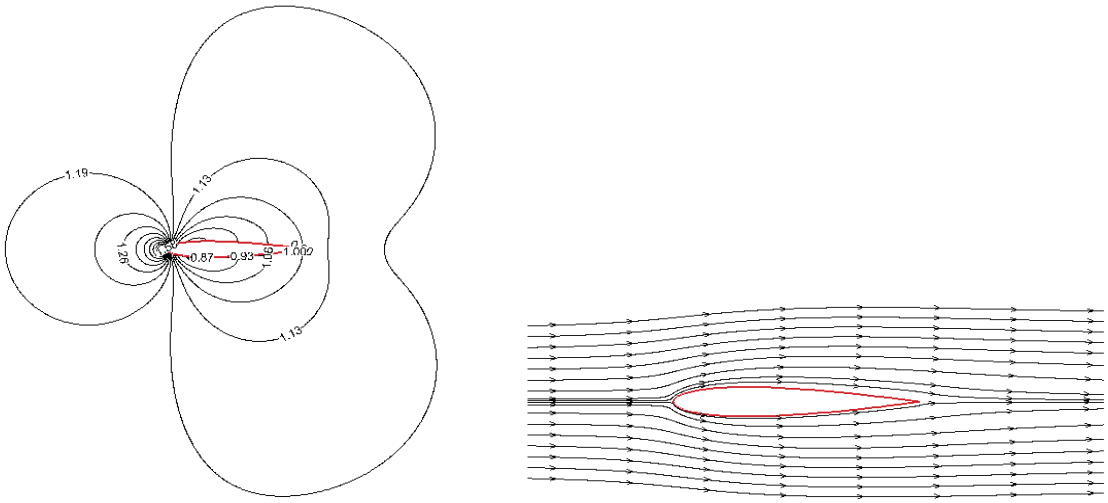


Figure 2.7 Pressure difference ($\Delta p = p - p_\infty$) contours and streamlines for flow over a NACA0012 airfoil at *Re* = 500,AOA=0°.

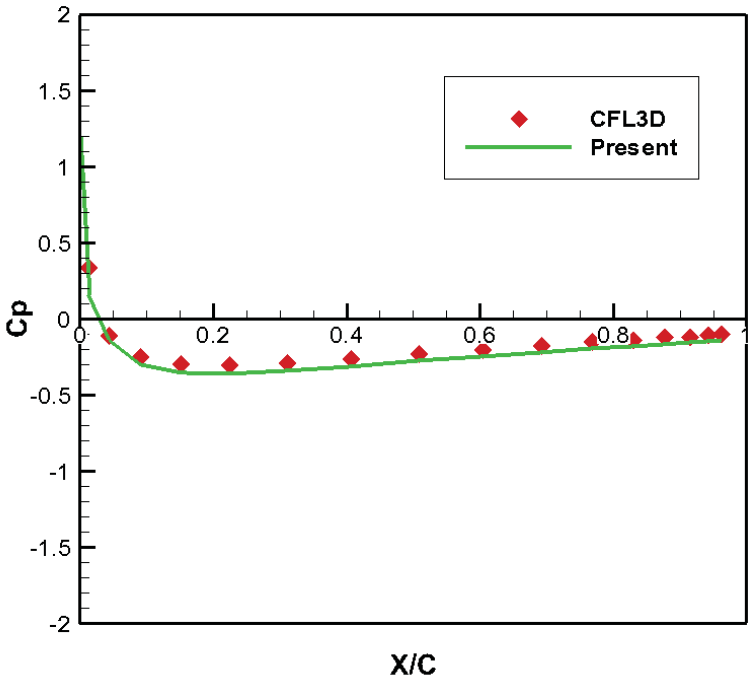
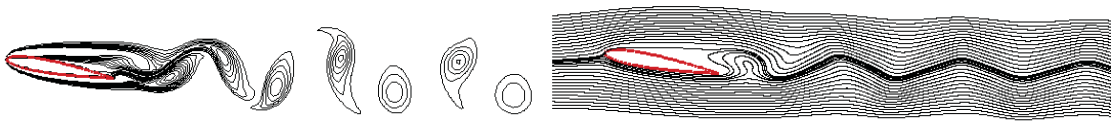
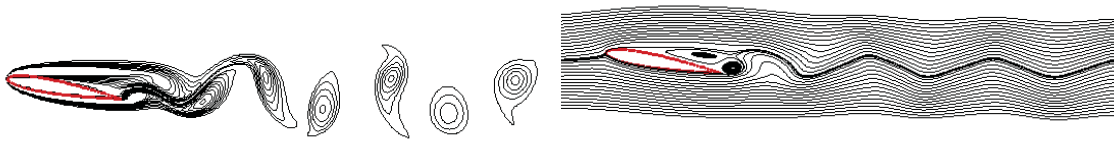


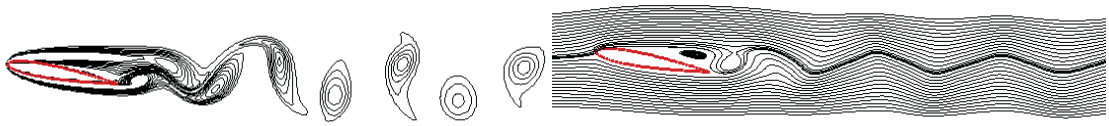
Figure 2.8 Distribution of pressure coefficient along the boundary of NACA0012 airfoil at Re=500, AOA=0°.



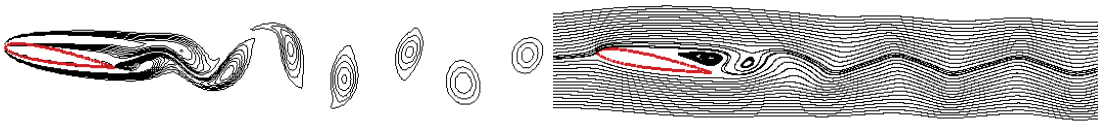
1/4 T



2/4 T



3/4 T



4/4 T

Figure 2.9 Vorticity contours and streamlines for flow over a NACA0012 airfoil at Re=1000 and AOA=10° in a cycle

CHAPTER 3

Adaptive Mesh Refinement-enhanced Local DFD Method

In general, uniform Cartesian mesh is not an ideal mesh for flow problems with different resolution regions. For example, flow past blunt bodies at high Reynolds number would need much finer mesh to resolve the thin boundary layer near solid walls than other flow regions. Solving these flow problems on a uniform Cartesian mesh usually needs very expensive computation. Therefore, from the viewpoint of computational efficiency, adaptive mesh refinement (AMR) is a desirable technique for Cartesian mesh solvers. In this Chapter, the recently proposed stencil mesh refinement (Ding and Shu, 2006) technique is introduced into LDFD method. The implementation is described in details and the efficiency improvement is checked through numerical experiments.

3.1 Review for Adaptive Mesh Refinement

A large number of AMR algorithms have been proposed in the past decades, and they can be categorized based on the mesh type, i.e., structured mesh refinement and unstructured mesh refinement. One representative of structured grid approaches is adaptive Cartesian mesh refinement proposed by Berger et al. (1984, 1998). In their approach, mesh refinement starts from the coarsest or base grid. When a cell in the base grid experiences a large gradient of variable of interest and is flagged for refinement, it can be split into sub-grids. Sub-grids can be generated recursively, i.e., the fine sub-grids will contain even finer sub-grid within their boundaries until

satisfactory resolution is achieved. Connectivity between different levels of grids is recorded in a manner of tree structures. A large amount of published articles on AMR algorithms take this hierarchy of mesh structures. (Pember et al 1995; Khokhlov 1998; Liu et al 2009, etc)

Durbin and Iaccarino (2002) presented a different AMR algorithm in which the process of mesh refinement starts from an underlying, notional structured grid or background mesh with the finest resolution. The notional grid consists of "active" and "*iblanked*" nodes. The computation is performed on those "active" nodes, while the "*iblanked*" portion of the mesh is skipped from the computation, i.e., no variable is stored, nor are equations solved at "*iblanked*" locations. The adaptive refinement is carried out by adding grid lines where needed, and blanking out all but the portion of those lines that lie in the area where higher resolution is required.

For both strategies, the "hanging nodes" at the fine/coarse cell interfaces need special treatment. This treatment involves detecting the hanging nodes, identification of the support points for interpolation to get the functional values at hanging nodes, reconstruction of the stencil for interpolation and discretization around the hanging nodes. This process brings more complexity into the coding.

Recently, Ding and Shu (2006) proposed a new AMR strategy, which is based on stencils rather than meshes. The key idea is to build up an adaptive hierarchy of

symmetric 5-points stencils in the domain, so that the central difference approximation can be constructed at each interior node. There are no "hanging nodes" at the fine/coarse stencil interfaces, and the information between the nodes at different resolution levels is naturally exchanged. The implementation is quite easy and has been successfully applied to simulate incompressible flows with simple geometries.

In the present work, the adaptive stencil refinement approach proposed by Ding and Shu (2006) is introduced to enhance the efficiency of LDFD method for simulation of flow problems with complex geometries. The details of the approach are described in the following sections.

3.2 Stencil Adaptive Mesh Refinement-enhanced LDFD

3.2.1 Two types of stencil and numerical discretization

As shown in Fig. 3.1, there are two types of stencil for a reference node *A*, where nodes marked 1,2,3,4 are the four supporting nodes. They are recorded as:

1^{st} support node of node A: SUP(A,1)=node 1

2^{nd} support node of node A: SUP(A,2)=node 2

3^{rd} support node of node A: SUP(A,3)=node 3

4^{th} support node of node A: SUP(A,4)=node 4

The four supporting nodes and node A form the stencil. It is noted that both types are the 5-points symmetric stencil, so that the finite difference method can be easily implemented to approximate the derivatives at the reference node. Type I is the

normal Cartesian coordinate stencil, Type II is the stencil rotated 45° with respect to the Cartesian coordinate axis. Numerical discretization of the first order derivative at node *A* for the two types of stencil can be written as

$$\text{Type I: } \left(\frac{\partial u}{\partial x}\right)_A = \frac{u_3 - u_1}{2h}, \quad \left(\frac{\partial u}{\partial y}\right)_A = \frac{u_4 - u_2}{2h} \tag{3.1a}$$

$$\text{Type II: } \left(\frac{\partial u}{\partial x}\right)_A = \frac{(u_2 + u_3) - (u_1 + u_4)}{2\sqrt{2}h}, \left(\frac{\partial u}{\partial y}\right)_A = \frac{(u_3 + u_4) - (u_1 + u_2)}{2\sqrt{2}h} \tag{3.1b}$$

where *h* is the mesh spacing. On the other hand, the Laplacian operator on the two types of stencil (Type I and Type II) can be written as

$$(\nabla^2 u)_A = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)_A = \frac{u_1 + u_2 + u_3 + u_4 - u_A}{h^2} \tag{3.2}$$

Equation (3.2) is very useful in the discretization of N-S equation since its diffusive operator is the Laplacian operator.

3.2. 2 Stencil refinement

As shown in Fig. 3.2, when refinement is needed, four new nodes 1', 2', 3', 4' are added by simply taking the midpoint of the edge $\overline{12}, \overline{23}, \overline{34}, \overline{41}$ for Type I, or by taking the midpoint of the edge $\overline{41}, \overline{12}, \overline{23}, \overline{34}$ for Type II.

The new supporting nodes for node *A* (nodes 1', 2', 3', 4') can be recorded as

SUP_{new}(A,1)=node 1'

SUP_{new}(A,2)=node 2'

SUP_{new}(A,3)=node 3'

SUP_{new}(A,4)=node 4'

which form another type of stencil. As shown in Fig. 3.2, it is obvious that in the

above refinement process, Type I stencil will change to Type II stencil while Type II stencil will change to Type I stencil. And for both cases, the mesh spacing is reduced by the rate of $\frac{\sqrt{2}}{2}$. As the refinement is carried out at node *A* level by level, the stencil type I and II appears alternately.

For the newly added nodes such as 1',2',3',4', it is necessary to find their supporting nodes and determine their functional values. We found that node *A* and its newly added nodes 1',2',3',4' have the same stencil type. Take node 1' in Fig. 3.2 as an example. Its stencil is shown in Fig. 3.3. The first supporting node of node 1' is located at node B, which can be either the second supporting node of node 1 or the first supporting node of node 2. The supporting nodes for node 1' can then be determined as follows:

SUP(1',1)=SUP(1,2) or SUP(2,1),whichever being located at node B

SUP(1',2)=node 2

SUP(1',3)=node A

SUP(1',4)=node 1

The functional value at node 1' can be obtained by taking the average of its 4 supporting nodes. The supporting nodes and the functional values of nodes 2',3',4' can be determined in a similar way. After the refinement, the governing equations are discretized at the original mesh nodes as well as the newly inserted nodes to get the numerical solution.

It should be noted that the discretization of the derivatives at a mesh node is always based on its support nodes. Therefore, it is necessary to record the connection between the support nodes and the mesh node at every level of refinement. This can make mesh refinement/coarsening very easy.

3.2.3 Solution-based mesh refinement or coarsening

In this study, the refinement and coarsening can be solution-based. That is, the adaptive mesh is refined or coarsened according to the characteristics of the flow. Refinement or coarsening takes place only after a solution is sufficiently converged. In our study, if the difference of a certain variable (such as temperature) between the node *A* and its support nodes is larger than a user-specified maximum value, the node A is flagged to be refined. On the contrary, if the difference is less than a predefined minimum value, the node A is flagged to be coarsened. It is obvious that the total number of the mesh nodes in the adaptive mesh depends on these predefined maximum and minimum values.

3.2.4 Stencil adaptive mesh refinement-enhanced LDFD

In LDFD, a Cartesian grid can be adopted and all derivatives in the PDEs can be approximated using central difference schemes. Therefore, stencil adaptive mesh refinement which is based on symmetric 5-points stencils can be straightforwardly used in LDFD discretization. At the start of computation the coarsest or base grid is specified by the user. The base grid denoted as G-1 normally adopts stencil Type I. It

will remain fixed for the duration of the computation. When a node is flagged for refinement, 4 new mesh nodes are added in its neighborhood. The newly inserted mesh nodes serve as the support nodes of the reference node. And they can find their own support nodes. Both the mesh nodes undergoing refinement and the newly inserted mesh nodes are of stencil Type II. The grid structure for these nodes is now denoted as G-2. This process can be repeated again and again until the desired resolution is reached, and they are denoted as G-*N*, *N*=1,2,3…NMAX. NMAX is set by user, representing the maximum level of refinement. It is found that the grid structure on the level of odd index of *N* is Type I stencil, and the one on the level of even index is Type II stencil. This fact makes it very convenient to retrieve the information of the mesh on different levels.

Some technical issues regarding implementation of stencil adaptive mesh refinement into LDFD are described as below:

Identification of status of newly inserted mesh node

Because the base grid does not change, the stored information of the mesh box containing a surface segment (as shown in Fig. 2.3) remains unchanged. Through the stored grid structure information related to G-*N*, one can trace back the information such as which mesh box the newly inserted mesh node belongs to, and identify which segments may have the potential intersection with the mesh edges of the newly added mesh node. The mesh edges, now defined as the connection between the reference

mesh node and its 4 support points, also has Type I or Type II stencil as shown in Figure 3.1.

With the refinement going on, the mesh box may contain many newly inserted mesh nodes and their mesh edges attached. Therefore it is necessary to exclude those mesh edges which have no possibility of intersecting with the surface segment instead of trying to find the intersection for every mesh edge inside of the mesh box at high cost.

One way is to adopt the fast excluding test, as shown in Figure 3.4. Our goal is to find if two line segments $\overline{P1P2}$ and $\overline{Q1Q2}$ have intersection. Suppose that the rectangle containing $\overline{P1P2}$ as diagonal is R, and the rectangle contain $\overline{Q1Q2}$ as diagonal is T. If two rectangles R and T have no intersection, neither do the line segments $\overline{P1P2}$ and $\overline{Q1Q2}$. By this way, we can quickly discard those "doomed" mesh edges and focus on the ones which have potential intersection with the segment on the surface mesh.

Extrapolation scheme for Type I, Type II stencil

As shown in Fig.2.1, if the extrapolation point *E* has Type I stencil, then we can follow the same procedure as in Section 2.2 to directly implement the boundary condition and apply extrapolation scheme to get the functional value.

On the other hand, if the extrapolation point *E* has Type II stencil, as shown in Fig. 3.5, then extrapolation can be implemented along the mesh line $\overline{A1B1C1}$ at an angle of 45° to the horizontal line. After we enforce the boundary condition at intersection point *P1*, the functional values at *E* can then be evaluated by using equations (2.3) and (2.6).

3.3 Numerical validation

The flow problem of natural convection in a concentric annulus is selected here as the test case to validate the AMR-enhanced LDFD method.

Governing Equations and Numerical Discretization

The Navier-Stokes equations in the vorticity-stream function formulation are taken as the governing equations for the problem, which are written as:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \omega \tag{3.3}$$

$$u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = \Pr(\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}) - \Pr \cdot \mathrm{Ra} \cdot \frac{\partial T}{\partial x} \tag{3.4}$$

$$u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \tag{3.5}$$

where $\omega, \psi, T, \Pr, \text{and Ra}$ are the vorticity, stream function, temperature, Prandtl number and Rayleigh number, *u, v* are the components of velocity in the *x* and *y* directions, which can be calculated by

$$u = \frac{\partial \psi}{\partial y}, v = -\frac{\partial \psi}{\partial x} \tag{3.6}$$

The geometry of the problem is given in Fig.3.6. It is a concentric annulus between a square outer cylinder and a circular inner cylinder. The reference length is taken as the side length of square *L*. The radius ratio is defined as $rr = L/2R$, where R is the radius of the inner circular cylinder. Clearly, this is a problem with irregular geometry in the Cartesian coordinate system.

The boundary conditions on two impermeable isothermal walls are given by

$$\psi = u = v = 0, \omega = \frac{\partial^2 \psi}{\partial n^2}, T = 1 \tag{3.7}$$

on the inner cylinder and

$$\psi = u = v = 0, \omega = \frac{\partial^2 \psi}{\partial n^2}, T = 0, \tag{3.8}$$

on the outer cylinder.

In the AMR-enhanced LDFD method, the second-order central difference scheme is applied in both the *x* and *y* directions to approximate the spatial derivatives. Equations (3.3)-(3.5) can be discretized by the present method at a mesh node (x_i, y_j) as

$$\frac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{\Delta x^2} + \frac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{\Delta y^2} = \omega_{i,j} \tag{3.9}$$

$$u_{i,j}\frac{\omega_{i+1,j}-\omega_{i-1,j}}{2\Delta x}+v_{i,j}\frac{\omega_{i,j+1}-\omega_{i,j-1}}{2\Delta y}= \tag{3.10}$$
$$\Pr\cdot\left(\frac{\omega_{i+1,j}-2\omega_{i,j}+\omega_{i-1,j}}{(\Delta x)^2}+\frac{\omega_{i,j+1}-2\omega_{i,j}+\omega_{i,j-1}}{(\Delta y)^2}\right)-\Pr\cdot Ra\cdot\frac{T_{i+1,j}-T_{i-1,j}}{2\Delta x}$$

$$u_{i,j}\frac{T_{i+1,j}-T_{i-1,j}}{2\Delta x}+v_{i,j}\frac{T_{i,j+1}-T_{i,j-1}}{2\Delta y}= \tag{3.11}$$
$$\frac{T_{i+1,j}-2T_{i,j}+T_{i-1,j}}{(\Delta x)^2}+\frac{T_{i,j+1}-2T_{i,j}+T_{i,j-1}}{(\Delta y)^2}$$

In equations (3.9)-(3.11), if one or more of nodes $(i-1,j)$, $(i+1,j)$, $(i,j-1)$, $(i,j+1)$ are not located in the physical domain, then the functional values at these nodes can be evaluated by using equations (2.3) and (2.6). The resultant algebraic equations are then solved by SOR method.

Numerical results and discussion

At first, the present method is validated by its application to simulate natural convection in the concentric annulus between inner circular and outer square cylinders. In this study, the Prandtl number is fixed at 0.71. The cases with the radius ratios of *rr*=1.67, 2.5, 5.0 and Rayleigh numbers of 10^4, 10^5, and 10^6 were studied. The mesh size of initial uniform mesh adopted in our computation is 71×71.

The local heat transfer rate on the inner cylinder can be computed by

$$q=h(T_i-T_o)=-k\frac{\partial T}{\partial n} \tag{3.12}$$

where *h* represents the local heat transfer coefficient, *k* is the thermal diffusivity. Because $T_i-T_o=1$, from Eq.(3.12), we can get

$$h = -k\frac{\partial T}{\partial n} \tag{3.13}$$

Then the average heat transfer coefficient $\bar{h}$ for the inner circular cylinder can be computed as

$$\bar{h} = \frac{1}{2\pi}\int_0^{2\pi} h d\theta \tag{3.14}$$

The average Nusselt numbers for the outer and inner boundaries are respectively determined by

$$\overline{Nu}_i = \frac{\bar{h}_i S_i}{k}, \quad \overline{Nu}_o = \frac{\bar{h}_o S_o}{k} \tag{3.15}$$

where S_i and S_o are defined in the same way as in the work of Moukalled and Acharya (1996). In their work, the computational domain was taken as half of the physical domain due to the symmetry, so S_i and S_o are taken as half of the circumferential lengths of the inner and outer cylinder surfaces respectively. Since at steady state, the Nusselt numbers along the inner and outer walls are the same, there is no need to pay separate attention to $\overline{Nu}_i$ and $\overline{Nu}_o$. Thus in this study, we only show the value of $\overline{Nu}_i$.

The maximum stream function value $\psi_{\max}$ and the average Nusselt number $\overline{Nu}_i$ obtained by the present method are compared with those of Moukalled and Acharya (1996) as well as those of Shu et al (2001) in Table 3.1. It should be noted that due to different ways of non-dimensionalization between the work of Moukalled and Acharya (1996) and the present study, the equivalent $\psi_{\max}$ in Table 3.1 is the one given from

Moukalled and Acharya (1996) multiplying by the Prandtl number. It is noted that the reference length used in the Rayleigh number is the side length of square *L*. From Table 3.1, it can be seen that the present results agree very well with available data in the literature. The streamlines and isotherms for $Ra=10^6$ are shown in Fig. 3.7.

To study the efficiency of the adaptive mesh refinement-enhanced LDFD method, cases with aspect ratio of *rr*=2.5 and Rayleigh numbers of 10^4, 10^5 and 10^6 are considered. In this study, the temperature is adopted as the indicator of the solution-based mesh refinement/coarsening. The maximum and minimum values of the difference of the temperature between a certain node and its support nodes $\Delta T_{max}, \Delta T_{min}$, which are used as the flag to implement refinement/coarsening, are given as $\Delta T_{max} = 0.25, \Delta T_{min} = 0.003$.

It is found that, to achieve the acceptable accuracy, the mesh must be fine enough to capture geometric features of the complex boundary as well as the gradient of the variables. Therefore, the mesh refinement is applied to the region:

1) Near the boundary (to capture the thin boundary layer);
2) The gradient of the variables is large (to capture the rapid change of the flow).

In other regions, relatively coarse mesh can be used. The combination of fine and coarse meshes can greatly save the computational effort while keeping the accuracy of solution.

Table 3.2 compares the efficiency and accuracy of LDFD method with and without adaptive mesh refinement, where Method 1 denotes the LDFD-Cartesian mesh solver with adaptive refinement and Method 2 denotes the solver without adaptive refinement. It should be noted that the relaxation factors of SOR iteration for ψ, ω, T and time interval Δt in the Euler implicit scheme are fine-tuned to their optimal values for both methods. All the simulations were running on one host (CPUs: 24×2.73Ghz; RAM: 31.42GB) in Intel PCclusters.

It can be seen from Table 3.2 that, as Rayleigh number increases, more and more mesh points are needed to keep accuracy of numerical solution. For any case, to reach the same order of accuracy, the final number of field nodes needed by Method 1 is far less than that of Method 2, and the running time for Method 1 is also less than Method 2. This indicates that the efficiency of the LDFD-Cartesian mesh method is improved greatly by implementing the adaptive refinement technique. Fig. 3.8 shows the adaptive mesh according to the temperature field. Clearly, very fine mesh points are distributed in the region where the temperature gradient is high.

3.4 Concluding remarks

Stencil Adaptive Mesh Refinement, an efficient and fully solution-adaptive stencil refinement algorithm, is coupled with the LDFD method for the simulation of flow problems with irregular geometries. Numerical experiments have been performed to examine the performance of the AMR enhanced LDFD method. It has been demonstrated that with the present method, the numerical computation is much more

efficient than its original version while good accuracy is maintained.

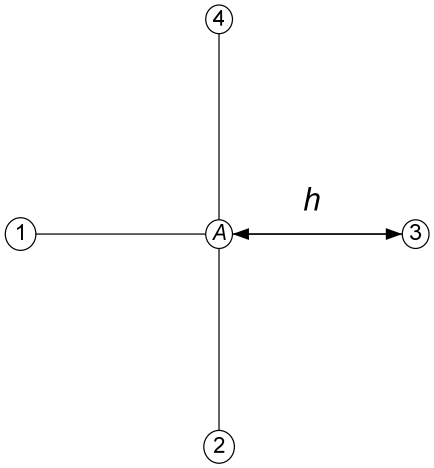
Table 3.1 Comparison of results for natural convection in concentric annuli between an inner circular cylinder and an outer square cylinder (Pr=0.71)

Ra	rr	$\psi_{\max}$			$\overline{Nu}_i$		
		Present	Shu et al (2001)	Moukalled et al (1996)	Present	Shu et al (2001)	Moukalled et al (1996)
10^4	5.0	1.76	1.71	1.73	2.065	2.082	2.071
	2.5	1.00	0.97	1.02	3.222	3.245	3.331
	1.67	0.50	0.49	0.50	5.353	5.395	5.826
10^5	5.0	10.10	9.93	10.15	3.781	3.786	3.825
	2.5	8.31	8.10	8.38	4.902	4.861	5.080
	1.67	5.08	5.10	5.10	6.184	6.214	6.212
10^6	5.0	20.93	20.98	25.35	6.362	6.106	6.107
	2.5	24.44	24.13	24.07	9.181	8.898	9.374
	1.67	19.86	20.46	21.30	12.30	12.00	11.62

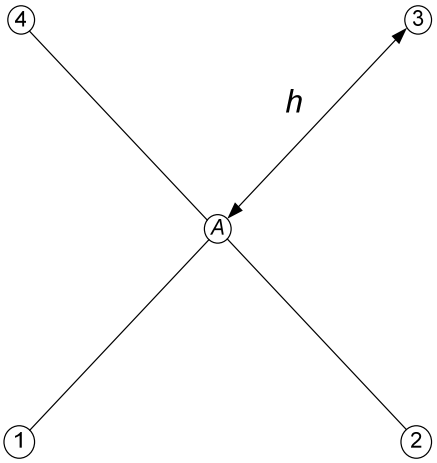
Table 3.2 Comparison of the number of nodes and running time needed to achieve the similar accuracy by the LDFD-Cartesian mesh solver with and without adaptive refinement ($Ra=10^4,10^5,10^6$, Pr=0.71,rr=2.5)

Method 1: LDFD with adaptive mesh refinement
Method 2: LDFD without adaptive mesh refinement

Ra	Method	No. of Nodes (before refinement)	No. of Nodes (after refinement)	Running Time (sec)	ψ_{max}	$\overline{Nu}_i$
10^4	Method 1	13×13=289	383	0.828	1.05	3.327
	Method 2	35×35=1225	1225	3.016	1.05	3.368
	Moukalled et al (1996)	-	-	-	1.02	3.331
10^5	Method 1	21×21=441	981	5.422	8.35	5.101
	Method 2	65×65=4225	4225	79.67	8.29	4.904
	Moukalled et al (1996)	-	-	-	8.38	5.080
10^6	Method 1	41×41=1681	2379	42.25	24.74	9.321
	Method 2	71×71=5041	5041	117.41	24.44	9.181
	Moukalled et al (1996)	-	-	-	24.07	9.374

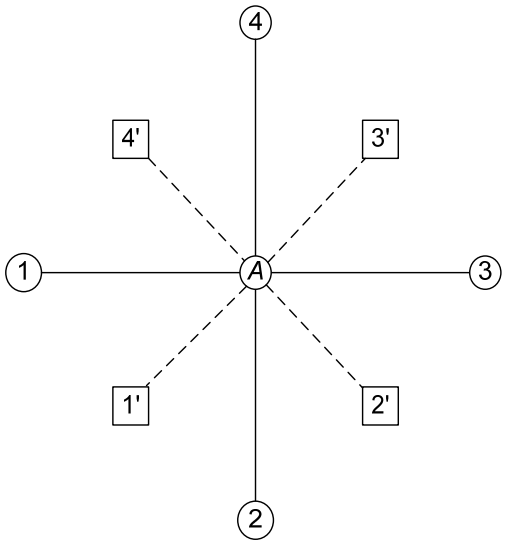


Type I

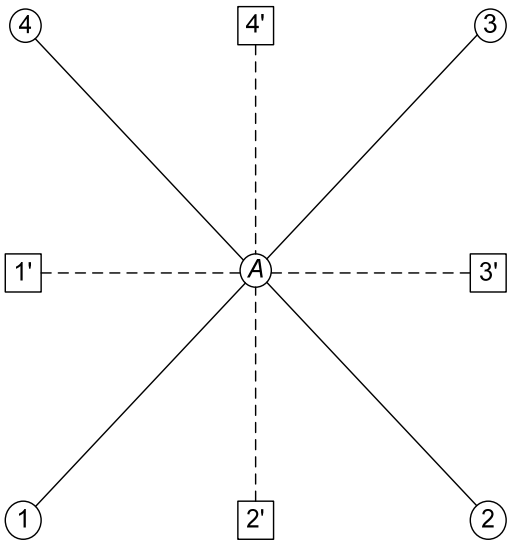


Type II

Figure 3.1 Two types of stencil on uniform Cartesian mesh



(a) Type I →Type II



(b) Type II →Type I

Figure 3.2 Transformation of two types of stencil when refinement is performed

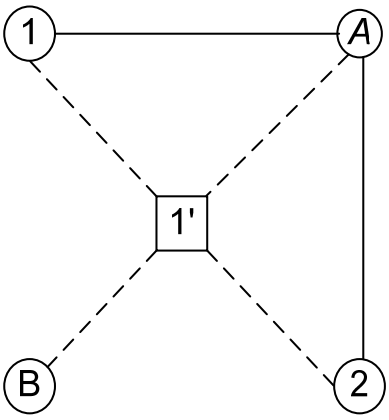


Figure 3.3 The stencil type of node 1’

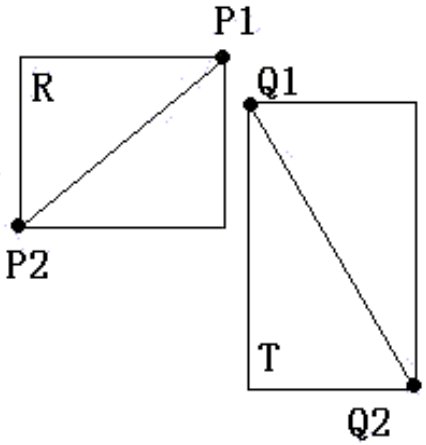


Figure 3.4 Fast excluding test

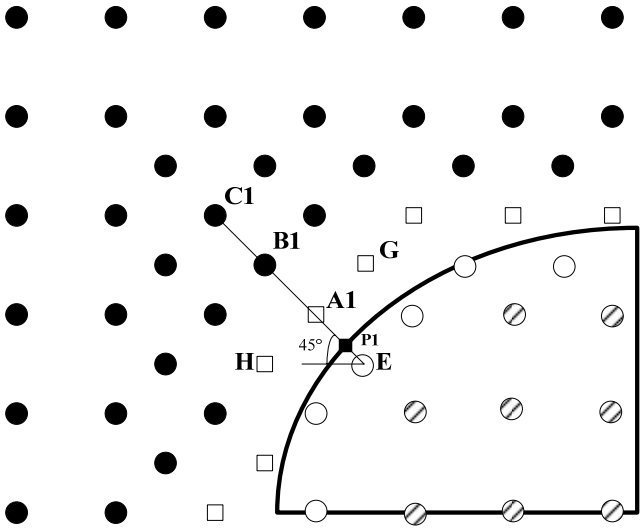


Figure 3.5 Configuration of extrapolation in stencil Type II in AMR -LDFD

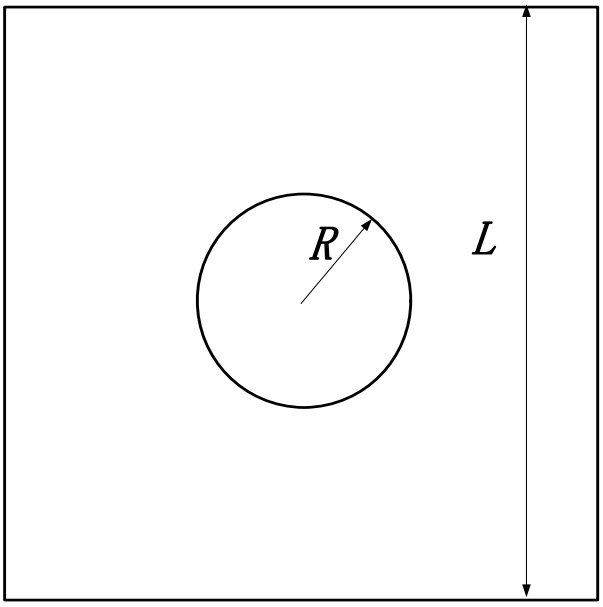
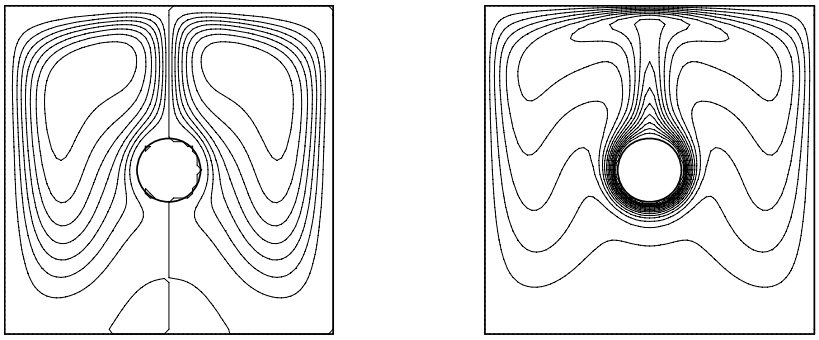
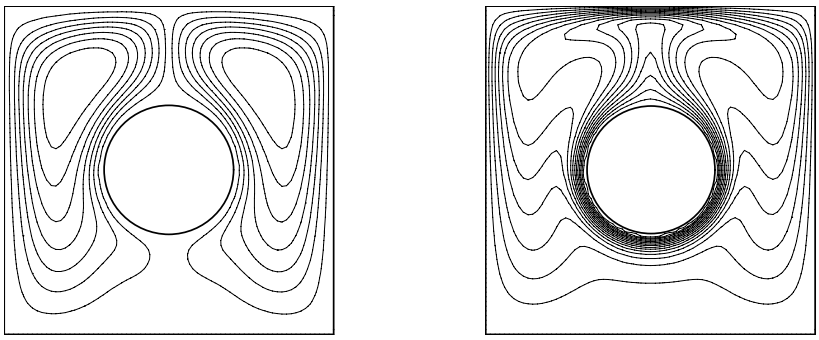


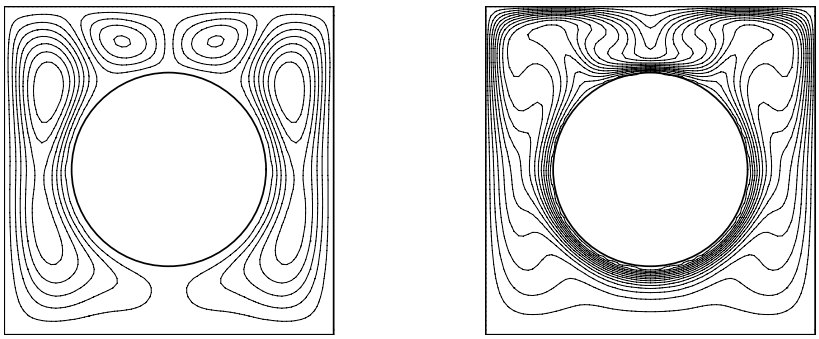
Figure 3.6. Schematic of the natural convection problem



rr=5.0

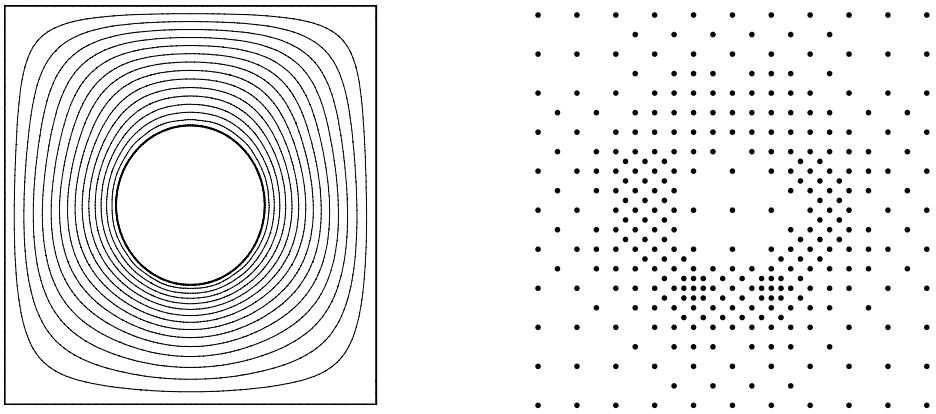


rr=2.5

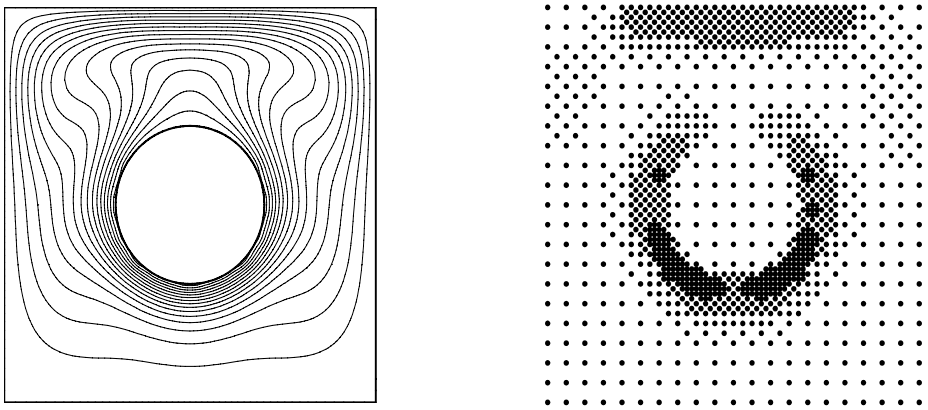


rr=1.67

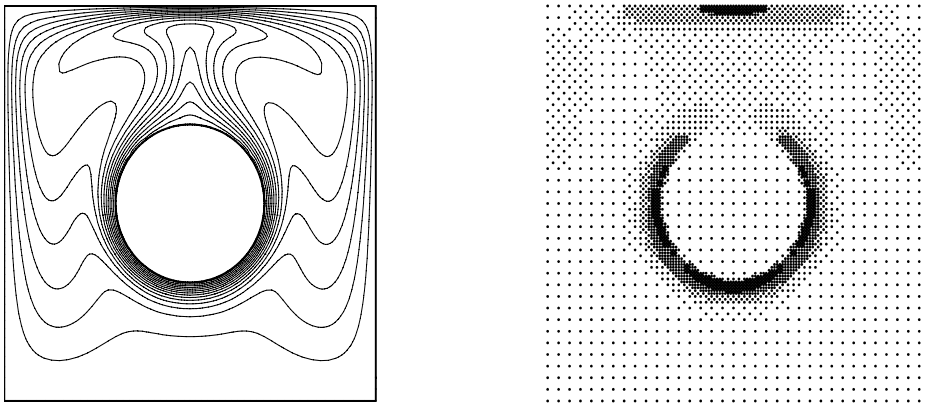
Figure 3.7 Streamlines and Isotherms in concentric annulus between inner circular cylinder and outer square cylinder (Ra=10^6, Pr=0.71,*rr*=5.0,2.5,1.67)



$Ra=10^4$



$Ra=10^5$



$Ra=10^6$

Figure 3.8 Isotherms and adaptive refined meshes based on temperature field ($Ra=10^4,10^5,10^6$, Pr=0.71,rr=2.5)

CHAPTER 4

LDFD-Immersed Boundary Method (LDFD-IBM) and Its Application to Simulate 2D Flows around Stationary Bodies

In this chapter, a hybrid LDFD and Immersed Boundary Method (LDFD-IBM) is presented. The comparisons between the new method, conventional IBM and original LDFD are made based on numerical experiments. After validation, the method is applied to study the fluid dynamics of steady and unsteady flows past single (multiple) cylinder(s) at a variety of Reynolds numbers.

4.1 The Immersed Boundary Method (IBM)

The Immersed Boundary Method (IBM) is first proposed by Peskin in the 1970s to investigate blood flow in the human heart (Peskin, 1972). The major advantages of IBM are the simplicity and the easy implementation of the method for flow problems with complex geometries. With this method, the governing equations can be solved on a regular domain. The presence of solid boundary immersed in the flow field is taken into account by putting an additional forcing term in the momentum equation. After the work of Peskin (1972), numerous research works have been made to further improve the performance of IBM. Goldstein et al (1993) proposed a model named the virtual boundary method. Lai and Peskin (2000) presented a formally second-order accurate IBM with adoption of a well-chosen Dirac delta function. Linnick and Fasel (2005) used the fourth-order compact finite-difference schemes for approximation of

spatial derivatives in the IBM application. Feng and Michaelides (2004) combined the IBM with the lattice Boltzmann method (IB-LBM) to simulate the motion of rigid particles in the flow field. Later, Niu et al. (2006) proposed the momentum exchange–based IB-LBM for simulation of several incompressible flows. Peng et al. (2006) developed the multi-block IB-LBM, and simulated the flows around a circular cylinder and an airfoil. So far, IBM is only applied to incompressible viscous flows.

The basic idea of IBM is that, the rigid solid boundary, which is represented by a set of Lagrangian points, is treated as a deformable one with high stiffness. A small distortion of the boundary will lead to a force which tends to restore the boundary into its original position. As a result, the governing equations for incompressible viscous flows past blunt bodies are almost the same as Eqs. (2.26)~(2.27) in Chapter 2, except that an additional force f_b is included in the momentum equation, as shown in Eq.(4.1),

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \Delta \mathbf{u} + f_b . \tag{4.1}$$

Note that Eq. (4.1) is solved on the Eulerian mesh, thus the conventional IBM requires a distribution of this body force from the Lagrangian points to the Eulerian mesh, including exterior and interior of the object.

It is clear that the evaluation of the force term is crucial for the performance of IBM. Different ways of deriving the forcing term lead to different variants of IBM.

4.2 Disadvantages of conventional IBM

The conventional IBM suffers from two major drawbacks. One is the flow penetration into the immersed boundary, primarily due to lack of high accuracy in the implementation of the no-slip boundary condition. The flow penetration suggests that there are mass and momentum exchanges across the immersed boundaries, which are not only unphysical but also significantly affect the accuracy of the scheme. Another drawback is the first-order accuracy in representing the immersed boundary, which is due to the use of the Dirac delta function in the information exchange between the Lagrangian and Eulerian points. In the convectional IBM, the distribution of restoring force at the boundary (Lagrangian) nodes to the Eulerian mesh points and the interpolation of flow velocity at the Eulerian mesh points to the boundary nodes are through the Dirac delta function, which only has first order accuracy.

To remove the drawback of flow penetration, Shu et al. (2007) did some analysis and found that unsatisfying of no-slip boundary condition in IBM is in fact due to pre-calculated restoring force. Using the fractional step technique, they concluded that, adding a body force in the momentum equations is equivalent to making a correction in the velocity field. To enforce the no-slip boundary condition, the velocity correction (restoring force) should be considered as unknown, which is determined by enforcing the no-slip boundary condition. In the work of Shu et al. (2007), the flow penetration is avoided. However, the numerical accuracy near the immersed boundary is still of first-order. In addition, it is necessary to find and store the intersection points between

the immersed boundary and mesh lines. This may bring some inconveniences in the numerical computation. Recently, Wu and Shu (2009) further incorporated the idea of enforcing no-slip boundary condition into the conventional IBM and presented the boundary condition-enforced IBM. The drawback of flow penetration is also removed, but the first order delta function interpolation is still used. It seems that under the current framework of IBM, the drawback of low order accuracy due to the use of delta function is difficult to overcome. There is a need to develop more accurate and efficient solvers to remove this drawback.

4.3 Combination of LDFD and IBM

The features of LDFD and IBM are similar, i.e., the governing equations can be applied to anywhere in the flow domain as long as the PDEs hold, no matter whether the point is located inside the fluid domain or outside of the fluid domain. However, the way of treating the embedded solid boundaries is very different. In LDFD, the existence of the immersed boundary affects the flow field through the correction for the functional values at the exterior dependent points; while in IBM, the effect of immersed boundary is imposed through modification of the governing equations by explicitly adding a source term (force term f_b) as shown in Eq. (4.1).

It is known that conventional IBM suffers from flow penetration because the boundary condition is approximately satisfied (Shu et al 2007). This is however not the case in the LDFD method, in which the physical boundary condition is accurately satisfied at the

intersection points such as p1, p2 in Fig. 2.1 in Chapter 2. On the other hand, it is not a trivial job to implement the pressure boundary condition at the solid wall of complex geometries in the LDFD method, which requires an extrapolation of the pressure values along the normal direction to the solid wall. However, such a complicated process is completely circumvented in the IBM. In order to combine the merits of both LDFD and IBM, we will incorporate the idea of IBM into the LDFD method, and present a new hybrid numerical method, i.e., LDFD-IBM.

The idea of LDFD-IBM is that, the governing equations are solved at all the mesh nodes (referred as the Eulerian points in conventional IBM) in the computational domain, regardless they are inside the solid bodies or inside the fluid field. The velocity boundary conditions are accurately enforced at the intersection points of the mesh lines and the boundary (referred as the Lagrangian points in the conventional IBM). The functional values at the intersection points are used to determine the solution at the interior dependent points via an interpolation scheme with second- or higher order of accuracy.

Clearly, the present approach combines the advantages of conventional IBM and LDFD method in the sense that:

(1) There is no need to implement the pressure boundary condition at the solid surface;

(2) Velocity boundary conditions are enforced accurately;

(3) An interpolation scheme rather than extrapolation scheme is adopted to obtain the approximate solution at the dependent points, which makes the computation more stable.

The disadvantage of LDFD-IBM is that it is relatively expensive to solve the equations on all the mesh nodes in the region, while LDFD only solves the equations on the interior nodes.

4.4 Procedure of LDFD-IBM

Same as in the conventional IBM, the momentum equation (4.1) is solved in the LDFD-IBM method. To represent the solid boundary correctly, we use approximate forms of the solution to evaluate the velocity at interior dependent points adjacent to the immersed boundary, and set their velocity to the wall velocity at the exterior dependent points adjacent to the immersed boundary. In this way the additional force is implicitly determined.

Figure 4.1 gives an example of implementing the LDFD-IBM for a curved solid boundary immersed in a fixed Cartesian mesh. In Fig. 4.1, the interior nodes are represented by the solid circle (such as $B1$, $C1$), and the exterior nodes are represented by shade circles (such as *D1, D2*). The empty squares which are adjacent to the immersed boundary represent the interior dependent points (such as $A1$, $A2$) and empty circles which are adjacent to the immersed boundary represent the exterior

dependent points (such as *E1, E2*). The small solid squares denote the intersection points of mesh lines and the immersed boundary, i.e., the Lagrangian nodes in conventional IBM (such as $p1$, $p2$, $q1$, $q2$).

In the LDFD-IBM, the governing equations are discretized at all mesh nodes. Take the discretization of the second-order derivatives at exterior dependent point *E2* as an example. The discrete form of the second-order derivatives in the x and y directions at *E2* can be approximated by

$$\frac{\partial^2 u_{E2}}{\partial x^2} = \frac{1}{h^2}(u_{A1} - 2u_{E2} + u_{D2}) + O(h^2), \tag{4.2}$$

$$\frac{\partial^2 u_{E2}}{\partial y^2} = \frac{1}{h^2}(u_{A2} - 2u_{E2} + u_{D1}) + O(h^2), \tag{4.3}$$

in which the velocities at $A1$, $A2$, $E2$, $D1$, $D2$ are needed for the discretization. To enforce the no-slip boundary condition, the velocity at $A1$ can be evaluated by an interpolation scheme along the x direction, which involves three points $p1$, $B1$, $C1$:

$$\begin{aligned} u_{A1} = {} & \frac{(x_{A1} - x_{B1})(x_{A1} - x_{C1})}{(x_{p1} - x_{B1})(x_{p1} - x_{C1})} u_{p1} + \frac{(x_{A1} - x_{C1})(x_{A1} - x_{p1})}{(x_{B1} - x_{C1})(x_{B1} - x_{p1})} u_{B1} \\ & + \frac{(x_{A1} - x_{p1})(x_{A1} - x_{B1})}{(x_{C1} - x_{p1})(x_{C1} - x_{B1})} u_{C1} \end{aligned} . \tag{4.4}$$

Here, $B1$ and $C1$ are the interior mesh nodes, and $p1$ is the intersection point between the horizontal mesh line and the immersed boundary. Similarly, the spatial discretization of the derivatives in the y direction for $E2$ needs the flow information at

$A2$. The velocity at the point $A2$ can be updated by the following interpolation scheme that involves three points $p2$, $B2$, $C2$,

$$u_{A2} = \frac{(y_{A2} - y_{B2})(y_{A2} - y_{C2})}{(y_{p2} - y_{B2})(y_{p2} - y_{C2})} u_{p2} + \frac{(y_{A2} - y_{C2})(y_{A2} - y_{p2})}{(y_{B2} - y_{C2})(y_{B2} - y_{p2})} u_{B2} + \frac{(y_{A2} - y_{p2})(y_{A2} - y_{B2})}{(y_{C2} - y_{p2})(y_{C2} - y_{B2})} u_{C2} \tag{4.5}$$

where $B2$, $C2$ are the interior mesh points, and $p2$ is the intersection point of a vertical mesh line with the immersed boundary.

The velocity at the intersection points $p1$ and $p2$ is assigned as the immersed boundary velocity, i.e.,

$$\begin{aligned} u_{p1} &= U_b \\ v_{p1} &= V_b \end{aligned}, \tag{4.6}$$

$$\begin{aligned} u_{p2} &= U_b \\ v_{p2} &= V_b \end{aligned} \tag{4.7}$$

where U_b, V_b are velocity components of the immersed boundary. The velocity of immersed boundary is supposed to be prescribed during the computation, and is zero for stationary solid boundaries.

Note that in equations (4.6) and (4.7), the no-slip boundary condition is directly implemented through the velocity at points *p1* and *p2*. The approximation in Eq.(4.4) and (4.5) have second-order of accuracy, and in the context of IBM they are

equivalent to solving the momentum equation with an implicit force to change the local velocity to the values determined by Eqs. (4.4) and (4.5).

In LDFD-IBM, since *E2, D1, D2* are inside the solid body, their velocities are simply assigned to the body velocities, i.e.:

$$u_{E2} = U_b \qquad (4.8)$$
$$v_{E2} = V_b$$

$$u_{D1} = U_b \qquad (4.9)$$
$$v_{D1} = V_b$$

$$u_{D2} = U_b \qquad (4.10)$$
$$v_{D2} = V_b$$

Obviously, in the above process, the solid boundary condition is implemented accurately. Again in the context of IBM, it is equivalent to solving the momentum equation with an implicit force to change the local velocity at E2, D1,D2 to the values determined by Eqs. (4.8)~(4.10).

For the computation of momentum equation at each time step, the velocity at interior dependent points needs to be updated by interpolation at least once, either along the *x* direction or along the *y* direction. It should be noted that two distinct approximate solutions may be obtained at some interior dependent points. Taking the point A1 as an example, the velocity can be obtained by interpolation along the *x* direction, i.e. using B1, C1, p1 via Eq. (4.4); on the other hand, it can also be calculated by interpolation along the *y* direction, i.e. using interior nodes R, G, and the boundary

point $q1$. To uniquely determine the value at these interior dependent points such as $A1$ and $A2$, we choose to use the average of approximate solutions from the x- and y-directions.

The discretization schemes of the same order as the LDFD method in Chapter 2 is adopted here to solve the N-S equations, i.e., the second-order finite difference scheme for the approximation of spatial derivatives and the explicit three-step formulation (as shown in Section 2.5.2.2) for the approximation of the time derivative. The fractional step method is employed to couple the momentum equations and the continuity equation in solving the N-S equations. The solution procedure is also very similar to the LDFD method, as shown in the following:

$$\frac{\tilde{\mathbf{u}}^{n+1/3}-\mathbf{u}^{n}}{\Delta t/3}=-H(\mathbf{u}^{n})-G(p^{n})+\frac{1}{\mathrm{Re}}L(\mathbf{u}^{n})+f_b \tag{4.11}$$

$$\frac{\tilde{\mathbf{u}}^{n+1/2}-\mathbf{u}^{n}}{\Delta t/2}=-H(\tilde{\mathbf{u}}^{n+1/3})-G(p^{n})+\frac{1}{\mathrm{Re}}L(\tilde{\mathbf{u}}^{n+1/3})+f_b \tag{4.12}$$

$$\frac{\tilde{\mathbf{u}}^{n+1}-\mathbf{u}^{n}}{\Delta t}=-H(\tilde{\mathbf{u}}^{n+1/2})-G(p^{n})+\frac{1}{\mathrm{Re}}L(\tilde{\mathbf{u}}^{n+1/2})+f_b \tag{4.13}$$

$$\frac{\mathbf{u}^{*}-\tilde{\mathbf{u}}^{n+1}}{\Delta t}=Gp^{n} \tag{4.14}$$

$$Lp^{n+1}=\frac{1}{\Delta t}(D\,\mathbf{u}^{*}) \tag{4.15}$$

$$\mathbf{u}^{n+1}=\mathbf{u}^{*}-\Delta t G(p^{n+1}) \tag{4.16}$$

where H denotes the discrete advection operator, G is the discrete gradient operator, L the discrete Laplacian operator, D is the divergence operator. Superscripts n, n+1/3, n+1/2, and n+1 denote the time levels.

It should be noted that, when solving Eqs. (4.11)~(4.13), the additional force f_b is added implicitly through updating the velocities at the dependent points. It is non-zero only at the interior and exterior dependent points. In fact, the velocity at an interior dependent point is computed according to Eqs. (4.4) and (4.5), and the velocity at an exterior dependent point or exterior node is set to the wall velocity (Eqs. (4.8)~(4.10)).

After $\tilde{\mathbf{u}}^{n+1}$ has been calculated by Eq. (4.13), equation (4.14) is used to compute the intermediate velocity $\mathbf{u}^*$, which is required to solve the pressure Poisson equation (4.15). Once the pressure p^{n+1} is obtained, the velocity $\mathbf{u}^{n+1}$ can be determined by Eq. (4.16). It should be noted that through Eqs. (4.15)~(4.16), the pressure is obtained by solving the governing equation directly, and there is no special treatment of pressure boundary condition along the solid bodies involved.

To sum up, the present solver has the following features:

(1) There is no need to explicitly calculate the restoration force at the Lagrangian points, and also the distribution of the restoring force from the Lagrangian points

to the Eulerian nodes. The presence of the solid boundaries is reflected by updating of the velocity at interior and exterior dependent points.

(2) The no-slip boundary conditions at the solid wall is implemented more accurately than conventional IBM, and consequently the flow penetration is avoided.

(3) Since the pressure at all the Eulerian nodes is obtained by solving the pressure Poisson equation, the treatment of Neumann boundary condition for pressure at the solid surface is no longer required.

4.5 Numerical applications

4.5.1 Decaying vortex

In this section, the numerical accuracy of LDFD and LDFD-IBM is examined. The test example is the Taylor-Green decaying vortex problem. This problem is often used to determine the accuracy of a numerical scheme since the analytical solution is available. Kim et al (2001) calculated this problem in a quadrilateral embedded domain. Uhlmann (2005) solved this case in an embedded circular domain. We solve this problem in a square domain $\Omega = [-1.5,1.5] \times [-1.5,1.5]$ in which a solid circle with radius 0.5 is embedded. Reynolds number here is taken as 100. The analytical solutions of the 2-D decaying vortices are

$$\begin{aligned} u(x,y,t) &= -\cos(\pi x)\cdot\sin(\pi y)\exp(-2\pi^2 t/\mathrm{Re}) \\ v(x,y,t) &= \sin(\pi x)\cdot\cos(\pi y)\exp(-2\pi^2 t/\mathrm{Re}) \\ p(x,y,t) &= -\frac{1}{4}(\cos(2\pi x)+\cos(2\pi y)\exp(-4\pi^2 t/\mathrm{Re}) \end{aligned} \tag{4.17}$$

In this test, the analytical solution (4.17) provides:

1) initial field at t=0;

2) The time-dependent boundary conditions at the boundary of square domain;

3) The time-dependent variable values at the circumference of the embedded circle;

4) the exact solution which is used to compute the numerical error so that the overall accuracy of solution can be measured.

Several uniform meshes (21 x 21, 41 x 41, 51 x 51,81 x 81, 101 x 101, 161 x 161) are used in the simulations. The iteration is advanced for $0 \le t \le 0.3$ using a time step of $\Delta t = 0.00001$. Figure 4.2 shows the position of the embedded circle and contours of vorticity at t=0.3 with the mesh of 161 x 161. The solution at time level t=0.3 is selected to do the comparison. The numerical error of u in terms of L_2 norm is defined as

$$L_2 error = \sqrt{\frac{\sum_N (u^{numerical} - u^{exact})^2}{N}} \tag{4.18}$$

The superscripts 'numerical' and 'exact' represent the values obtained from the present LDFD method and LDFD-IBM and from Eq. (4.17). N is the total number of interior mesh nodes between the square outer boundary and the circular inner boundary (i.e. the mesh nodes with STATUS=1 excluding the external boundary nodes) in the flow domain.

Figure 4.3 shows the convergence rate for spatial accuracy. The 2nd order accuracy is also included for comparison. It is found that for relatively finer mesh 41 x 41~161 x

161, both LDFD and LDFD-IBM have essentially 2nd order accuracy (the slope of the convergence rate line is about 2.0 for both LDFD and LDFD-IBM). This indicates that they are truly 2nd order numerical methods.

Decaying vortex problem is a tough problem for the simulation over relatively coarse mesh such as mesh size of 21 x 21. From Fig. 4.3, it is found that the performance of LDFD-IBM on the mesh of 21 x 21 is better than LDFD method in terms of numerical error. This is because LDFD uses the extrapolation scheme when obtaining the approximate solution at exterior dependent points, while LDFD-IBM uses the interpolation scheme to do the functional value approximation at interior dependent points. It is well known that the accuracy of extrapolation is worse than interpolation when the nodal spacing is large.

4.5.2 Flows past a stationary circular cylinder

Incompressible, viscous flow around a circular cylinder is a classic problem in fluid mechanics and has been studied extensively. There are numerous theoretical, experimental and numerical results available in the literature. It is well known that for this problem different Reynolds number could lead to dramatic changes in flow behaviors. It is generally agreed that in two dimensions, the vortex shedding begins at a critical Reynolds number around 49. For the case of Reynolds numbers less than the critical value ($Re_{critical}$=49), the flow is essentially a steady one. Above this critical Reynolds number, the introduced perturbation will trigger the vortex shedding process

to form a Von Karman vortex street. The flow will exhibit an unsteady feature. In this study, the LDFD-IBM is used to solve this problem with Reynolds numbers ranging from 10 to 200, expectedly with a variety of flow patterns. Here, the Reynolds number Re is defined as $\mathrm{Re} = \frac{U_\infty D}{\nu}$, where D is the cylinder diameter, and ν is the kinematic viscosity.

The problem configuration is shown in Fig. 4.4, which is an incompressible, viscous fluid flow at a constant velocity $U_\infty = 1$ past a stationary cylinder of diameter D.

The two-dimensional Navier-Stokes equations (2.26) and (2.27) are taken as the governing equations for this problem. The boundary conditions of the problem are:

At the inlet, a free stream velocity profile is specified, that is,

$$\begin{cases} u = 1 \\ v = 0 \end{cases} \tag{4.19}$$

On the cylinder surface, no-slip boundary condition is implemented by

$$\begin{cases} u = 0 \\ v = 0 \end{cases} \tag{4.20}$$

On the top and bottom boundary, the symmetric boundary condition is applied, that is,

$$\begin{cases} \dfrac{\partial u}{\partial y} = 0 \\ v = 0 \end{cases} \tag{4.21}$$

At the far field downstream boundary of the cylinder, the zero gradient boundary condition is applied by

$$\begin{cases} \dfrac{\partial u}{\partial x} = 0 \\ \dfrac{\partial v}{\partial x} = 0 \end{cases} \tag{4.22}$$

At beginning of numerical simulation, two kinds of initial flow field are assumed, one is the undisturbed velocity field at t=0, i.e.

$$\begin{cases} u\big|_{t=0} = 1.0 \\ v\big|_{t=0} = 0.0 \end{cases} \tag{4.23}$$

the other is the unsymmetrical initial flow field used to act as the initial perturbation, which is given by

$$\begin{cases} u\big|_{t=0} = 1.0 - 0.05 * x / \sqrt{x^2 + y^2} \\ v\big|_{t=0} = 0.05 * y / \sqrt{x^2 + y^2} \end{cases} \tag{4.24}$$

In the simulation, an initial Cartesian mesh of 161×101 is used. The initial uniform mesh is obviously not fine enough when Re becomes larger and larger. This is because the boundary layer will be thinner and thinner as Re increases. Therefore, the mesh refinement algorithm introduced in Chapter 3 is adopted .To simplify the simulation, a pre-refined grid with 8 refinement levels is generated near the cylinder because the thin boundary layer is expected in this region. The pre-refined grid keeps no change throughout the iteration, as shown in Fig. 4.5, which has 45363 nodes. The mesh spacing across the cylinder is 0.0125D.

4.5.2.1 Steady flow over a stationary circular cylinder

The LDFD-IBM is firstly applied to simulate the steady flow over a stationary circular cylinder with Re=20 and 40. Figure 4.6a illustrates the streamlines when the

flow reaches its final steady state. For both cases, a pair of vortices develops behind the cylinder and is perfectly aligned. This is consistent with previous observations. To demonstrate that the present approach has no flow penetration to the boundary of immersed object, the streamlines of conventional IBM as shown in the paper of Wu and Shu (2009) are displayed in Figure 4.6b for comparison, in which there are some streamlines of conventional IBM passing through the solid body. Obviously this unphysical drawback has been removed by the present LDFD-IBM.

Quantitative comparisons with other researchers' results are made in Table 4.1, regarding the parameters for the recirculation region, such as the length of the recirculation region, L_s (from the rearmost point of the cylinder to the end of the wake), separation angle θ_s and drag coefficient C_D. A good agreement has been obtained for both cases.

4.5.2.2 Unsteady flow over a stationary circular cylinder

Simulation of the vortex shedding process behind a circular cylinder is a standard test case for validating new numerical approaches to simulate unsteady flows. Figure 4.7 shows the instantaneous vorticity contours and streamlines near the wake at Re=100 ~ 200. It is found that for this range of Re number, the flow field eventually evolves into a periodic oscillatory pattern, in agreement with previous experimental and numerical findings (Braza et al 1986, Liu et al 1998, Park et al 1998, Jordan and Fromm 1972).

The time-evolution of the flow field can be quantitatively represented by the variation of the drag and lift coefficients at the surface of body, which are defined by

$$C_D = \frac{\mathbf{F}_D}{\frac{1}{2}\rho U^2 D}, \quad C_L = \frac{\mathbf{F}_L}{\frac{1}{2}\rho U^2 D} \tag{4.25}$$

where $\mathbf{F}_D$ is the drag force and $\mathbf{F}_L$ is the lift force acting on the circular cylinder. In this study, the drag force and lift force are computed by integrating the local pressure and stress distributions along the cylinder wall.

Figures 4.8 show the final periodic state of these two parameters for Re=100, 185, 200. "Time" here is non-dimensionalized by speed of incoming flow and reference length. It can be observed that lift and drag coefficients show obvious periodic oscillations for all three cases, which also indicate the periodic variation of flow field. In addition, it can also be found that the lift coefficient oscillates with larger amplitude than the drag coefficient, and the drag coefficient varies twice as frequently as the lift coefficient. These phenomena are consistent with those observed by other researchers. The reason lies in the fact that the drag coefficient is affected by vortex shedding process from both sides of the cylinder.

The drag coefficient, lift coefficient and Strouhal number ($St = fD/U$, where f is the shedding frequency) are listed in Table 4.2. The vortex shedding frequency is obtained by measuring the oscillation period of the lift coefficient. In Table 4.2, the present

results are quantitatively compared with results from other researchers. It can be observed that good agreement has been achieved.

4.5.3 Flows over a pair of circular cylinders

Flow past a pair of circular cylinders can be found in many engineering applications, such as piers and bridge pilings. This problem has been studied experimentally and numerically by many researchers (Bearman et al 1973, Williamson 1985, Zdravkovich 1977, Ding et al 2007). In this study, this problem will be investigated by using LDFD-IBM method.

Three standard classifications of the configuration of a pair of cylinders are side-by-side, tandem and staggered arrangements. There are three parameters which can significantly affect the flow structures: the Reynolds number Re, the transverse gap spacing T, and the longitudinal gap L, as shown in Figure 4.9

Experiments showed that there exists a range of flow regimes characterized by the critical gap. For example, for the side-by-side arrangements, Williamson (1985) found that three flow patterns can be observed for $T<2.2D$, $2.2D<T<5D$, $T>5D$,respectively. Zdravkovich (1977) identified three flow regimes for the tandem arrangement, i.e. , $L<1.2\text{-}1.8D$, $1.2\text{-}1.8D<L<3.4\text{-}3.8D$ and $L>4D$. These critical gaps may vary in a certain range according to the Reynolds number.

In the present study, the numerical simulations are performed for the cases with different transverse and longitudinal gaps for the side-by-side and tandem arrangements at Re=100 and Re=200. The boundary conditions are taken to be the same as those for the flow over one circular cylinder in Section 4.5.2, i.e., Eqs (4.19)~(4.22). And disturbed initial condition (4.24) is taken as the initial flow field.

4.5.3.1 Side-by-side arrangement

In this arrangement, three transverse gaps of 1.5D, 3D and 4D are used for the simulations. The computational domain is taken as 50D x 40D. Uniform Cartesian mesh of 201×161 is taken as the initial mesh. To get satisfactory resolution around the boundaries of two cylinders, we use stencil mesh refinement technique in Chapter 3 to perform 10 levels of mesh refinement, and the final mesh contains 66739 nodes, 70621 nodes, 71637 nodes for cases of T=1.5D, 3D, 4D, respectively. Figure 4.10a shows the local refined mesh around the two cylinders for T=3D.

Figure 4.11 and Figure 4.13 present instantaneous vorticity contours and streamlines for the flow around two cylinders with a transverse gap of 1.5*D* at Re=100 and Re=200. As indicated by Zdravkovich (1977), T=1.5D is within the range of intermediate critical gap (1.1D<*T*<2.2D), where the occurrence of biased flow pattern can be observed in experiments. As reported by Bearmann et al (1973), this biased flow pattern is bistable, i.e., the narrow and wide wakes, and the direction of the gap flow, switch from one cylinder to the other. From our numerical results, it is found that this

flow pattern is successfully reproduced. Figures 4.12 and 4.14 show the time evolution of drag and lift coefficients. It can be observed that drag and lift coefficients exhibit irregular variation with time, which implies that the irregular vortex shedding occurs behind the pair of cylinders.

When T>2D, there are two types of synchronized Karman vortex streets appearing in the experiments (Zdravkovich 1977): one is symmetric (anti-phase) and the other is anti-symmetric (in-phase). Our numerical solutions for the transverse gaps of 3*D* and 4*D* successfully reproduce both phenomena, and they can be seen in Figs. 4.15, 4.17, 4.19, 4.21, where the plots of instantaneous vorticity contours and streamlines are presented. In most cases, the anti-phase type of vortex streets occurs in the flow behind two cylinders. The only exception is the case of T=3*D* at Re=100, where the wake appears as clearly in-phase type of vortex. This is consistent with observations reported from other researchers (Chang et al 1990, Ding et al 2007, Wu 2010). The time evolutions of drag and lift coefficients are shown in Figs.4.16, 4.18, 4.20, 4.22. As shown in these figures, drag and lift coefficients vary synchronously with time, which also confirm the synchronized behavior of vortex shedding from the upper and lower cylinders.

The drag coefficient C_D, lift coefficient C_L and Strouhal number *St* for the case of *T* = 1.5D, 3*D* and 4D at *Re* = 100, 200 are presented in Tables 4.3~4.5, together with

some benchmark solutions available in the literature. It can be found that the present results are in reasonable agreement with previous data.

4.5.3.2 Tandem arrangement

In this arrangement, the longitudinal gaps of 2.5D, 5.5D are considered. The computational domain is again taken as 50D x 40D. Uniform Cartesian mesh of 201×161 is taken as the initial mesh. 10 levels of mesh refinement are performed by stencil mesh refinement technique, and the final mesh contains 71123 nodes, 72607 nodes for cases of T=2.5D, 5.5D, respectively. Figure 4.10b shows the local refined mesh around the two cylinders for T=5.5D.

Figures 4.23~4.24 present instantaneous vorticity contours and streamlines for L=2.5*D* at Re=100 and Re=200. For the case of Re=100, it can be observed that the flow maintains a steady state though the Reynolds number is greater than the critical value for one isolated cylinder ($\mathrm{Re} \approx 49$). The shear layer separated from the upstream cylinder reattaches on the surface of the downstream cylinder. This phenomenon is called "quasi-steady reattachment" by Zdrawkovich (1985). When Reynolds number increases, the flow field behind the upstream cylinder remains steady, i.e., quasi-steady attachment, but the flow behind the downstream cylinder becomes unsteady, as shown in Figure 4.24. The flow behind the downstream cylinder at Re=200 finally evolves into an oscillatory pattern with obvious periodicity. The corresponding drag and lift coefficients for Re=200 are shown in Fig. 4.25. It can be seen that the drag coefficients

of both upstream and downstream cylinders remain constant, but the lift coefficients vary with time.

When the longitudinal gap is increased from 2.5*D* to 5.5*D*, the flow patterns are completely different from those for the cases of *L*=2.5D, as shown in Figures 4.26 and 4.28. The separated shear layer from the upstream cylinder no longer reattaches to the front of downstream cylinder. Instead, a Karman street is formed behind the upstream cylinder. The temporal histories of drag and lift coefficients are shown in Figs. 4.27 and 4.29. The figures show that vortex shedding frequencies for the two cylinders are identical and the shedding behind the cylinders is synchronized nearly in anti-phase. This "lock-in" phenomenon confirms the findings of experiments by Rockwell (1998).

The drag coefficient C_D, lift coefficient C_L and Strouhal number *St* for the case of *L* = 2.5D, 5.5*D* at *Re* = 100, 200 are presented in Table 4.6. Some results from the literature are also included in this table. It can be found that the present results agree well with previous data.

4.5.4 Flow over three circular cylinders

To further test our LDFD-IBM for flows over multiple bodies, in this section, the flow over three identical circular cylinders in equilateral triangular array is numerically investigated by LDFD-IBM.

In this study, three configurations of three identical circular cylinders in equilateral triangular array are taken, as shown in Figure 4.30. The distance between circular cylinder centers is 3.0*D*, where *D* is the diameter of the cylinder.

The computational domain is taken as 50*D* x 40*D*. Uniform Cartesian mesh of 201×161 is taken as the initial mesh. 8 levels of mesh refinement are performed around the cylinders by stencil mesh refinement technique, and the final mesh contains 81061 nodes for Type I, 81125 nodes for Type II, 81061 nodes for Type III, respectively. Figure 4.31 shows the local refined mesh around the three cylinders for Types I, II, III.

All the simulations are conducted at Re=100 with undisturbed initial field by Eq. (4.23). The instantaneous vorticity contours and streamlines for three types of configuration are presented in Figures 4.32, 4.34, 4.36, 4.37.

For arrangement Type I, it is found from Figure 4.32 that the vortex shedding developed behind Cylinders #1 and #2 is suppressed due to the presence of Cylinder #3 and the flow patterns behind them show nearly symmetric feature with respect to the incoming stream direction. A relatively complete vortex shedding is formed behind the downstream cylinder (Cylinder #3).

For arrangement Type II, the prominent feature of the flow pattern is an individual vortex shed behind Cylinder #1. Because Cylinders #2 and #3 are in the tandem

arrangement, quasi-steady attachment occurs in the flow field behind Cylinder #2, and the vortex shedding behind Cylinder #3 is evidently disturbed by the wake of Cylinder #1.

For arrangement Type III, unlike the finding of Bao et al (2010) (see Figure 4.38) where only one type of synchronized flow is observed (perhaps they used disturbed initial field), both types of synchronized Karman vortex streets appear in present results (see Figures 4.36~4.37).

At the initial stage of the simulation, the flow pattern is completely symmetric with respect to the free stream direction, as shown in Fig. 4.36 (at t=110). An anti-phase synchronized vortex shedding can be observed behind downstream cylinders #1 and #3. It is also found that the flow behind the Cylinder #2 is in a steady state, and symmetric recirculation zone is attached behind it.

However, the wake of Cylinder #2 sandwiched by the effect of downstream cylinders could lead to unstable flow structures. As shown in the history of lift coefficients (Figure 4.39), one can see clearly that the flow around Cylinder #2 becomes more and more unstable. At around t=220, it triggers the flow deflecting from the state of anti-phase vortex shedding. An in-phase synchronized flow behind the downstream cylinders can be observed, as shown in Figure 4.37 (at t=500), which agree well with the numerical results obtained by Bao et al (2010) (as shown in Figure 4.38). The

temporal drag coefficient C_D, lift coefficient C_L are presented in Figure 4.40 when the flow eventually settles into in-phase vortex shedding.

4.5.5 Flow over four circular cylinders

Flows over four equally spaced cylinders of equal diameter are computed by LDFD-IBM in this section. The configuration is shown in Figure 4.41, in which G is denoted as the minimum distance between the centers of 4 cylinders. The computations were performed at Re=200, based on one cylinder diameter and the spacing of cylinders G=3.0D.

The computational domain is taken as 50D x 40D. Uniform Cartesian mesh of 201×161 is taken as the initial mesh. 8 levels of mesh refinement are performed around the cylinders, and the final mesh contains 97121 nodes.

Figure 4.42 shows the instantaneous vorticity contours and streamlines, in which an synchronized in-phase shedding behind the downstream cylinders can be seen clearly. According to Farrant et al (2000), the vortices formed in between the cylinders are weaker than those on the outside. This is confirmed by our results as shown in Figure 4.42 where the size of vortex in part (a) is larger than that in part (b) in vorticity contours.

Figure 4.43 shows the lift and drag coefficients for about three cycles. From Fig. 4.43, it is found that the flow patterns around all cylinders are periodic with same frequencies, which suggests that the shedding is synchronized. In-phase shedding can be observed clearly from the lift variation. The amplitudes of the lift and drag for downstream cylinders are larger than those of the upstream cylinders, which indicates that the shedding behind upstream cylinders is suppressed by the downstream cylinders. The C_D and C_L curves obtained by Farrant et al (2000) are also included for comparison. It should be noted that Farrant et al (2000) used time step rather than physical time to record the time history of C_D and C_L, therefore, the real time is the time step multiplying by time step size Δt=0.1. A good agreement of present results with those of Farrant et al (2000) can be observed.

4. 6 Concluding remarks

In this study, we propose LDFD-IBM method, which combines the merits of both LDFD and IBM. It solves the governing equations in the whole domain without the need of imposing the pressure condition at the solid wall. Therefore, as compared to original LDFD, it is easier to implement. As the no-slip boundary condition is accurately enforced in the LDFD-IBM, the flow penetration that often appears in the conventional IBM is totally avoided. The flexibility of LDFD-IBM is demonstrated by its application to simulate flows around single and multiple circular cylinders. The good agreement between present numerical results and benchmark solutions shows the good accuracy and robustness of LDFD-IBM.

Table 4.1 Comparison of geometrical and dynamical parameters for flow past one cylinder at Re=20 and 40

		L_s/a	θ_s	C_D
Re=20	Dennis and Chang (1970)	1.88	43.7°	2.05
	He and Doolen (1997)	1.84	43.0°	2.15
	Calhoun (2002)	1.82	45.5°	2.19
	Tuann and Olson (1978)	1.80	44.1°	2.25
	Ding et al (2007)	1.88	43.8°	2.14
	Present	1.80	43.8°	2.22
Re=40	Dennis and Chang (1970)	4.69	53.8°	1.522
	Calhoun (2002)	4.36	54.2°	1.62
	Tuann and Olson (1978)	4.20	54.8°	1.675
	Ding et al (2007)	4.64	52.8°	1.58
	Present	4.45	53.7°	1.567

Table 4.2 Comparison of drag coefficients, lift coefficients and Strouhal number for flow past one cylinder at Re=100 ~200

		C_D	C_L	St
Re=100	Braza *et al* (1986)	1.364 ± 0.015	± 0.25	0.160
	Liu et al (1998)	1.350±0.012	±0.339	0.164
	Park et al (1998)	1.33	±0.33	0.165
	Present results	1.330±0.010	±0.341	0.166
Re=185	Lu et al (1996) Experimental	1.28	-	0.19
	Lu et al(1996) Numerical	1.30 ± 0.05	± 0.62	0.192
	Guilmineau et al(2002)	1.287 ± 0.048	± 0.63	0.195
	Present	1.257±0.040	± 0.65	0.195
Re=200	Braza *et al* (1986)	1.40 ± 0.05	± 0.75	0.200
	Liu et al (1998)	1.31±0.049	±0.69	0.192
	Ding et al (2004)	1.327±0.045	±0.60	0.196
	Present results	1.30±0.041	±0.69	0.200

Table 4.3 Comparison of drag coefficients, lift coefficients and Strouhal number for flow past two side-by-side cylinders T=3D for Re=100

Results		C_D	C_L	St
Ding et al (2007)	Upper cylinder	1.560±0.038	-0.131±0.253	0.182
	Lower cylinder	1.560±0.038	0.131±0.253	0.182
Chang et al (1990)	Upper cylinder	1.533±0.04	-0.108±0.31	0.18
	Lower cylinder	1.533±0.04	0.108±0.31	0.18
Lee et al (2009)	Upper cylinder	1.48±0.04	-0.135±0.465	-
	Lower cylinder	1.48±0.04	0.135±0.465	-
present results	Upper cylinder	1.457±0.04	-0.120±0.31	0.181
	Lower cylinder	1.457±0.04	0.120±0.31	0.181

Table 4.4 Comparison of drag coefficients, lift coefficients and Strouhal number for flow past two side-by-side cylinders T=3D for Re=200

		C_D	C_L	St
Ding et al (2007)	Upper cylinder	1.548±0.083	-0.104±0.866	0.215
	Lower cylinder	1.548±0.083	0.104±0.866	0.215
present results	Upper cylinder	1.536±0.110	-0.11±0.911	0.221
	Lower cylinder	1.536±0.110	0.11±0.911	0.221

Table 4.5 Comparison of drag coefficients, lift coefficients and Strouhal number for flow past two side-by-side cylinders T=4D for Re=100

		C_D	C_L	St
Ding et al (2007)	Upper cylinder	1.514±0.013	0.077±0.348	0.184
	Lower cylinder	1.514±0.013	-0.077±0.348	0.184
Farrant et al(2001)	Upper cylinder	1.44±0.03	0.08±0.32	0.172
	Lower cylinder	1.44±0.03	-0.08±0.32	0.171
Lee et al(2009)	Upper cylinder	1.44±0.03	0.079±0.415	-
	Lower cylinder	1.44±0.03	-0.079±0.415	-
Bao et al (2010)	Upper cylinder	1.429	0.071±0.288	-
	Lower cylinder	1.429	-0.071±0.288	-
present	Upper cylinder	1.435±0.026	0.07±0.422	0.183
	Lower cylinder	1.435±0.026	-0.07±0.422	0.183

Table 4.6 Comparison of drag coefficients, lift coefficients and Strouhal number for flows over a pair of tandem cylinders at Re=100 and Re=200.

	Results	C_D		C_L		St
		Upstream cylinder	Down-stream	Upstream cylinder	Down-stream	
L=2.5D Re=100	Present	1.135	-0.0868	0.0	0.0	-
	Ding et al (2007)	1.163	-0.0895	0.0	0.0	-
	Mittal et al (1997)	1.271	-0.075	0.0	0.0	-
	Wu(2010)	1.146	-0.077	0.0	0.0	-
L=2.5D Re=200	Present	0.996	-0.181	0.0 ± 0.024	0.0 ± 0.223	0.194
	Wu(2010)	1.02	-0.18	0.0 ± 0.023	0.0 ± 0.24	0.19
	Ding et al (2005)	1.022	-0.22	0.0 ± 0.002	0.0 ± 0.023	0.19
L=5.5D Re=100	Present	1.223 ± 0.015	0.759 ± 0.121	0.0 ± 0.367	0.0 ± 1.307	0.156
	Ding et al (2007)	1.329 ± 0.013	0.858 ± 0.125	0.0 ± 0.330	0.0 ± 1.554	0.16
	Mittal et al (1997)	1.433 ± 0.015	0.952 ± 0.164	0.0 ± 0.403	0.0 ± 1.741	0.168
	Wu(2010)	1.301 ± 0.012	0.761 ± 0.139	0.0 ± 0.374	0.0 ± 1.527	0.154

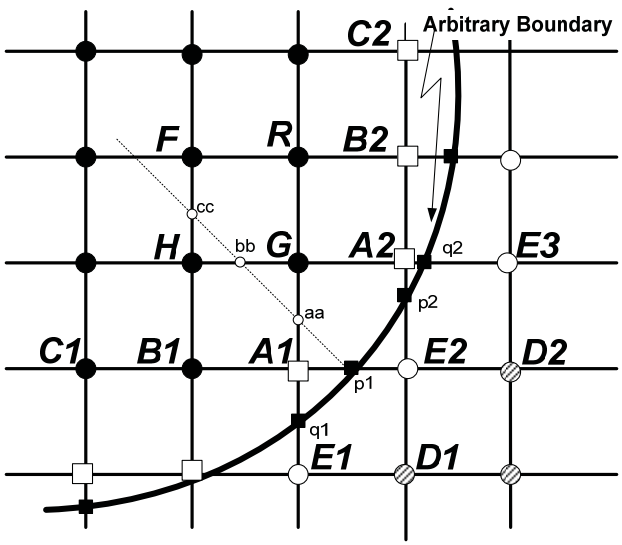


Figure 4.1 Configuration of LDFD-IBM

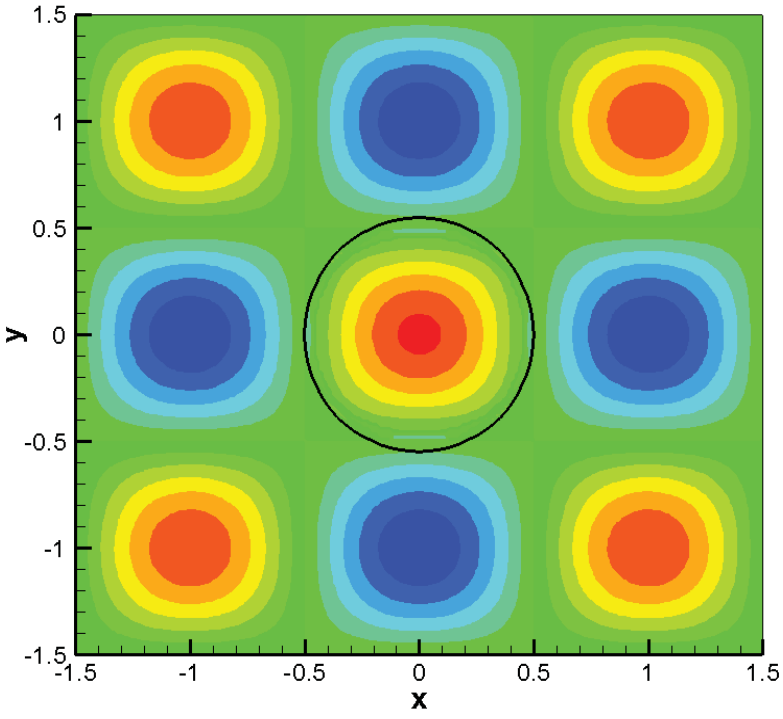


Figure 4.2 Position of the embedded circle and the contours of vorticity at t=0.3 for decaying vortex problem

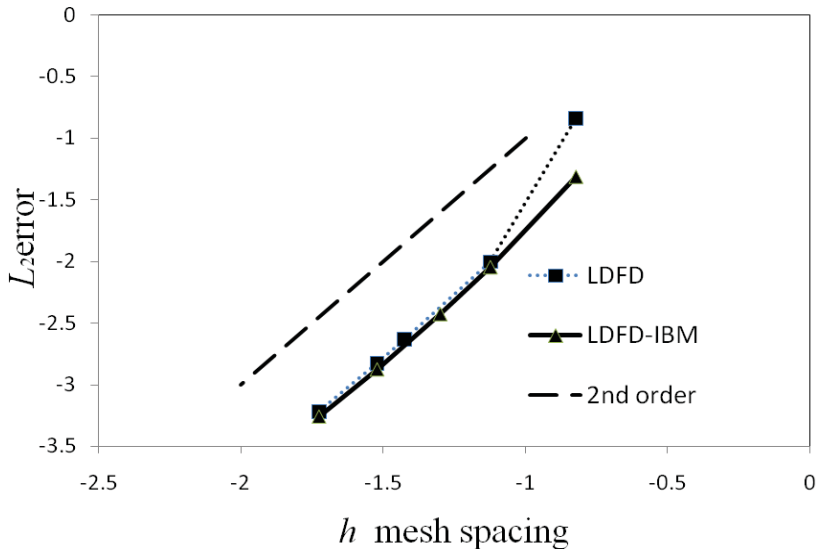


Figure 4.3 Convergence rate for decaying vortex problem

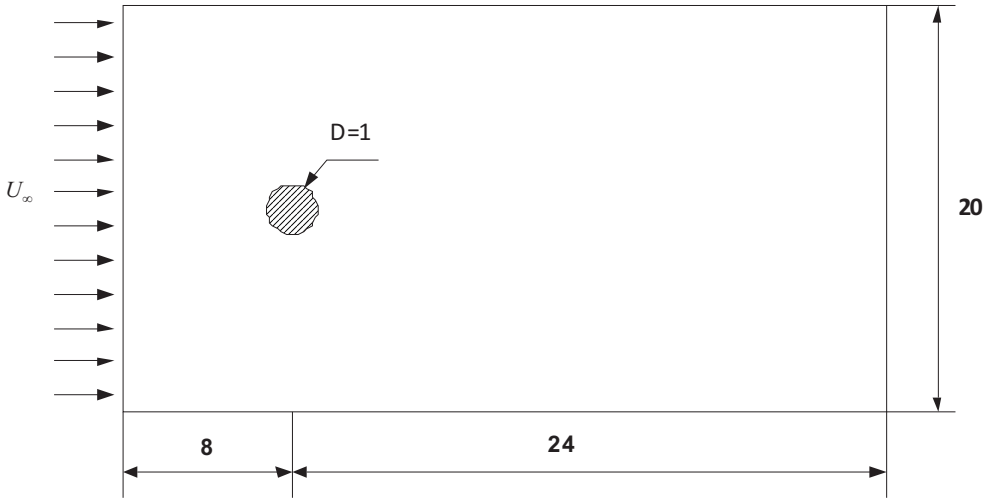


Figure 4.4 Computational domain for simulation of flow around a circular cylinder

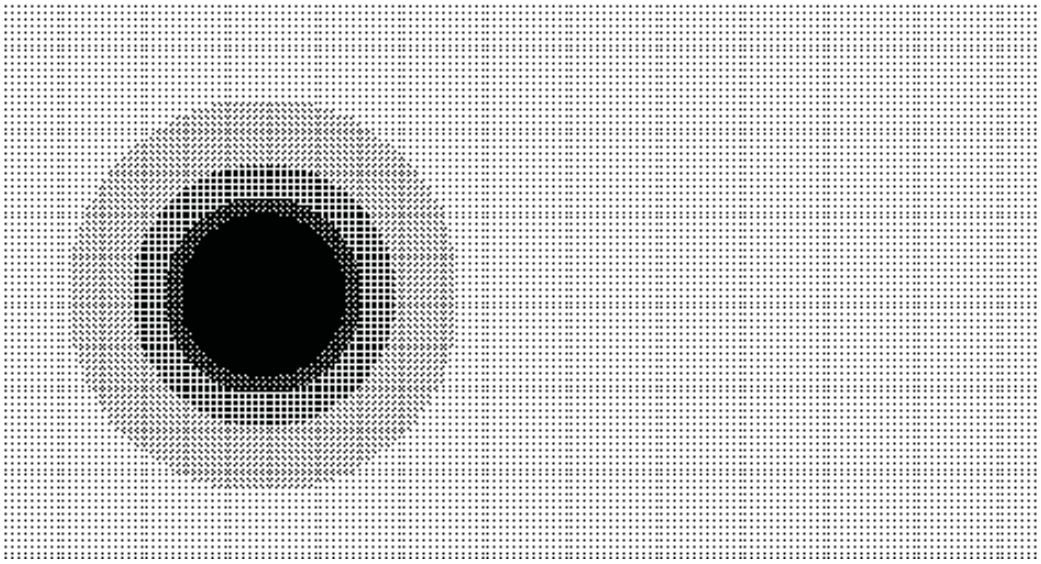
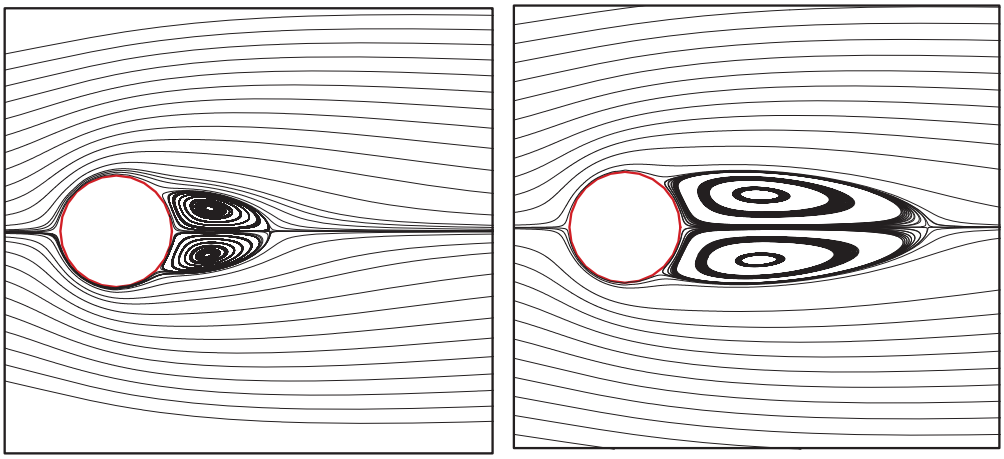
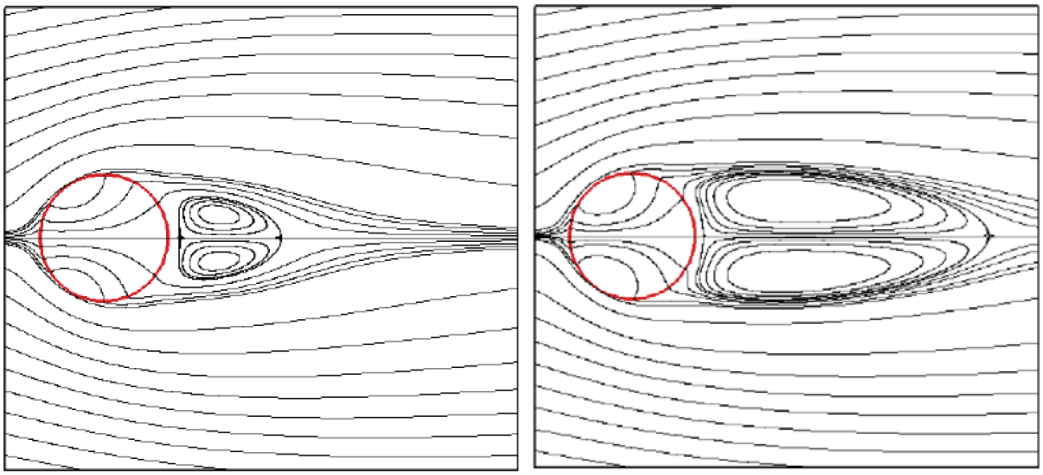


Figure 4.5 Local refined mesh for simulation of flow past a circular cylinder (45363 nodes)



Re=20 Re=40

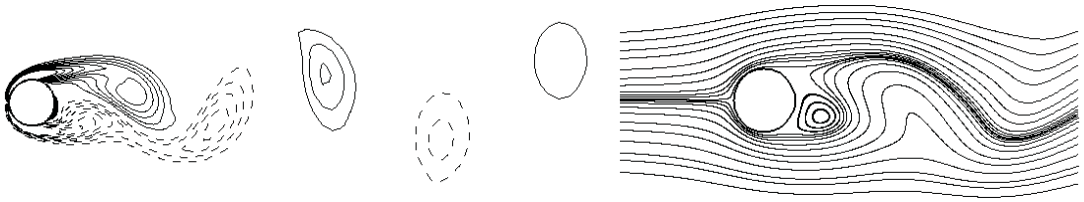
(a) Results of present method



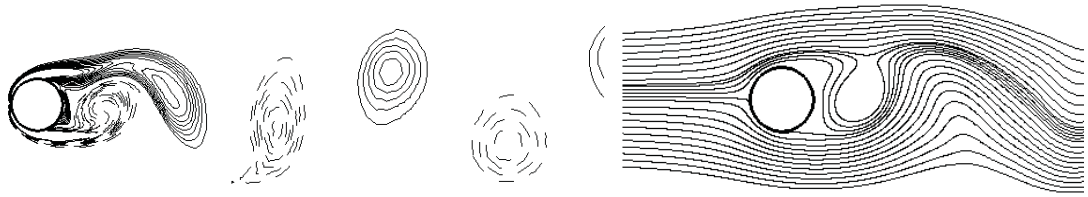
Re=20 Re=40

(b) Results from conventional immersed boundary method (Wu and Shu,2009)

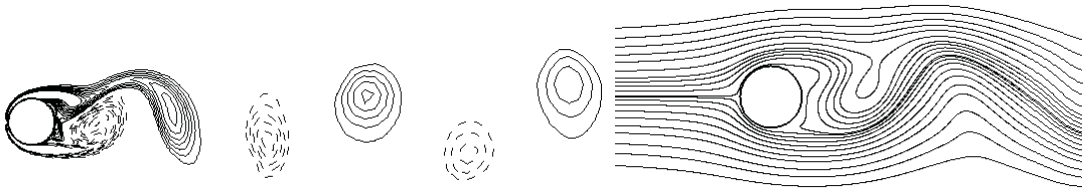
Figure 4.6 Streamlines for steady flow with Re=20 and 40



Re=100

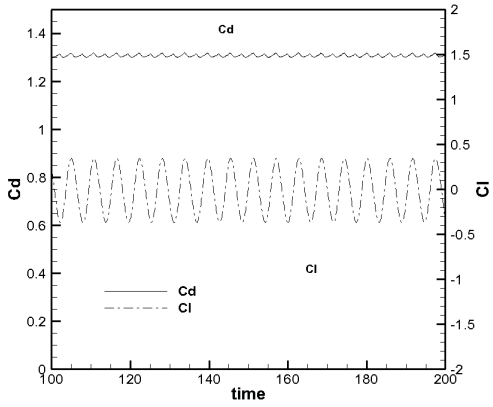


Re=185

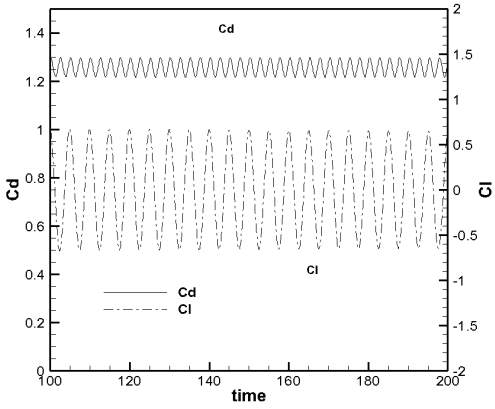


Re=200

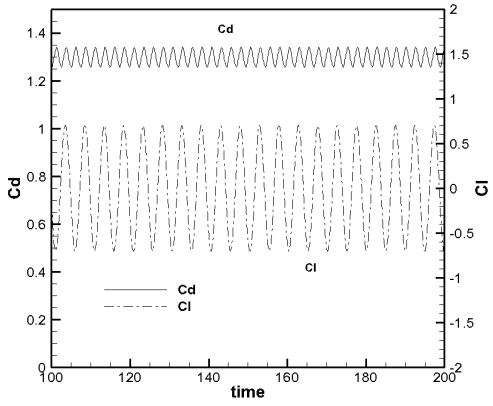
Figure 4.7 Instantaneous vorticity and streamlines for Re=100, 185,200



Re=100



Re=185



Re=200

Figure 4.8 The time-evolution of Lift and Drag coefficients for Re=100,185,200

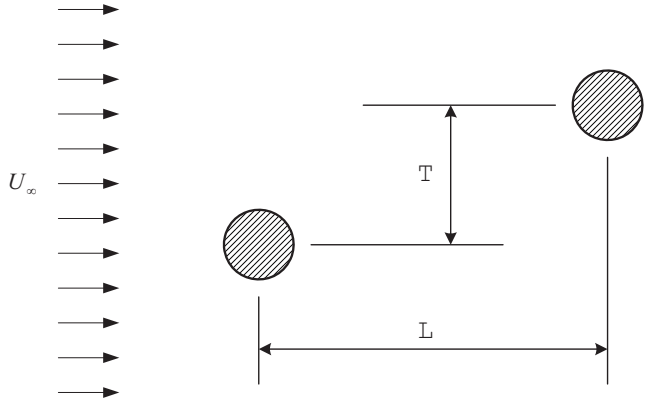
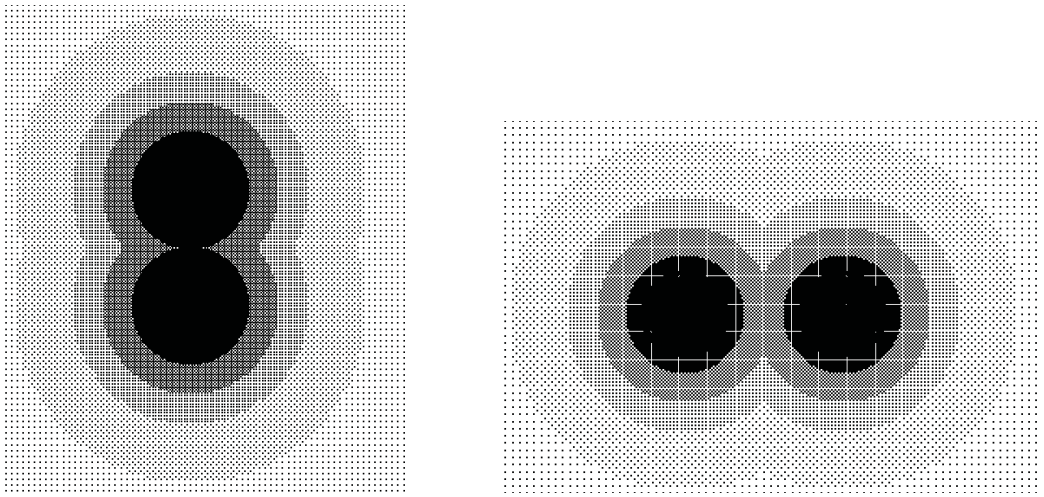


Figure 4.9 Configuration of flow past a pair of cylinders



(a) Side by side arrangement (T=3D) (b) Tandem arrangement(L=5.5D)

Figure 4.10 Local refined mesh for simulation of flow past a pair of circular cylinders

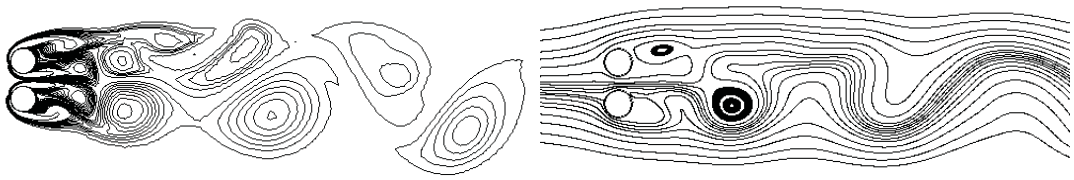


Figure 4.11Vorticity contours and streamlines for flow over a pair of side-by-side cylinders (T=1.5D) at Re=100

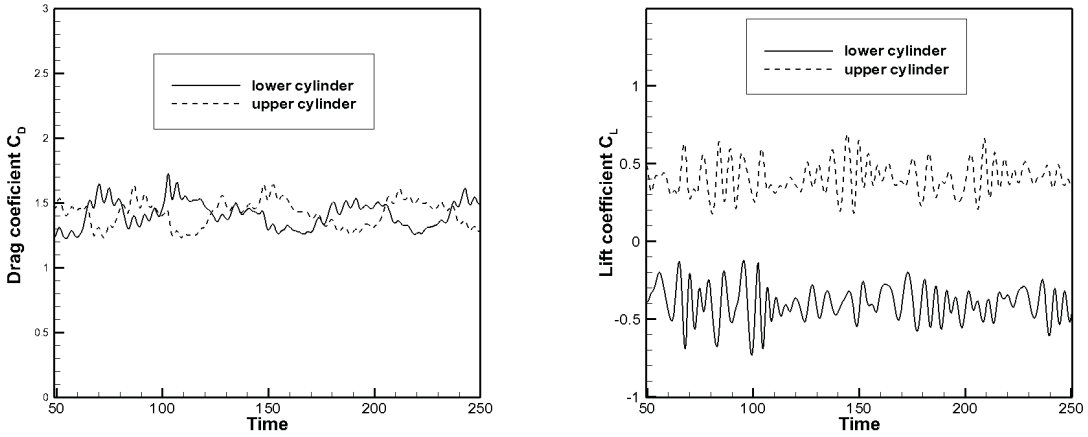


Figure 4.12 Time histories of drag (C_D) and lift (C_L) coefficients of flow past a pair of side-by-side cylinders (T=1.5D, Re=100)

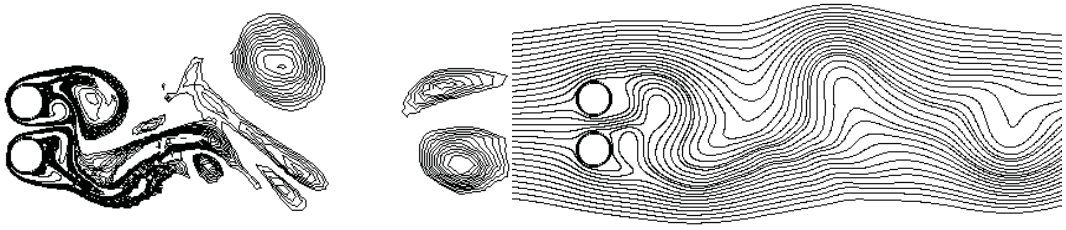


Figure 4.13 Vorticity contours and streamlines for side-by-side cylinders (T=1.5D) at Re=200

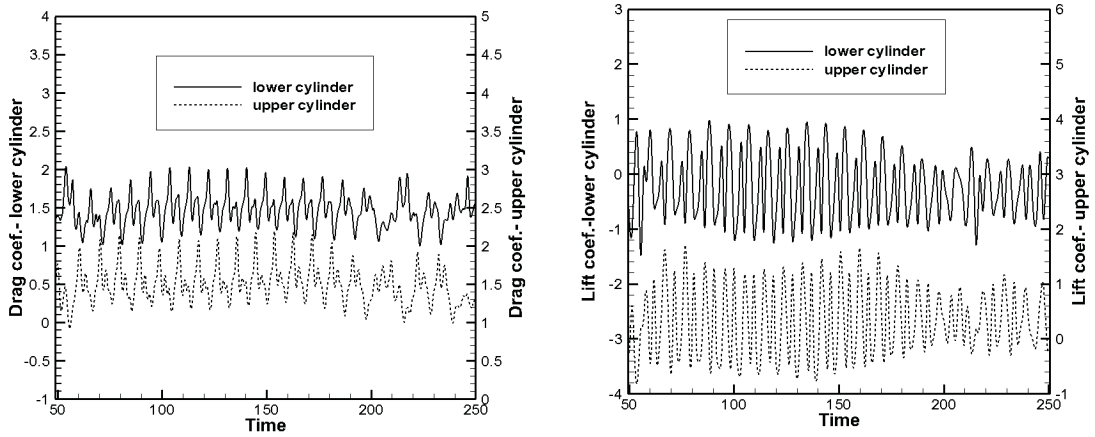


Figure 4.14 Time histories of C_D and C_L for side-by-side cylinders (T=1.5D, Re=200)

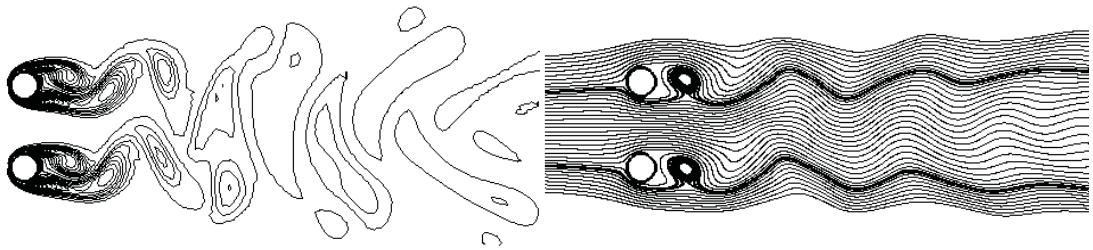


Figure 4.15 Vorticity contours and streamlines for flow over a pair of side-by-side cylinders (T=3D) at Re=100

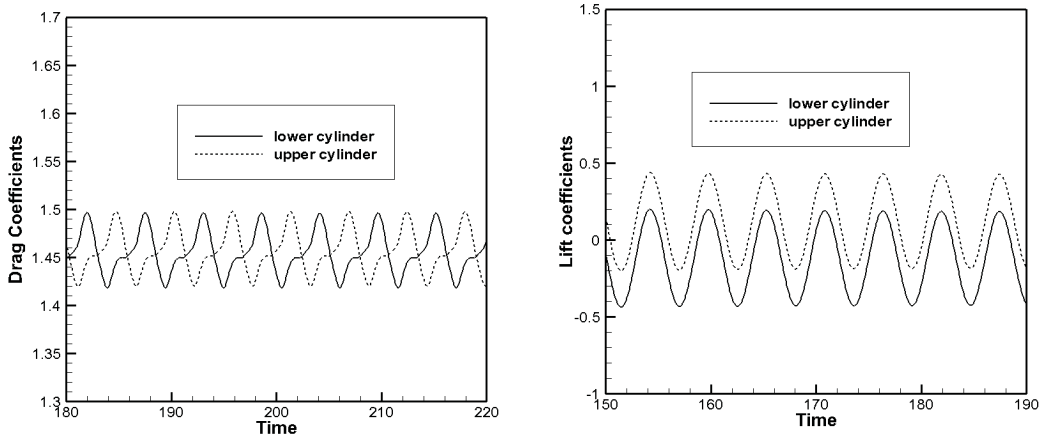


Figure 4.16 Drag and lift coefficients of flow past a pair of side-by-side cylinder (T=3D) at Re=100

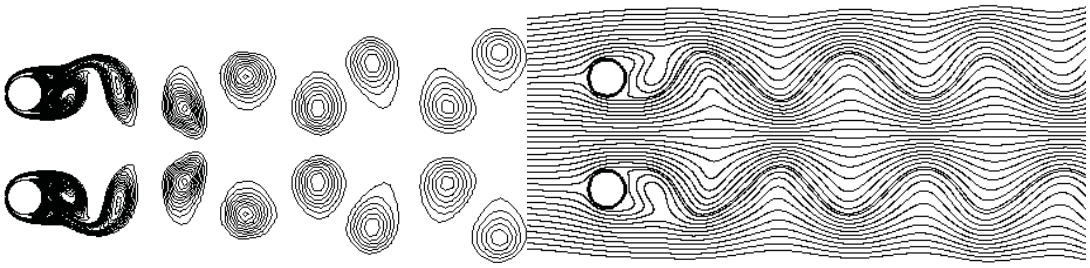


Figure 4.17 Vorticity contours and streamlines for side-by-side cylinders (T=3D) at Re=200

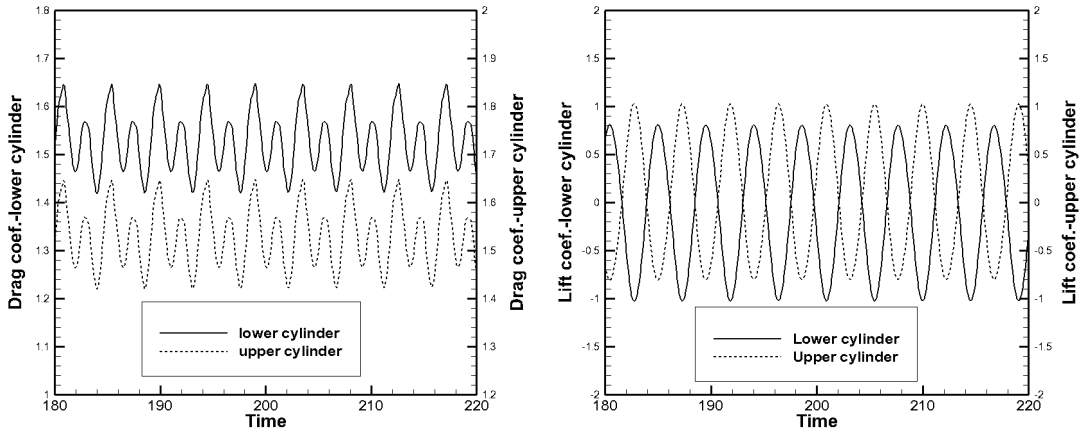


Figure 4.18 Drag and lift coefficients of side-by-side cylinders (T=3D) at Re=200

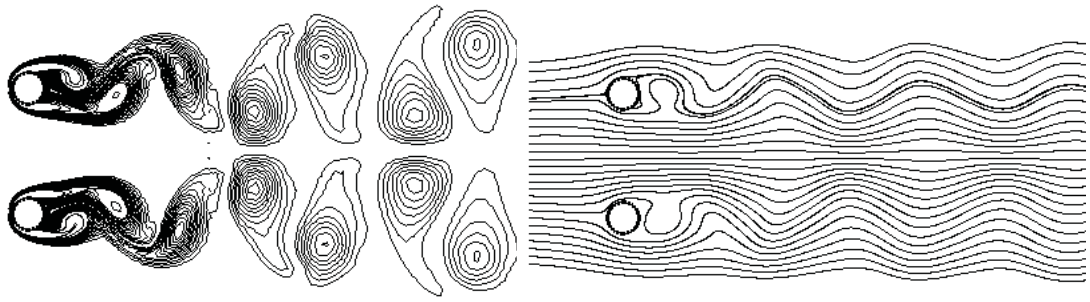


Figure 4.19 Vorticity contours and streamlines for side-by-side cylinders (T=4D) at Re=100

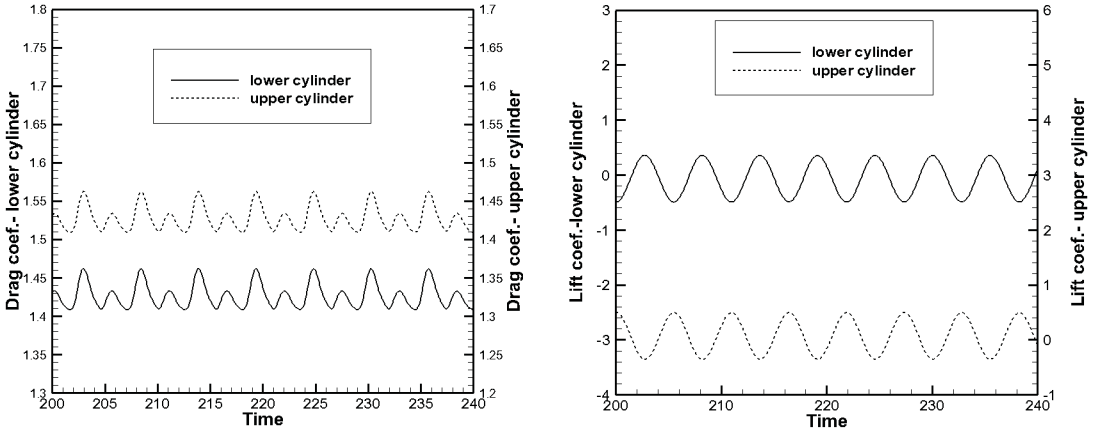


Figure 4.20 Drag and lift coefficients of side-by-side cylinder (T=4D) at Re=100

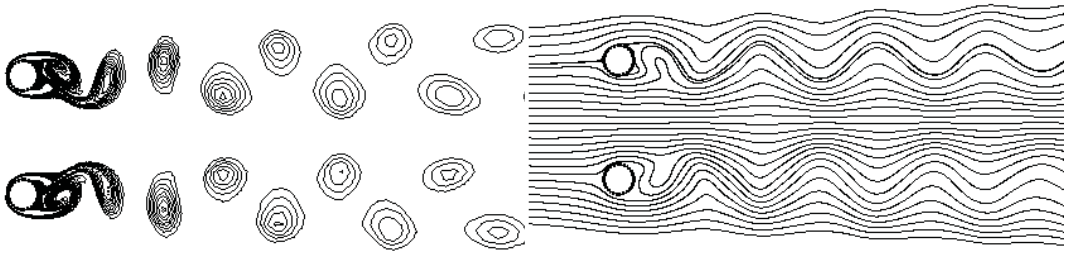


Figure 4.21 Vorticity contours and streamlines for side-by-side cylinders (T=4D) at Re=200

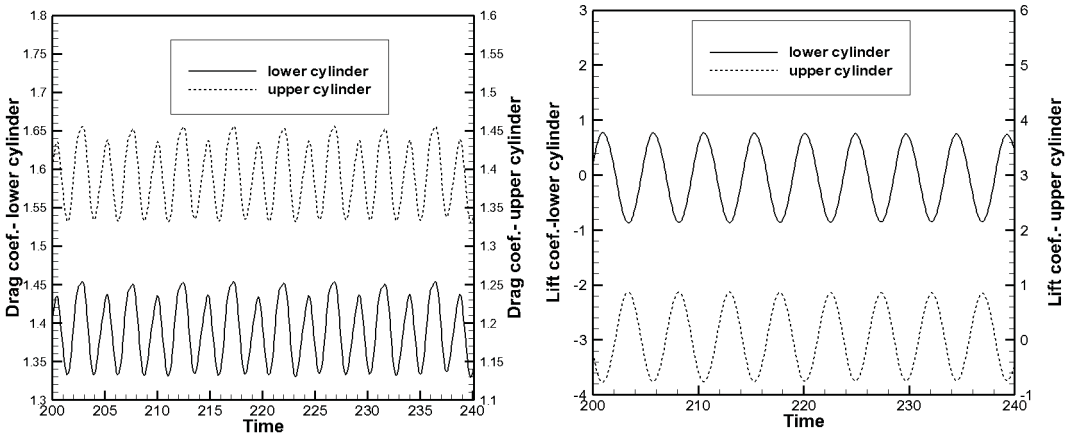


Figure 4.22 Drag and lift coefficients of side-by-side cylinder (T=4D) at Re=200

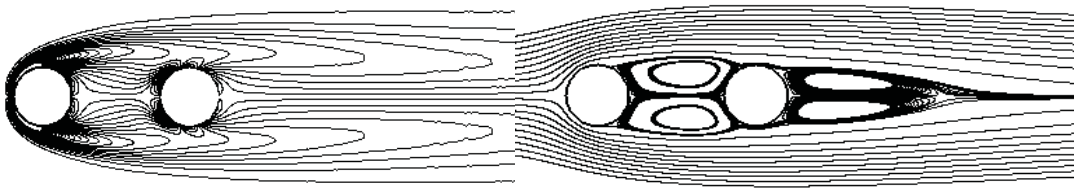


Figure 4.23 Vorticity and streamlines for tandem cylinders (L=2.5D) at Re=100

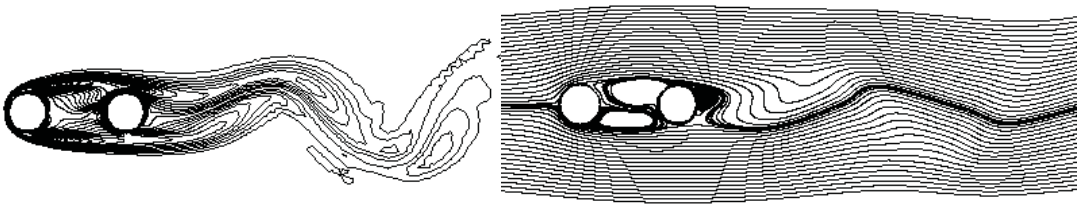


Figure 4.24 Vorticity and streamlines for tandem cylinders (L=2.5D) at Re=200

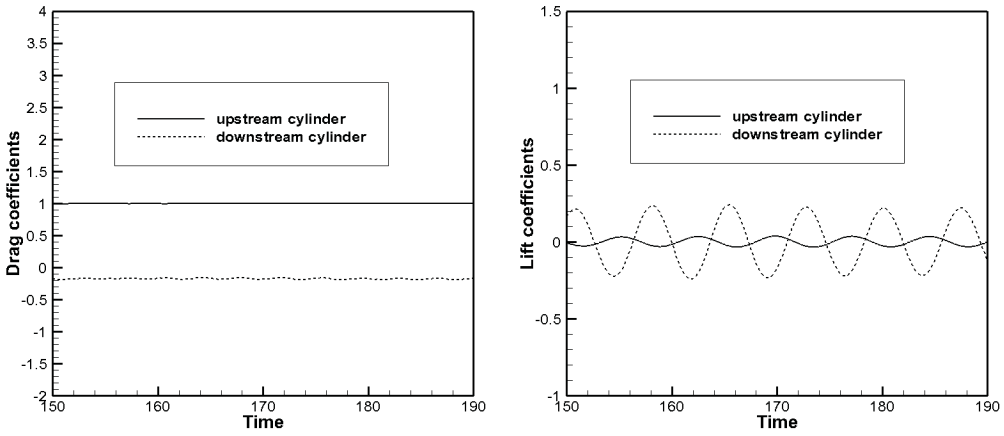


Figure 4.25 Drag and lift coefficients of tandem cylinders (L=2.5D) at Re=200

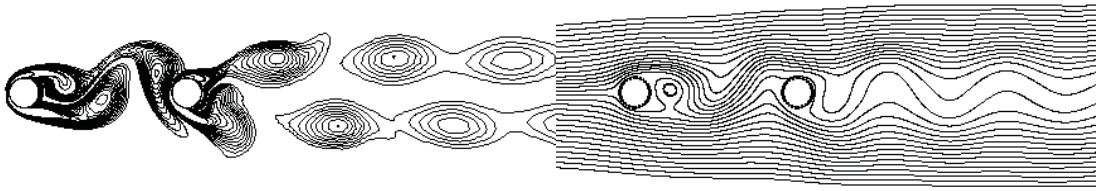


Figure 4.26 Vorticity and streamlines for tandem cylinders (L=5.5D) at Re=100

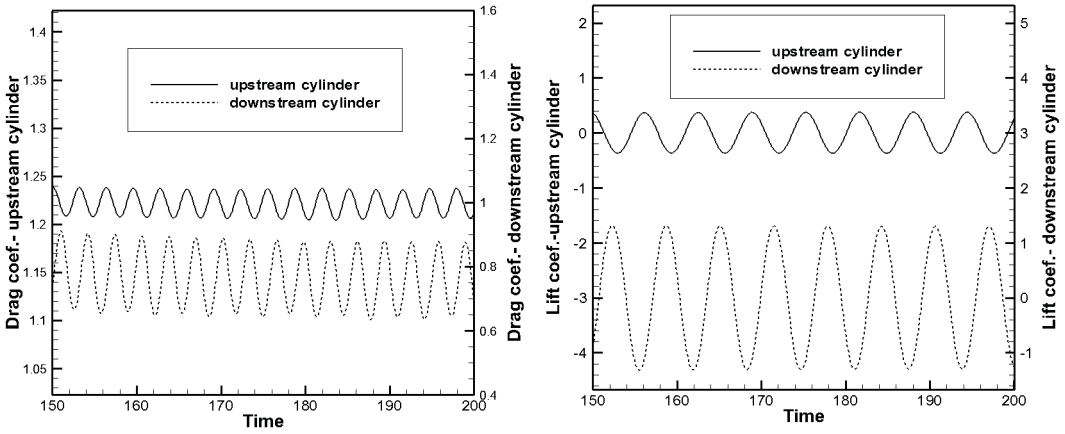


Figure 4.27 Drag and lift coefficients of tandem cylinders (L=5.5D) at Re=100

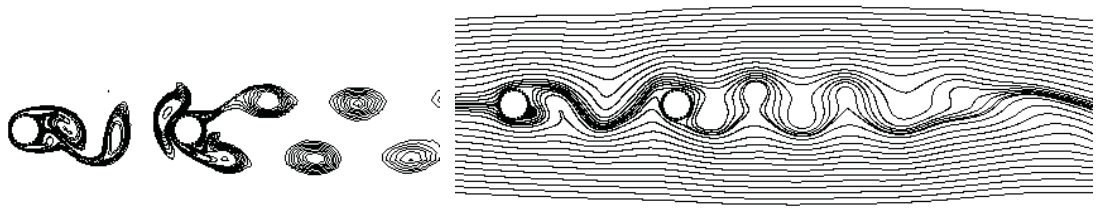


Figure 4.28 Vorticity and streamlines for tandem cylinders (L=5.5D) at Re=200

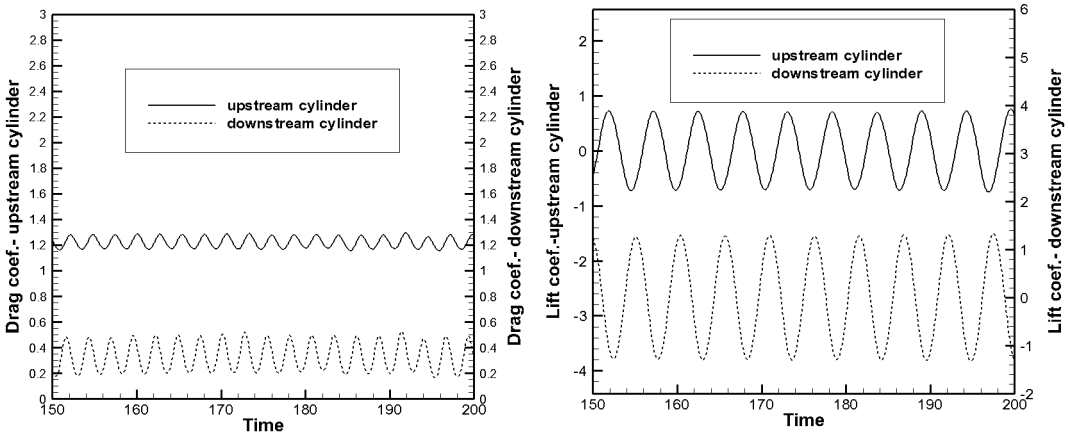
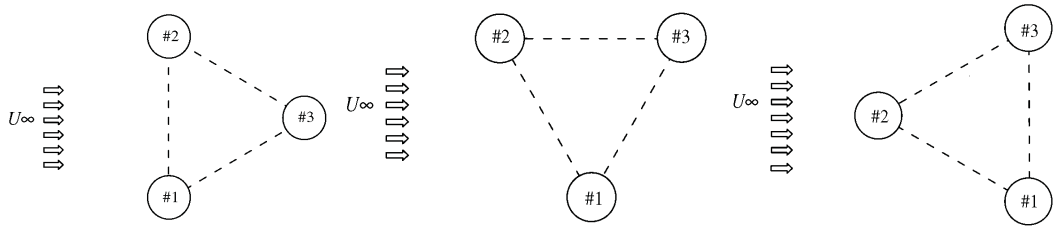
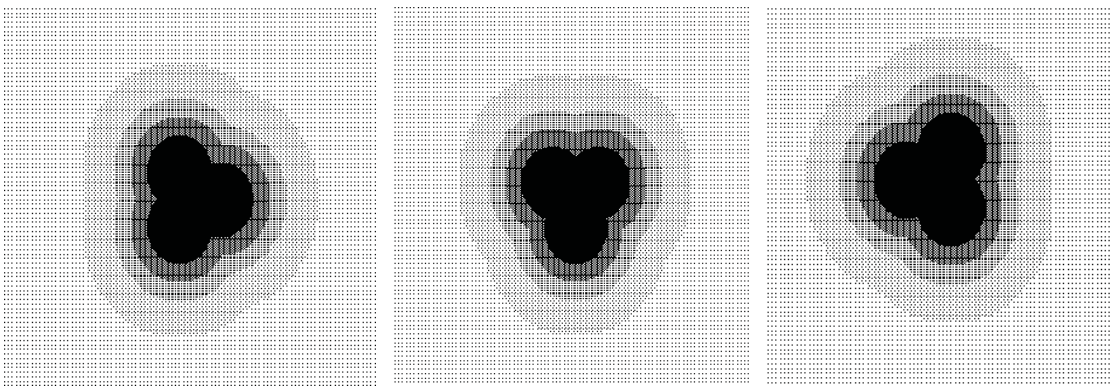


Figure 4.29 Drag and lift coefficients of tandem cylinders (L=5.5D) at Re=200



Type I Type II Type III

Figure 4.30 Configuration of different types of arrangement for flow past three cylinders



Type I Type II Type III

Figure 4.31 Local refined mesh for simulation of flow past three circular cylinders

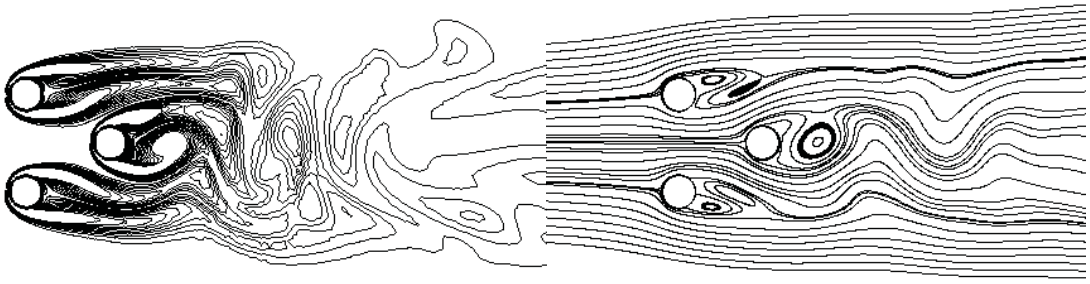


Figure 4.32 Instantaneous vorticity contours and streamlines for Type I (Re=100)

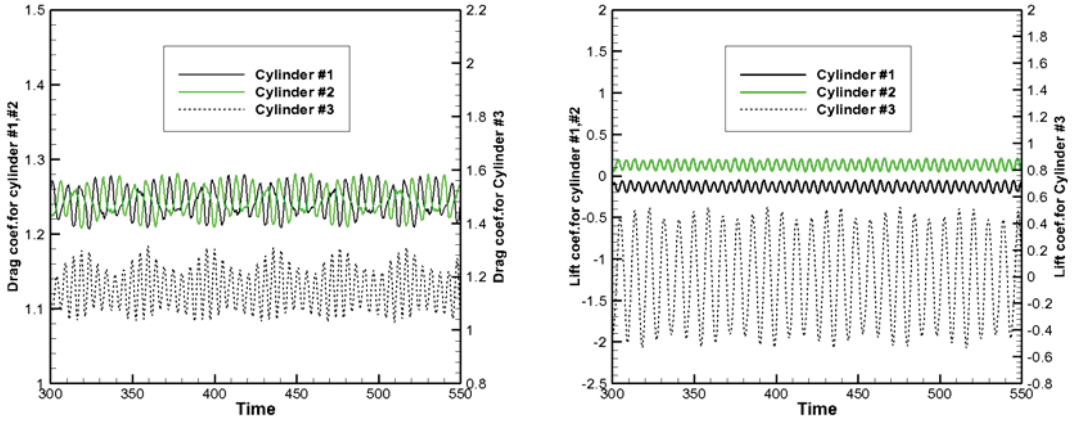


Figure 4.33 Drag and lift coefficients of flow past three cylinders (Type I) at Re=100

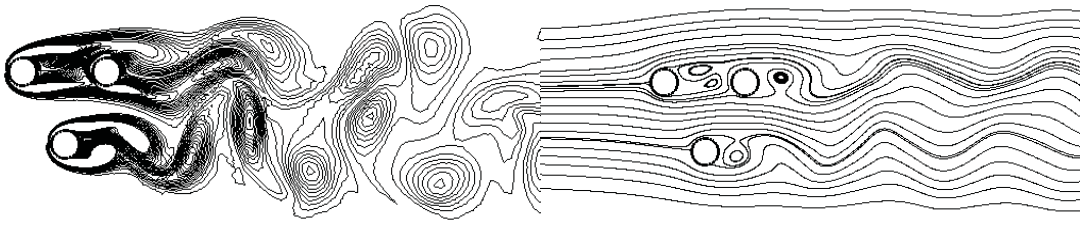


Figure 4.34 Instantaneous vorticity contours and streamlines for Type II (Re=100)

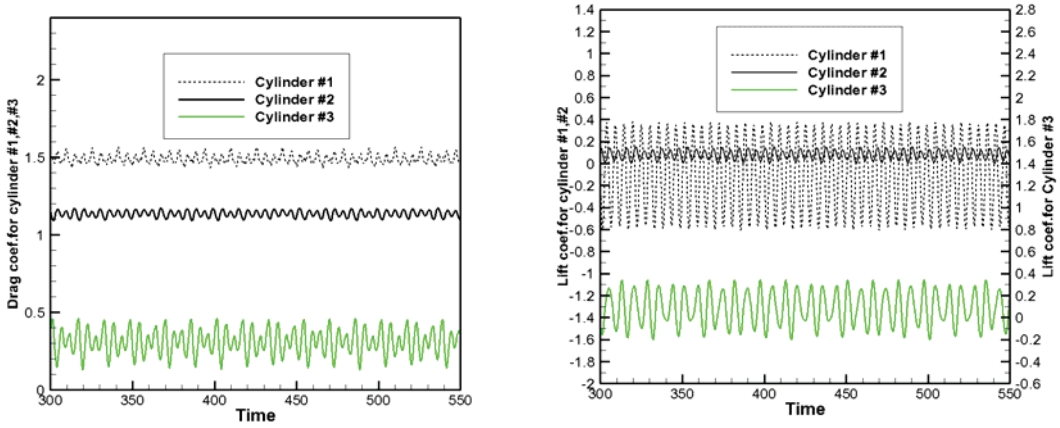


Figure 4.35 Drag and lift coefficients of flow past three cylinders (Type II) at Re=100

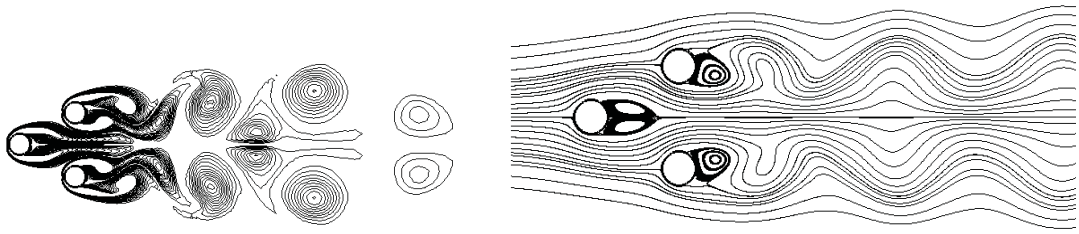


Figure 4.36 Instantaneous vorticity contours and streamlines for Type III (anti-phase)

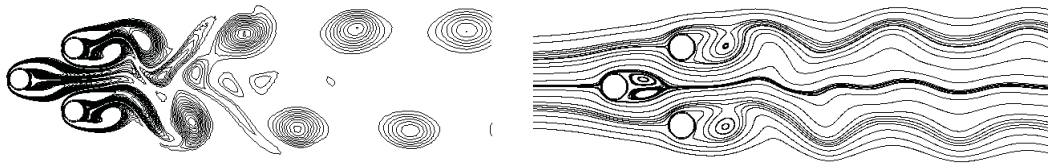


Figure 4.37 Instantaneous vorticity contours and streamlines for Type III (in-phase)

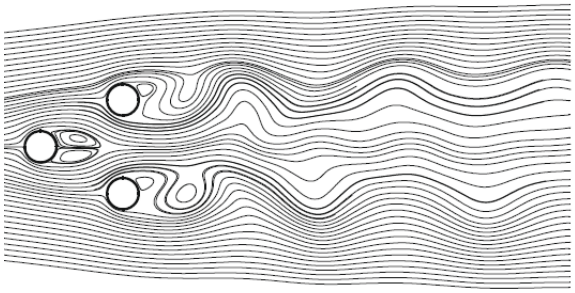


Figure 4.38 Instantaneous streamlines for Type III (in-phase) obtained by Bao et al (2010)

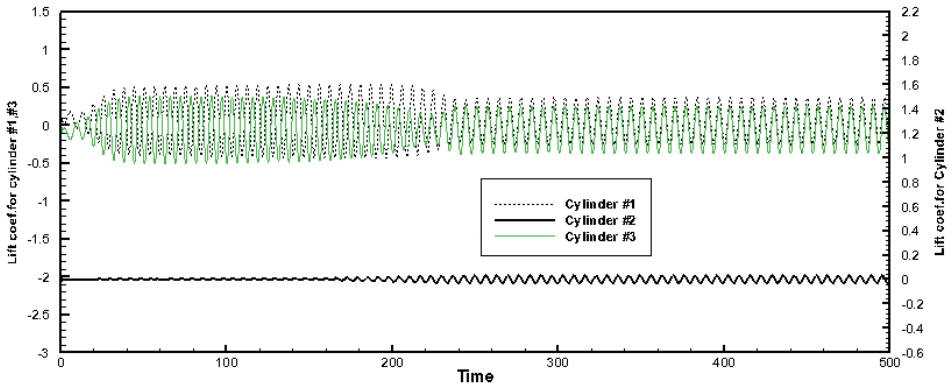


Figure 4.39 History of lift coefficients of three cylinders (Type III) at Re=100

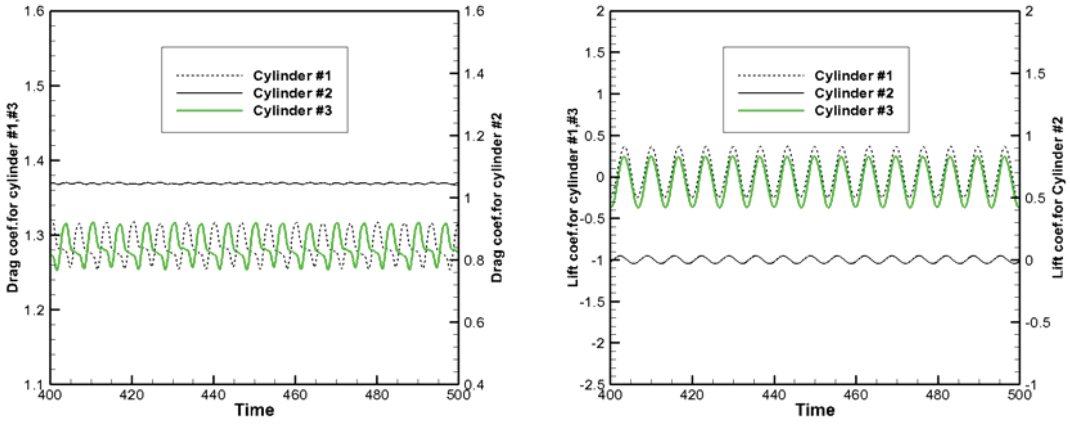


Figure 4.40 Drag and lift coefficients of flow past three cylinders (Type III) at Re=100 (In-phase)

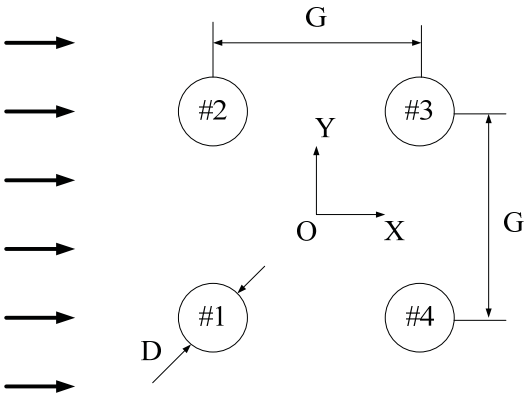


Figure 4.41 Configuration of flow past four equispaced cylinders

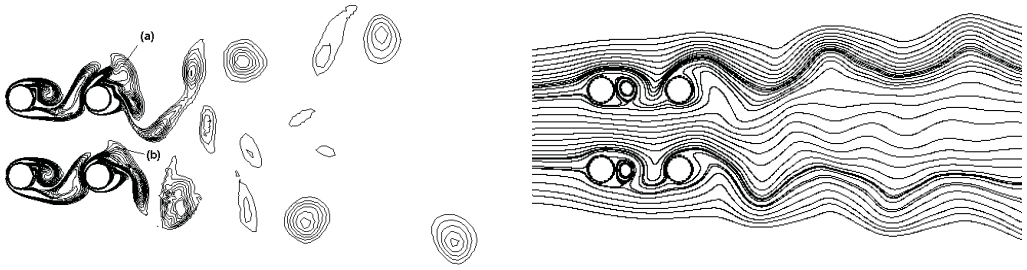
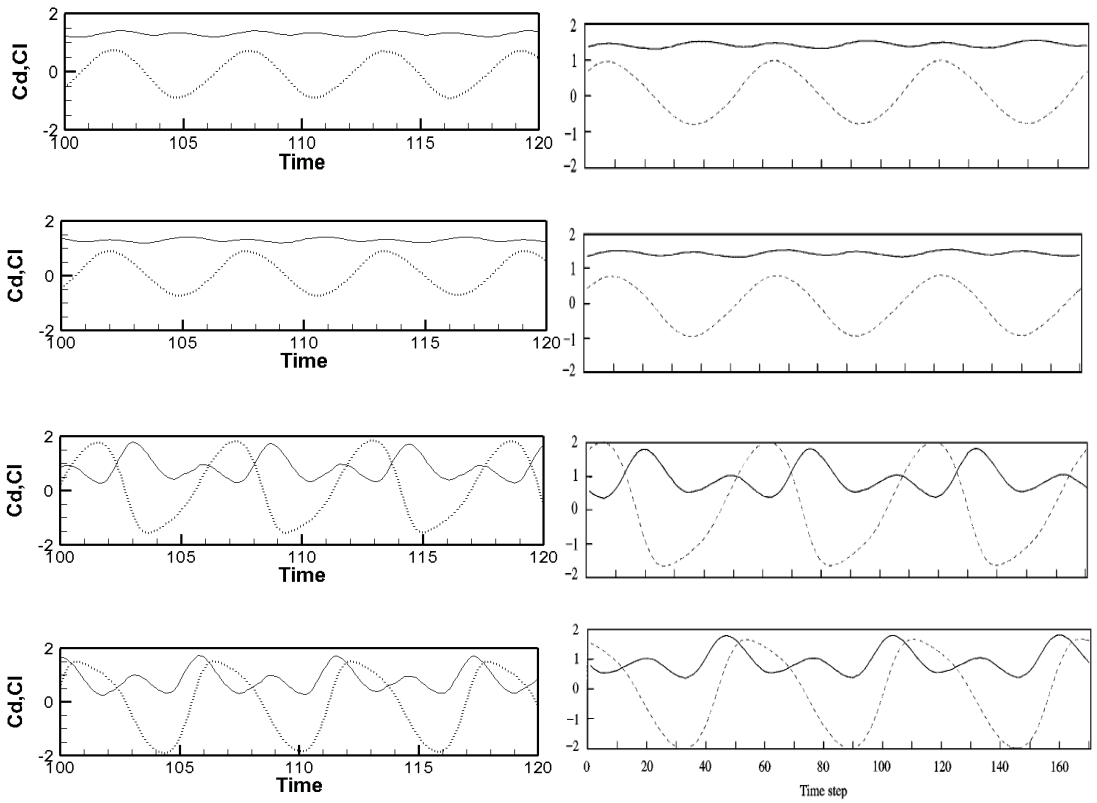


Figure 4.42 Instantaneous vorticity contours and streamlines for 4 equispaced cylinders at Re=200 and G=3D.



(a) left: present results (b)right: Farrant et al (2000)

Figure 4.43 Drag and lift coefficients C_D and C_L for 4 cylinders: the curves correspond to cylinders 1-4, Top to Bottom, respectively

CHAPTER 5

Application of LDFD and LDFD-IBM to Simulate Moving Boundary Flow Problems

One of big challenges in computational fluid dynamics (CFD) is to simulate flows around moving complex bodies. In classical numerical methods, one usually resorts to the fitted-body mesh strategies, either using the fixed reference frame (inertial reference frame) or moving the reference frame with the moving body (non-inertial reference frame). The first approach requires continuous re-meshing around the body (Tezduyar et al, 1992), which is very laborious and computational expensive. Comparatively, the second approach is more efficient and simpler for implementation, and thus is often adopted in practice (Lu and Dalton,1996; Guilmineau and Queutey, 2002). However, since the non-inertial system is an accelerated reference system associated with the moving body, in principle it cannot be applied to moving boundary problems with more than one bodies moving at different speeds.

As non-conforming mesh methods, LDFD method and LDFD-IBM can be easily extended to handle the moving boundary problems without the above mentioned drawbacks. In this chapter, the strategies of the LDFD method and LDFD-IBM for moving boundary problems for incompressible flows are presented, respectively. The numerical tests are carried out to validate the methods, including the simulation of flows past single/multiple moving objects.

5.1 Status changes in moving boundary problems

To apply the LDFD method and the LDFD-IBM for simulation of flows with moving bodies, the fixed reference frame is adopted, and the mesh is also fixed. The moving objects are immersed in the fixed mesh, and are allowed to move freely through the mesh lines.

From the implementation point of view, the main difference between flows with stationary boundaries (as shown in Chapters 2 and 4) and flows with moving boundaries lies in the fact that during the computation, the status of nodes changes in the latter but remains the same in the former. This is because when the solid body is moving, the position of the boundary on the Cartesian mesh changes with the time, and consequently the status of the mesh nodes changes. The movement of the rigid objects can be tracked by the status changes of mesh nodes in the domain. These statuses are the same as those defined in the LDFD method and LDFD-IBM,i.e., STATUS=1 represents the interior point which is in the flow domain and not adjacent to the solid boundary; STATUS=2 represents the interior dependent point near the immersed boundary and inside the flow; STATUS=3 represents the exterior dependent point near the immersed boundary but inside of the solid body; STATUS=4 represents the outside point which is located in the solid body but not adjacent to the solid boundary.

Figure 5.1 shows an example. For the simulation, we need to pay attention to the following two situations:

Situation 1: Suppose the position of the boundary at the time level n is represented by the solid curve, and the boundary at the time level n+1 is represented by the dashed curve, from the time level n to n+1, the status of $A2$ is changed from an exterior dependent node at the time level n to an interior dependent node at the time level n+1, i.e., $\text{STATUS}^{n}(A2)=3 \rightarrow \text{STATUS}^{n+1}(A2)=2$.

Situation 2: Suppose the solid curve denotes the boundary at the time level n+1, the dashed curve denotes the boundary at the time level n, then from the time level n to n+1, the status of $A2$ is changed from interior dependent node at the time level n to the exterior dependent node at the time level n+1, i.e., $\text{STATUS}^{n}(A2)=2 \rightarrow \text{STATUS}^{n+1}(A2)=3$.

The change of node status suggests that the solid node becomes a fluid node, or vice versa. These new fluid nodes may pose a challenge in the discretization of the time derivative, i.e., we need to know their flow variables at the previous time level although they are solid nodes at that time. In the following, we therefore focus on how to handle this issue in LDFD and LDFD-IBM.

If the time interval Δt is sufficiently small to make sure that, when the moving body sweeps across the mesh, it moves less than one mesh spacing in one time step. In other words, it makes the smooth change of status at a mesh node, which follows a certain sequence, i.e., from STATUS=1 →STATUS=2 →STATUS=3 → STATUS=4 or vice versa. In this way, the boundary of the solid bodies moving across the mesh only incurs the status exchange between exterior dependent nodes and interior dependent nodes along the moving boundary in one time step.

In *Situation 1*, the mesh node is the exterior dependent point at the previous time level, its functional values at that time level have been obtained through the extrapolation for the LDFD method. From the LDFD point of view, these values obtained from extrapolation are indeed the flow values at the mesh node in the previous time level. Therefore, these values can be used to implement the time derivative discretization in the moving boundary problems in the current time level when the mesh node becomes the interior dependent point. In *Situation 2*, the mesh node is exterior dependent point at the current time level, no discretization is needed for LDFD method.

For the LDFD-IBM, because the variables at every mesh node are calculated by solving the governing equations, the change of node status seems not to be a problem. However, the time advancing of the momentum equation needs to be treated carefully to avoid introducing additional numerical errors (see the details in Section 5.2.2).

5.2 Methodologies and procedures

5.2.1 LDFD for moving boundary problems

As shown in Figure 5.1, suppose that the solid curve represents the position of the boundary. The functional value at the exterior dependent point *A2* needs to be calculated for the use in the discretization of derivatives at $A1$ and $B2$. To uniquely determine the functional value at $A2$, particularly for the time-derivative discretization, a local-polynomial-based extrapolation along the normal direction to the boundary $\vec{n}$ through *A2* is constructed. For example, the values at points $P1$, $P2$, $P3$ (as shown in Figure 5.1),

which are the intersection points of the perpendicular line to the boundary (through *A2*) and the grid lines, can be used in the construction of the extrapolation scheme for an arbitrary smooth function *f*, which yields

$$f_{A2} = \frac{(r_{A2} - r_{P2})(r_{A2} - r_{P3})}{(r_{P1} - r_{P2})(r_{P1} - r_{P3})} f_{P1} + \frac{(r_{A2} - r_{P3})(r_{A2} - r_{P1})}{(r_{P2} - r_{P3})(r_{P2} - r_{P1})} f_{P2} + \frac{(r_{A2} - r_{P1})(r_{A2} - r_{P2})}{(r_{P3} - r_{P1})(r_{P3} - r_{P2})} f_{P3} \quad (5.1)$$

where r is the distance from the certain point to mesh node *A2*, an example being $r_{P2} = \sqrt{(x_{P2} - x_{A2})^2 + (y_{P2} - y_{A2})^2}$. The functional values at the points *P2*, *P3* can be calculated directly by interpolation between nodes *A1* and *B1* (for the point *P2*), nodes *A0* and *B0* (for the point *P3*). The functional value of the boundary point *P1* is obtained directly from the boundary condition.

For *Situation 1* in Section 5.1 where $\text{STATUS}^{n}(A2)=3 \rightarrow \text{STATUS}^{n+1}(A2)=2$, at the time level n+1, *A2* is the interior node, at which the governing equations need to be solved. For the unsteady flow problem, the functional value at *A2* at the last time level is needed for the discretization of the time derivative. In the LDFD method, the functional value at *A2* at the last time level has been known from extrapolation (Eq. (5.1)). That is the beauty of LDFD method for moving boundary problems.

For *Situation 2* where $\text{STATUS}^{n}(A2)=2 \rightarrow \text{STATUS}^{n+1}(A2)=3$, the mesh node *A2* at the time level n is inside the fluid domain, but in the time level n+1, *A2* becomes the node outside of the domain. Since discretization at time level n+1 is only needed at the interior node, the solution of *A2* from time level n is discarded. The solution at *A2* at the time level n+1 can be approximated by extrapolation using Eq. (5.1).

Boundary conditions on the moving boundary

Velocity boundary conditions

For incompressible flows, the no-slip condition is applied. For the moving boundary problem, the velocity at intersection point of mesh lines and the boundary (such as $P1$) is set to be the same as the solid boundary velocity.

Pressure boundary conditions

The pressure on the boundary surface can be determined from the momentum equation normal to the solid wall. For moving boundary problems, the solid boundary velocity is not zero, and thus the gradient of the pressure along the normal direction can be written as,

$$\left(\frac{\partial p}{\partial n}\right)_{P1} = -\left.\frac{\partial (u_n)}{\partial t}\right|_{P1} + \frac{1}{\mathrm{Re}}\left[\frac{\partial^2 (u_n)}{\partial n^2}\right]_{P1} + \frac{\rho_{P1}(u_t)_{P1}^2}{R_{P1}} \tag{5.2}$$

where R_{P1} is the local radius of curvature along the solid boundary at the intersection point $P1$, $(u_n)_{p1}, (u_t)_{p1}$ are the normal and tangential components of the velocity at $P1$. If the trajectory of the moving object is prescribed, the first term at RHS of Eq. (5.2) is known. The derivatives in Eq. (5.2) can be discretized in the same way as described in Section 2.3.

5.2.2 LDFD-IBM for moving boundary problem

When applying the LDFD-IBM to moving boundary problems, it includes the simultaneous update of the velocity and the location of the moving boundaries. Let us

take Figure 5.1 as an example again. Suppose that at the time level n the boundary is located at position $\mathbf{X}^n$ (solid line in Figure 5.1) with velocity $\mathbf{U}_b^n$. At the time level n+1, the boundary moves to position $\mathbf{X}^{n+1}$ (dashed line in Figure 5.1) with velocity $\mathbf{U}_b^{n+1}$. Here we consider the two following effects of moving object separately:

(1) the effect from the change of the position of the boundaries, which we call the status-changing effect since it corresponds to the status change of mesh nodes;

(2) the effect from the change of the velocity of the boundaries, which we call the acceleration effect.

Therefore, LDFD-IBM forcing procedure for moving boundary problem basically consists of two steps:

Step 1: Estimate the flow field by considering the change in the position of the moving object only;

In this step, we only consider the status-changing effect. Suppose that the solutions of N-S equations such as $\mathbf{u}^n$ and p^n are known. We modify the velocity field from $\mathbf{u}^n$ to $\hat{\mathbf{u}}^n$ at time level n to account for the location change of the boundaries:

1) change the status of mesh nodes according to the new position of boundary $\mathbf{X}^{n+1}$;
2) keep the velocity unchanged for mesh nodes with STATUS=1 (i.e. interior nodes) , i.e. $\hat{\mathbf{u}}^n = \mathbf{u}^n$.
3) set the velocity at mesh nodes with STATUS=4 (i.e. exterior nodes not adjacent to the boundary $\mathbf{X}^{n+1}$) and STATUS=3 (i.e. exterior dependent points) to $\mathbf{U}_b^n$, i.e., $\hat{\mathbf{u}}^n = \mathbf{U}_b^n$. The velocity at boundary points which are the intersection of the

boundary and the mesh lines (i.e. Lagrangian points in conventional IBM) is also set as $\hat{\mathbf{u}}^n = \mathbf{U}_b^n$

4) use interpolation scheme (like Eq.(4.5)) to determine the velocity at those nodes with STATUS=2 (i.e. interior dependent nodes). There are three determining factors for the interpolation scheme, i.e., the position of boundary $\mathbf{X}^{n+1}$, the velocity at the mesh nodes with STATUS=1 ($\hat{\mathbf{u}}^n = \mathbf{u}^n$) and the velocity at Lagrangian points ($\hat{\mathbf{u}}^n = \mathbf{U}_b^n$), thus we can use *interpolation*($\mathbf{X}^{n+1}, \mathbf{u}^n, \mathbf{U}_b^n$) to represent the interpolation value.

Therefore, the modified velocity field $\hat{\mathbf{u}}^n$ at time level *n* can be written as

$$\hat{\mathbf{u}}_i^n = \begin{cases} \mathbf{u}_i^n & \text{if STATUS}(i, \mathbf{X}^{n+1}) = 1 \\ interpolation(\mathbf{X}^{n+1}, \mathbf{u}^n, \mathbf{U}_b^n) & \text{if STATUS}(i, \mathbf{X}^{n+1}) = 2 \\ \mathbf{U}_b^n & \text{if STATUS}(i, \mathbf{X}^{n+1}) = 3 \text{ or } 4 \end{cases} \tag{5.5}$$

Based on this modified velocity field $\hat{\mathbf{u}}^n$, we estimate the velocity field $\hat{\mathbf{u}}^{n+1}$ at time level *n*+1 without any immersed object effect via solving the following equation:

$$\frac{\hat{\mathbf{u}}^{n+1} - \hat{\mathbf{u}}^n}{\Delta t} = -H(\hat{\mathbf{u}}^n) - Gp^n + \frac{1}{\mathrm{Re}} L(\hat{\mathbf{u}}^n) \tag{5.6}$$

where H denotes the discrete advection operator, G is the discrete gradient operator, L is the discrete Laplace operator.

It should be noted that neither explicit force term nor implicit force term is presented in Eq. (5.6). In other words, $\hat{\mathbf{u}}^{n+1}$ is calculated without considering the immersed body in the flow field.

Step 2: Estimate the flow field by considering the change in the velocity of the moving object;

In this step, we include the acceleration effect into consideration, i.e., the velocity of boundary is changed from $\mathbf{U}_b^n$ to $\mathbf{U}_b^{n+1}$ at the same position $\mathbf{X}^{n+1}$. To do that, we solve the momentum equation with the force term f_M added explicitly, i.e.,

$$\frac{\tilde{\mathbf{u}}^{n+1}-\hat{\mathbf{u}}^{n}}{\Delta t}=-H(\hat{\mathbf{u}}^{n})-Gp^{n}+\frac{1}{\mathrm{Re}}L(\hat{\mathbf{u}}^{n})+f_M^{n+1} \tag{5.7}$$

where $\tilde{\mathbf{u}}^{n+1}$ is the estimate of the velocity field at time level n+1 which takes both the status-changing effect and the acceleration effect into consideration. Here the time derivative is discretized via Euler explicit scheme for the convenience of analysis. In practice, the explicit three-step formulation described in Section 2.5.2.2 is used for the discretization of the time derivative.

The force term f_M can be calculated by

$$f_M^{n+1}=\frac{\tilde{\mathbf{u}}_b^{n+1}-\hat{\mathbf{u}}^{n+1}}{\Delta t} \tag{5.8}$$

where $\tilde{\mathbf{u}}_b^{n+1}$ is obtained by LDFD-IBM velocity updating described in Chapter 4, i.e.,

$$\tilde{\mathbf{u}}_b^{n+1} = \begin{cases} \hat{\mathbf{u}}_i^{n+1} & \text{if STATUS}(i, \mathbf{X}^{n+1}) = 1 \\ interpolation(\mathbf{X}^{n+1}, \hat{\mathbf{u}}^{n+1}, \mathbf{U}_b^{n+1}) & \text{if STATUS}(i, \mathbf{X}^{n+1}) = 2 \\ \mathbf{U}_b^{n+1`} & \text{if STATUS}(i, \mathbf{X}^{n+1}) = 3 \text{ or } 4 \end{cases} \tag{5.9}$$

Generally, this solution procedure can be used for the flow problems with moving or stationary boundaries. For the flow problem with stationary boundaries, since there is no status change for the mesh nodes, $\hat{\mathbf{u}}^n = \mathbf{u}^n$. Therefore, f_M is equivalent to the velocity updating in LDFD-IBM for stationary boundary problem in Chapter 4.

Note that the force term f_M in Eq. (5.8) is simply a Newtonian acceleration at time-level n+1. If the trajectory of the moving object is prescribed, say $\mathbf{X} = \mathbf{X}_e(t)$, the acceleration of solid body can be obtained directly, i.e.

$$\mathbf{a} = \frac{d^2\mathbf{X}_e}{dt^2}. \tag{5.10}$$

Eq. (5.10) can be used to replace Eq. (5.8) as the force term in Eq. (5.7), for the exterior dependent nodes and exterior nodes, because they are supposed to have the same speed as the moving object. However, when one considers more general cases, for example, Fluid-Solid Interaction (FSI), we still need to calculate f_M according to Eq. (5.8)**.**

After solving Eq. (5.7), one can follow the same solution procedure of LDFD-IBM for simulation of flow problems with stationary object through Eqs. (4.14)~ (4.16) to obtain the velocity and pressure field at time level n+1.

Therefore, the basic procedure of the LDFD-IBM for solving moving boundary problems can be summarized by the following sequence (advancing the solution from n to n+1 time level):

1) Use the solution at the n time level to initialize the velocity and pressure fields $\mathbf{u}^n, p^n$ (initially set f_M as zero if t=0);

2) Determine the new position of the immersed object $\mathbf{X}^{n+1}$; if the trajectory of the moving object is prescribed, this job can be easily done.

3) Determine the status of the mesh nodes when the moving object is at position $\mathbf{X}^{n+1}$;

4) obtain the modified velocity field $\hat{\mathbf{u}}^n$ via Eq. (5.5) and solve for $\hat{\mathbf{u}}^{n+1}$ from Eq.(5.6) without any force term;

5) solve for $\tilde{\mathbf{u}}^{n+1}$ from Eq. (5.7) in which f_M is added explicitly via Eq.(5.8);

6) solve Eq. (4.14) ~ (4.16) in the whole flow field to get $\mathbf{u}^{n+1}, p^{n+1}$.

5.3 Application of LDFD and LDFD-IBM to moving boundary problems

In this section, the LDFD and LDFD-IBM's strategies for moving boundary problems are used to simulate flow problems with the single or multiple moving bodies. Like in stationary boundary problems, the application of LDFD and LDFD-IBM to the moving boundary problems uses the second-order finite difference scheme for the approximation of spatial derivatives and the explicit three-step formulation (as shown in Section 2.5.2.2) for the approximation of the time derivative.

5.3.1 Flow past an oscillating circular cylinder

The problem of a cylinder oscillating laterally (cross-flow) in the free stream is a classic flow problem with moving boundaries. There are various experimental and numerical investigations available (Lu and Dalton 1996; Guilmineau and Queutey 2002).

The computational domain is shown in Figure 5.2, where D is the diameter of the cylinder. The circular cylinder is moved vertically with the speed v_T,

$$v_T = \frac{dy_e}{dt} \tag{5.11}$$

where $y_e = A_e \sin(2\pi f_e t)$, with A_e and f_e are the oscillating amplitude and excitation frequency, respectively. f_e is very important for the flow structure. The synchronization phenomenon has been observed in the wake of the cylinder when the frequency of vortex shedding in the wake synchronizes with the frequency of an imposed perturbation, i.e., this kind of synchronization occurs around f_e / f_o ~1, where f_e is the excitation frequency and f_o is the vortex-shedding frequency from the stationary cylinder. In this work, both LDFD and LDFD-IBM's strategies for moving boundary problems are used to perform calculations for Re=185, $A_e / D = 0.2$, where D is the diameter of the cylinder. The range of f_e / f_o=1.1, 1.12, 1.2 is considered. All the simulations start from the stable state of the flow past a stationary cylinder. Therefore, the field information of the flow past a non-oscillating cylinder at Re=185 reported in Chapter 4 can be used here, and f_o can be obtained.

An initial uniform Cartesian mesh of 161 x 121 is used as the background mesh. After 8 levels of mesh refinement with stencil refinement technique in Chapter 3, the final mesh has a total of 54002 nodes, as shown in Figure 5.3. Note that the finest mesh covers the area where the oscillating cylinder moves. To make sure that the moving distance of each node in one time step is less than the nodal distance, the time interval should be sufficiently small. The mesh spacing of the finest mesh (which is located at solid boundary and its vicinity) is about 0.0125, and the maximum estimating value of the velocity is about 2. So the time step to sweep one mesh spacing must be less than 0.0125/2=0.00625, and we select a time step of 0.002.

The grid independence study is carried out for the case with the excitation frequency of f_e / f_o=1.1. The time averaged drag coefficient and the Strouhal number *St* are reported in Table 5.1. It is found that the results on Grid 2 have a difference about 2% for the average drag coefficient as compared to those on Grid 3 (the finest grid). The Strouhal numbers for both grids are the same. This suggests that the resolution for Grid 2 is sufficiently fine to obtain an accurate solution. For the balance of accuracy and efficiency, we choose Grid 2 (total 54002 nodes) in the rest of this study. The results obtained by Guilmineau and Queutey (2002) are also included in the table, and serve as the benchmark solution. It can be seen from Table 5.1 that a good agreement is achieved between our results and those of Guilmineau and Queutey (2002).

Figures 5.4~5.6 show the drag and lift coefficients for different values of f_e / f_o obtained by the LDFD method and the LDFD-IBM, as well as the results from Guilmineau and Queutey (2002). It is found that the drag and lift coefficients exhibit regular signs of the influence of a higher harmonic wave. As the excitation frequency increases, the beating frequency decreases. The present results agree reasonably well with those of Guilmineau and Queutey (2002). It demonstrates that the moving boundary strategies of both LDFD and LDFD-IBM work very well.

The numerical results from LDFD method are shown in Figure 5.7, which qualitatively appear to be in good agreement with that obtained by Guilmineau and Queutey (2002). Figure 5.7a shows the instantaneous streamlines when the oscillating cylinder is at the extreme upper position for the case of f_e / f_o=1.1. The figure shows two saddle points in the form of intersecting streamlines. The centers of the closed streamlines suggest the existence of vorticity concentrations in those regions. This concentration of vorticity involves the entire near wake and results in a tighter vortex structure. The corresponding contours of vorticity can be seen in Figure 5.7b. The upper vortex has been diminished in strength to the extent that the lower vortex has become the dominant vortex and the upper vortex has rolled up tightly behind the cylinder.

5.3.2 Two cylinders moving with respect to each other

To demonstrate the ability of present methods to simulate flows past multiple moving objects, the flow past two cylinders moving with respect to one another is selected. This

example was originally investigated numerically by Russell and Wang (2003) and Xu and Wang (2006). The initial geometry for this test is shown in Figure 5.8. The far field boundary conditions are rigid walls. To avoid the impulsive start of the cylinders, both cylinders are oscillated about their initial positions for two periods (16 seconds), and then towards to each other, at a speed corresponding to Re=40. The motion of the lower cylinder is given by

$$x_{lc} = \begin{cases} \frac{1}{4}\sin(\frac{\pi t}{4}), & 0 \le t \le 16 \\ t - 16, & 16 \le t \le 32 \end{cases} \tag{5.12}$$

and the motion of the upper cylinder is given by

$$x_{uc} = \begin{cases} 16 - \frac{1}{4}\sin(\frac{\pi t}{4}), & 0 \le t \le 16 \\ 32 - t, & 16 \le t \le 32 \end{cases} \tag{5.13}$$

Due to good efficiency, LDFD-IBM is used to simulate this problem. The Cartesian mesh is refined by the stencil adaptive mesh refinement algorithm introduced in Chapter 3. The finest mesh covers the area where the oscillating cylinders move. The mesh spacing near the cylinders is about 0.0167D.

Fig. 5.9 shows the evolution of lift and drag coefficients for one of the cylinders versus time. The results from Xu and Wang (2006) are also included for comparison. Very good agreement is observed for present method.

It is found from Figure 5.9 that, when two cylinders approach each other, there is an increase in drag, implying that the 2 cylinders are repulsive to each other. When they move in close proximity, there is a drop in drag which means that they are attractive to each other. The drag then rises back when they move away.

Figures 5.10 and 5.11 present the vorticity and pressure contours of LDFD-IBM results when the two cylinders are closest to each other at t=24, and Figure 5.12 and 5.13 shows the vorticity and pressure field at t=32 when the cylinders are separated from each other by a distance of 16. The results from Xu and Wang (2006) as the benchmark solutions are also included. It is found that our results agree with those of Xu and Wang (2006) very well. It demonstrates that our moving boundary solvers can simulate the multi-bodies moving problems easily and accurately.

5.4 Concluding remarks

The different strategies of the LDFD method and the LDFD-IBM for simulating the moving boundary problems are presented. The consideration of status-changing effect plays a very important role in designing the solution procedures for moving boundary problems. LDFD can be applied to moving boundary problem directly if the time step is sufficiently small to make the status change a smooth one. A two-step forcing procedure is adopted in LDFD-IBM to consider status-changing effect as well as the acceleration effect of the moving objects. The methods have been applied to flows with single/multiple moving bodies, and the results are in good agreement with the benchmark solutions in the literature.

Table 5.1 Numerical and experimental values of $\overline{C}_D$, St at Re=185, A_e / D=0.2 , f_e / f_o=1.1 (Oscillating cylinder case)

For grid independent study:

Grid 1: initial mesh 161×121, after 6 levels mesh refinement totally 36155 mesh nodes;

Grid 2: initial mesh 161×121, after 8 levels mesh refinement totally 54002 mesh nodes;

Grid 3: initial mesh 161×121, after 10 levels mesh refinement totally 87557 mesh nodes;

		$\overline{C}_d$	St
LDFD method with grid independent study	Grid 1	1.412	0.240
	Grid 2	1.484	0.214
	Grid 3	1.454	0.214
Guilmineau and Queutey (2002) (mesh =240×200)		1.420	0.214

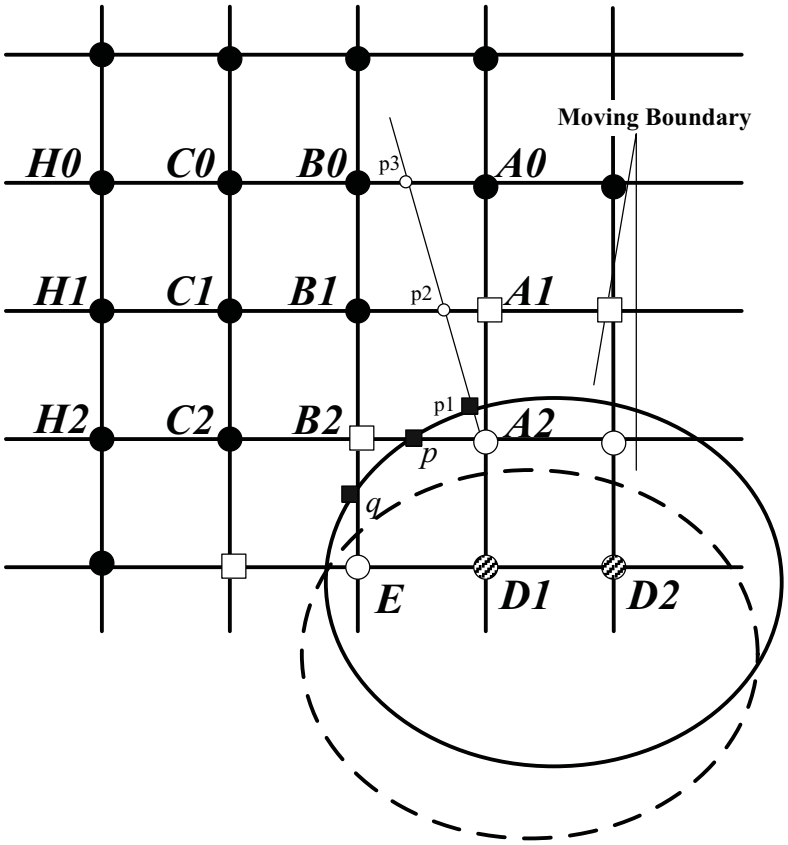


Figure 5.1 Configuration of moving boundary problem

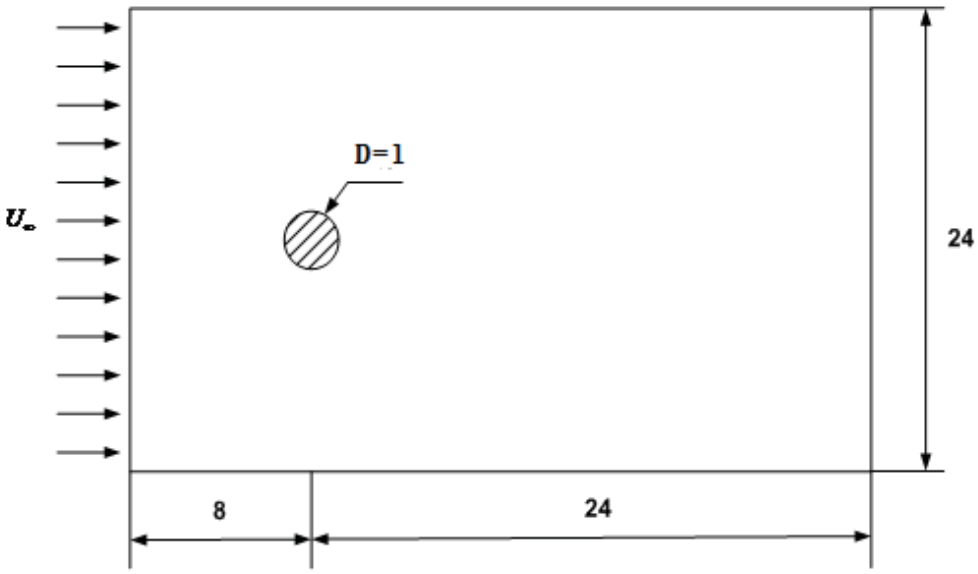


Figure 5.2 Computational domain for the flow around an oscillating circular cylinder

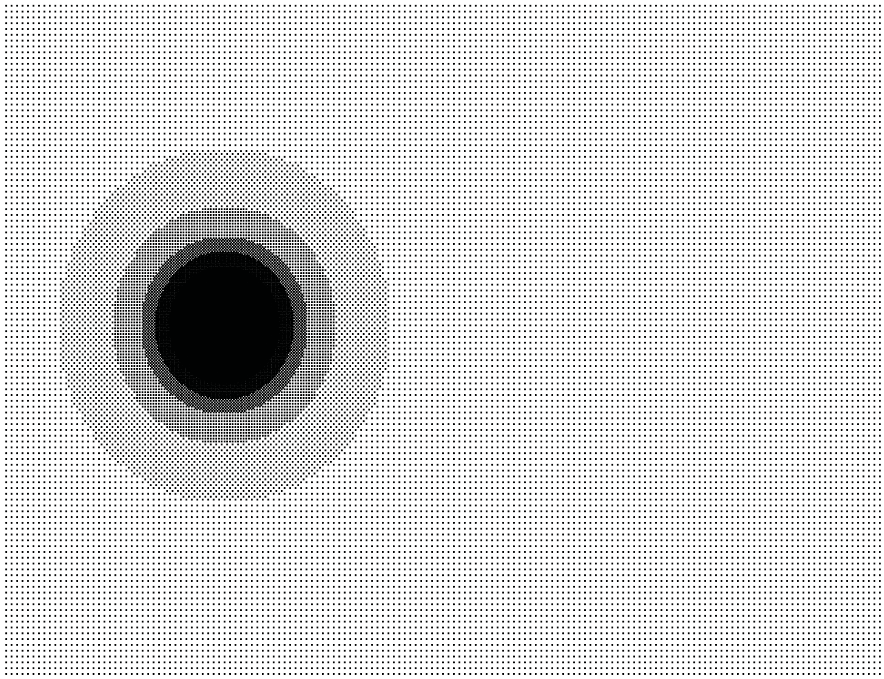
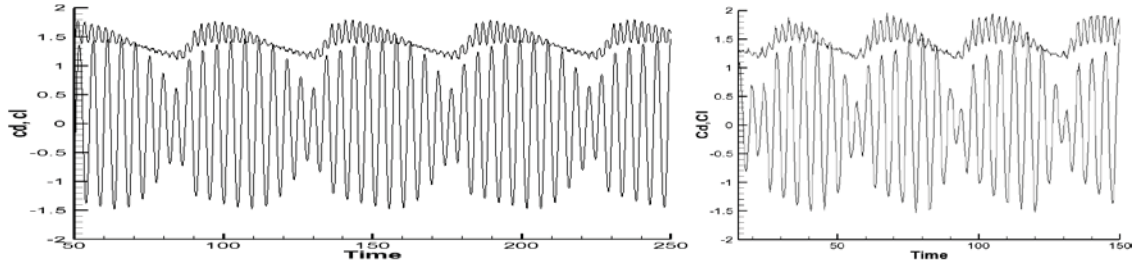
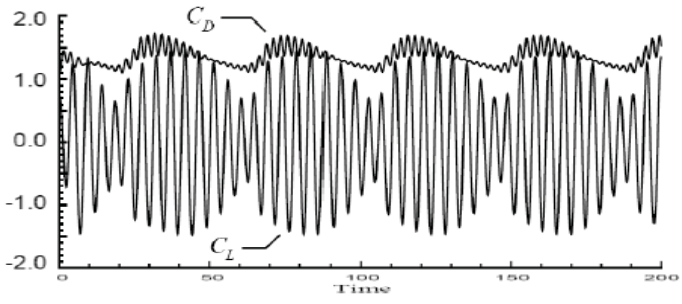


Figure 5.3 Mesh distribution for the flow around an oscillating circular cylinder

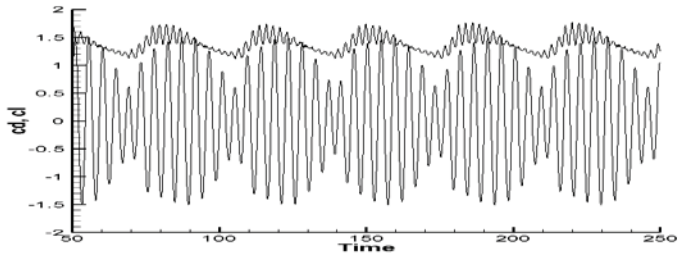


(a) LDFD-IBM (b) LDFD

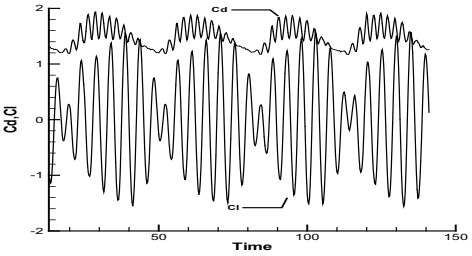


(c) Guilmineau & Queutey (2002)

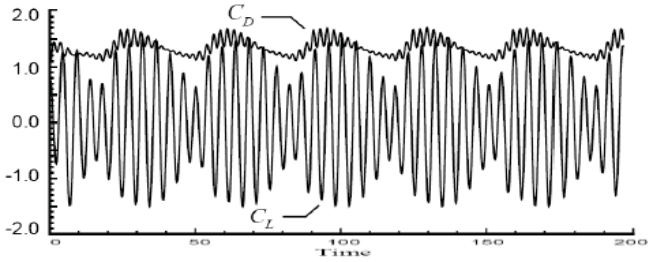
Figure 5.4 Drag and lift coefficients vs time for Re=185 and $A_e / D = 0.2$ for f_e / f_o=1.10



(a) LDFD-IBM

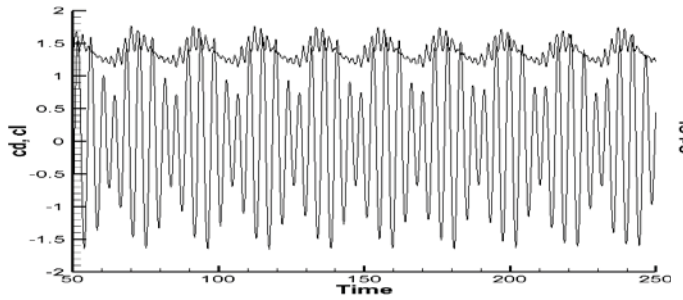


(b) LDFD

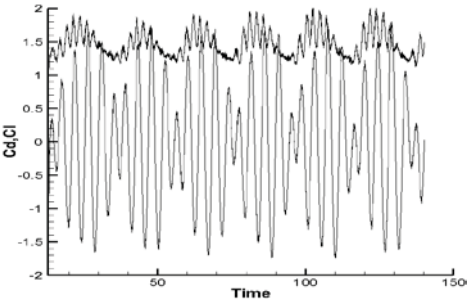


(c) Guilmineau & Queutey (2002)

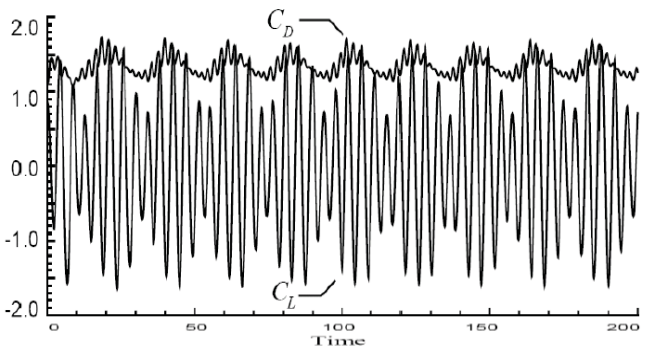
Figure 5.5 Drag and lift coefficients vs time for Re=185 and $A_e / D = 0.2$ for f_e / f_o=1.12



(a) LDFD-IBM

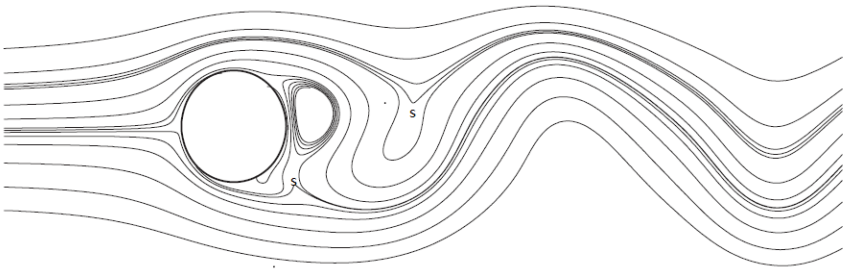


(b) LDFD

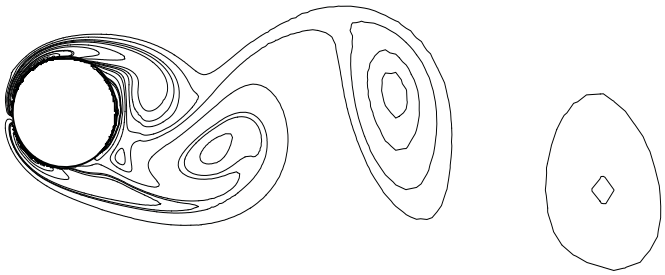


(c) Guilmineau & Queutey (2002)

Figure 5.6 Drag and lift coefficients vs time for Re=185 and $A_e / D = 0.2$ for f_e / f_o=1.20



(a) Streamlines (s-saddle point)



(b) Vorticity contours

Figure 5.7 Instantaneous streamlines and vorticity contours for Re=185 and *Ae*/*D*=0.2,*fe*/*fo*=1.10. The location of the cylinder is at its extreme upper position.

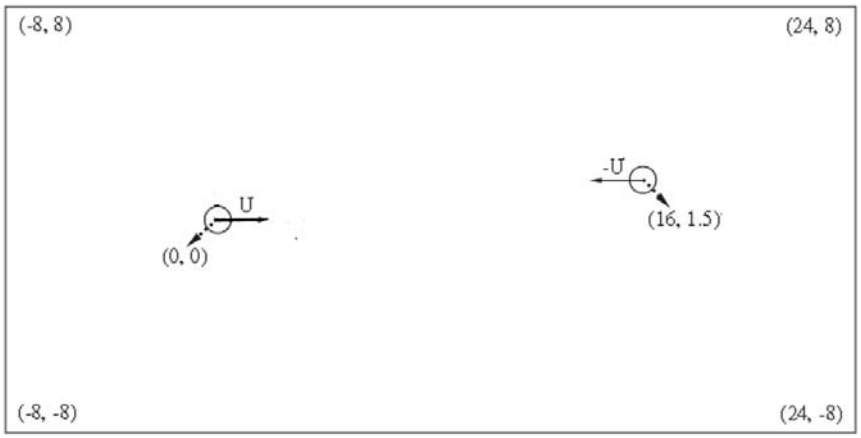
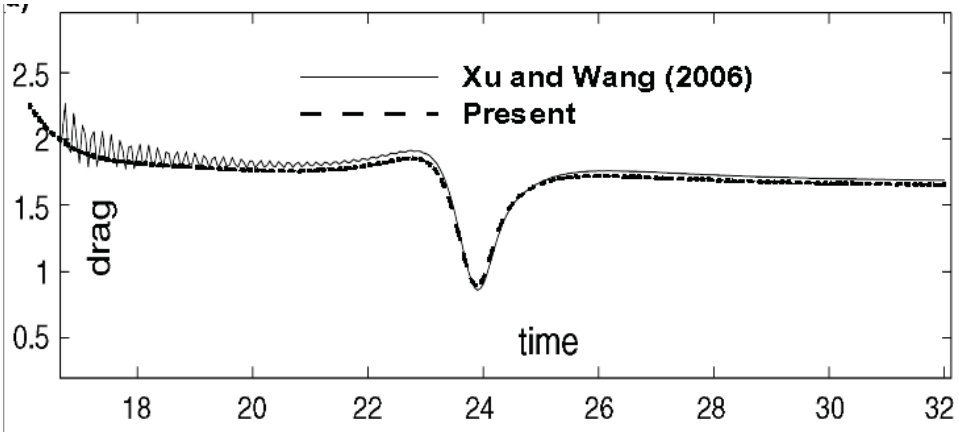
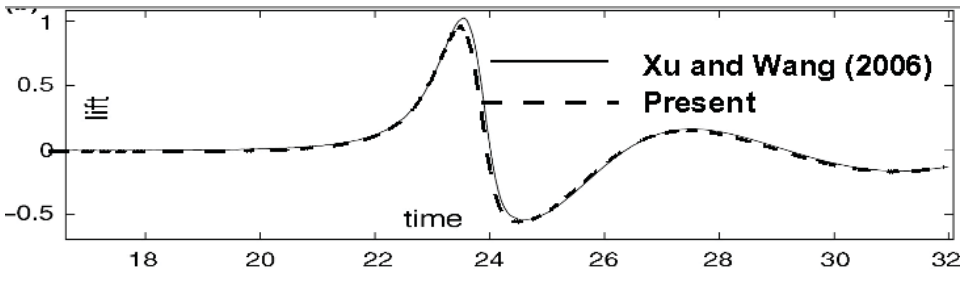


Figure 5.8 Geometry for flow past two cylinders moving with respect to each other.

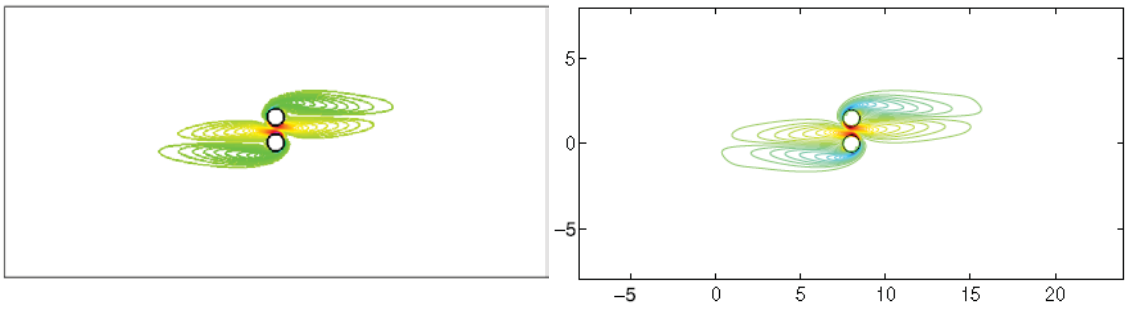


a) C_D number



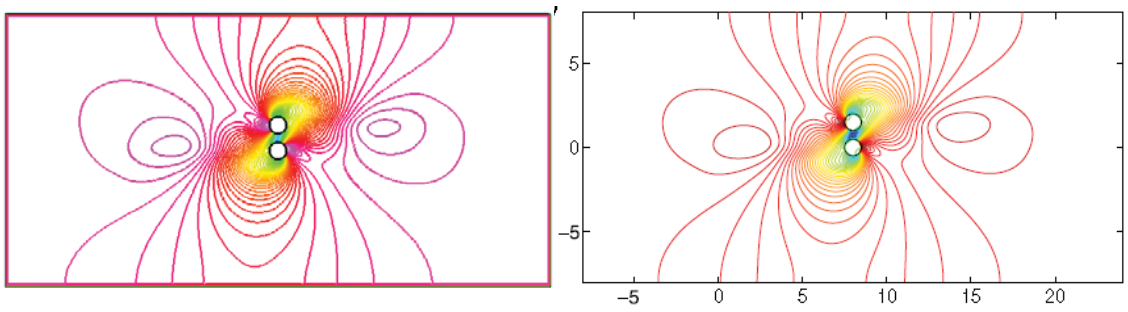
b) C_L number

Figure 5.9 Comparison of C_D and C_L with Xu and Wang (2006)



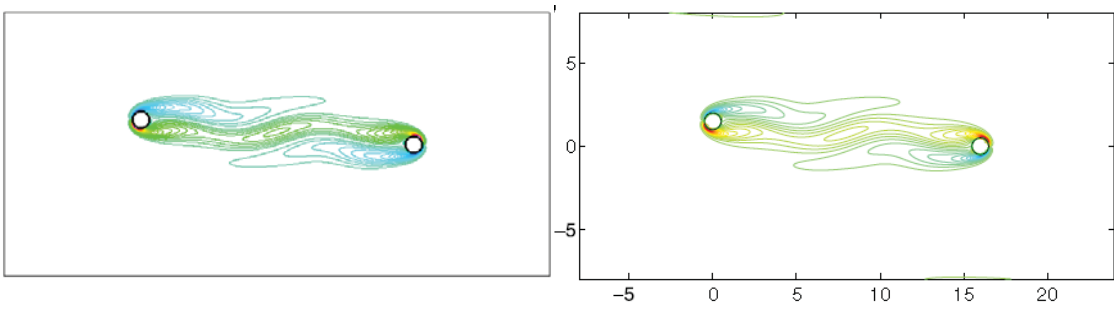
(a) present results (b) results of Xu & Wang (2006)

Figure 5.10 Vorticity contours when two cylinders are closest.



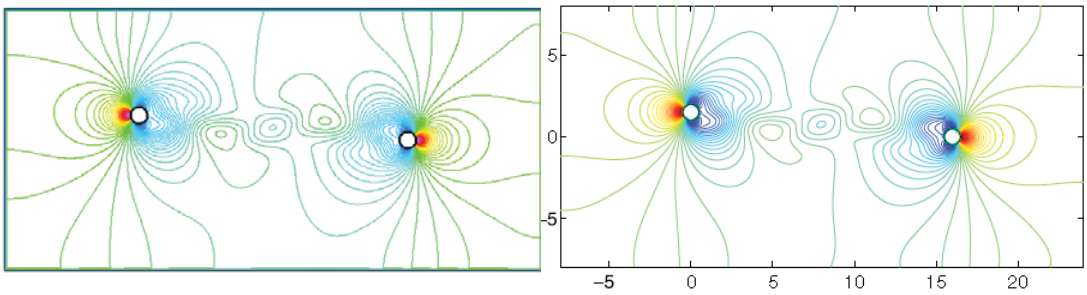
(a) present results (b) results of Xu & Wang (2006)

Figure 5.11 Pressure contours when two cylinders are closest.



(a) present results (b) results of Xu & Wang (2006)

Figure 5.12 Vorticity contours when two cylinders are separated by a distance of 16.



(a) present results (b) results of Xu & Wang (2006)

Figure 5.13 Pressure contours when two cylinders are separated by a distance of 16.

CHAPTER 6

Extension of LDFD-IBM to Simulate Three-dimensional Flows with Complex Boundary

The extension of LDFD and LDFD-IBM to simulation of three-dimensional flows is quite straightforward. Due to its easy implementation, the LDFD-IBM is selected to study three-dimensional flows with curved boundaries. The technical issues related to the three-dimensional applications are discussed in details, particularly for the surface description, the algorithm on fast identification of node status. To avoid the expensive computation on a uniform Cartesian grid, two mesh strategies are adopted, i.e., the non-uniform mesh and the combination of 2D refined mesh and 1D uniform mesh. Numerical tests show that the LDFD-IBM can accurately solve three-dimensional problems with ease.

6.1 The computational procedure for three-dimensional simulation

The LDFD-IBM proposed in Chapter 4 can be easily extended to three-dimensional simulation of flows with complex geometry. The numerical discretization and the treatment of boundary condition are almost the same as the two-dimensional case in Chapter 4, except that one additional coordinate Z needs to be considered.

The basic solution procedure of the LDFD-IBM for three-dimensional flows with stationary object can be outlined as follows:

(1) Set up the computational domain and generate the mesh;

(2) Identify the status of mesh nodes;

(3) Initialize the velocity and pressure fields;

(4) Discretize the spatial derivatives in the governing equations at every node by finite difference scheme, and the time derivative by explicit three-step formulation as described in Section 2.5.2.2 of Chapter 2;

(5) Interpolate the velocity at the interior dependent nodes using the Lagrange interpolation scheme, and enforce the solid body velocity at the exterior dependent nodes;

(6) Solve the resultant algebraic equations;

(7) Repeat (4)~(6) until the convergence criteria is met.

6.2 Mesh strategies for 3D cases

Currently, the 3D stencil AMR technique is still not available. To avoid using the 3D uniform Cartesian mesh, which is computationally very expensive, two meshing strategies are used in the present study. One is to use the non-uniform Cartesian mesh, in which the dense mesh is placed in the region of interest, e.g., the vicinity of the solid object. The other is to use the stencil adaptive refinement in two dimensions and a uniform mesh in the third dimension.

6.2.1 Non-uniform mesh

The non-uniform mesh for the computation is generated in such a way that the fine uniform mesh is placed around the solid body, and the stretched mesh is placed in the

other parts of the domain. This meshing strategy is adopted in the simulations of 3D flows past a sphere and 3D flows past a torus. A typical example of the generated non-uniform mesh is shown in Figure 6.1.

To approximate the derivatives in the governing equations on a non-uniform mesh, various numerical schemes can be chosen. Generally, the accuracy of these schemes depends on how many mesh nodes are used. In this work, we use two schemes: 3-points and 4-points, and the details are shown below.

3-points scheme

The stencil for the 3-points scheme is shown in Figure 6.2, in which symbol P represents the mesh node where the governing equations are discretized, and *W* represents its west neighbor and *E* represents its east neighbor, and the distances from *P* to its neighbors are Δ_1 (to point *W*) and Δ_2 (to point *E*), respectively. Using the Taylor series expansion truncated to the third order derivative term, we have:

$$\left.\frac{\partial f}{\partial x}\right|_P = \frac{1}{\Delta_1\Delta_2(\Delta_1+\Delta_2)}\left[(f_E - f_P)\Delta_1{}^2 - (f_W - f_P)\Delta_2{}^2\right] \tag{6.1a}$$

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_P = \frac{2}{\Delta_1\Delta_2(\Delta_1+\Delta_2)}\left[(f_E - f_P)\Delta_1 + (f_W - f_P)\Delta_2\right] \tag{6.1b}$$

It can be easily found that the 3-points scheme has second-order accuracy for the first-order derivative approximation, and first-order accuracy for the second-order derivative approximation. Eq. (6.1) automatically recovers the central difference schemes when $\Delta_1=\Delta_2$.

4-points scheme

Our goal is to maintain an overall second-order accuracy for spatial discretization. Obviously the accuracy of the 3-points scheme in Eq.(6.1b) is not sufficient for the approximation of second-order derivatives. In principle, to achieve second order accuracy when approximating the second-order derivatives on a non-uniformly distributed stencil, one needs to use at least 4 points. Therefore, two 4-points schemes are developed for the second-order derivatives at the point P, based on the specific stencils:

Backward 4-points scheme:

There are two points on the west of P and one point on the east of P, as shown in Figure 6.3a. The corresponding approximation can be written as

$$\frac{\partial^2 f}{\partial x^2} = 2.\frac{a_0 b_2 + b_0 a_2}{a_2 b_1 + b_2 a_1} \tag{6.2a}$$

where

$$\begin{aligned} a_0 &= \Delta_1(u_{W1} - u_P) - \Delta_3(u_W - u_P) \\ a_1 &= \Delta_1\Delta_3^2 - \Delta_3\Delta_1^2 \\ a_2 &= \Delta_1\Delta_3^3 - \Delta_1^3\Delta_3 \\ b_0 &= \Delta_1(u_E - u_P) + \Delta_2(u_W - u_P) \\ b_1 &= \Delta_1^2\Delta_2 + \Delta_2^2\Delta_1 \\ b_2 &= \Delta_2^3\Delta_1 - \Delta_1^3\Delta_2 \end{aligned} \tag{6.2b}$$

Forward 4-points scheme:

There are two points on the east of P and one point on the west of P, as shown in Figure 6.3b. The corresponding approximation scheme can be written as

$$\frac{\partial^2 f}{\partial x^2} = 2.\frac{a_0 b_2 - b_0 a_2}{b_2 a_1 - a_2 b_1} \tag{6.3a}$$

where

$$\begin{aligned} a_0 &= \Delta_2(u_{E1} - u_P) - \Delta_4(u_E - u_P) \\ a_1 &= \Delta_2\Delta_4^2 - \Delta_4\Delta_2^2 \\ a_2 &= \Delta_2\Delta_4^3 - \Delta_2^3\Delta_4 \\ b_0 &= \Delta_1(u_E - u_P) + \Delta_2(u_W - u_P) \\ b_1 &= \Delta_1^2\Delta_2 + \Delta_2^2\Delta_1 \\ b_2 &= \Delta_2^3\Delta_1 - \Delta_1^3\Delta_2 \end{aligned} \tag{6.3b}$$

For the interior mesh nodes which are not adjacent to the boundary, either Eq. (6.2) or Eq. (6.3) can be used. However, the discretization scheme for the nodes near the boundary must be chosen carefully. We choose the forward 4-points scheme for the mesh nodes next to the left boundary and the backward 4-points scheme for the mesh nodes next to the right boundary.

Since we use the 3-points scheme for the first-order derivative approximation, and the 4-points scheme for the second-order derivative approximation, the overall accuracy for the spatial discretization is maintained to be of second order.

6.2.2 Combination of stencil adaptive refinement and one-dimensional uniform mesh

This strategy is to use the two-dimensional stencil AMR technique to refine the mesh in the X-Y plane, and one-dimensional uniform mesh in the Z-direction. It is suitable for the simulation of three-dimensional flows with a dominant stream-wise direction (i.e. there exists much smaller velocity gradients in one direction than the other two).

Examples are the three-dimensional flows past a 3D circular cylinder.

6.3 Identification of node status in three-dimensions

Identification of node status is a preliminary to the implementation of LDFD-IBM. To identify the node status, we need to know the geometry of the solid surface, which can be defined in different ways.

6.3.1 Surface description

The surface description generally falls into two categories: analytical and numerical descriptions. In the analytical description, the geometry of the three-dimensional body can be explicitly described by mathematical functions. For example, a sphere centered at point O(x_0, y_0, z_0) with radius of the sphere R is determined by the solution of the equation,

$$(x-x_0)^2+(y-y_0)^2+(z-z_0)^2=R^2 \tag{6.4}$$

as shown in Figure 6.4. Analytical description can provide an exact representation of three-dimensional bodies. However, it is usually not available in practice for complex geometries.

Therefore, we often resort to the numerical description, or surface mesh. One of the widely used surface mesh in engineering applications is the triangular mesh, and one example is shown in Figure 6.5. It should be indicated that the surface mesh is an approximate representation of a solid surface, and the accuracy of this approximation

could influence the accuracy of the flow simulation. For example, if a very coarse surface mesh is used to represent a sphere, the simulation of the surrounding flows effectively changes into the simulation of flows over a polyhedron, which means that the physical problem is totally changed. This can be avoided by using a fine surface mesh. While a coarse surface mesh could damage the accuracy of the simulation, an extremely fine surface mesh would increase the computational cost unnecessarily. Therefore, we have to make a balanced choice: to construct a surface mesh at acceptable cost which on the other hand is fine enough to capture all the features of the three-dimensional flows properly.

6.3.2 Fast algorithm of identifying status of mesh nodes

We extend the idea of two-dimensional fast algorithm for identifying the status of mesh nodes in Chapter 2 to the three-dimensional case. Briefly, the procedure is:

(1) For each surface element, determine its maximum and minimum coordinates in x, y, z direction, i.e., *xmin*, *xmax*, *ymin*, *ymax*, *zmin*, *zmax*;

(2) If an uniform Cartesian mesh is used in the sub-zone Ω (coordinates starting from $\Omega_{x\min}, \Omega_{y\min}, \Omega_{z\min}$) which contains the immersed body, then divide (*xmin*- $\Omega_{x\min}$), (*xmax*- $\Omega_{x\min}$), (*ymin*- $\Omega_{y\min}$), (*ymax*- $\Omega_{y\min}$), (*zmin*- $\Omega_{z\min}$), (*zmax*- $\Omega_{z\min}$) by mesh spacing *dx*, *dy*, *dz,* separately. The calculated integer number will be used to locate the element in the uniform Cartesian mesh system in the sub-zone quickly.

(3) For the Cartesian mesh cell containing the surface element, check whether there is an intersection point between the node and its west, east, south, north, top, bottom neighbors, respectively. If yes, this node and its neighbor are the dependent points pair, i.e., one of them is the exterior dependent point, and the other is the interior dependent point.

(4) To determine which one is interior dependent point and which one is exterior dependent point, substitute the coordinates of dependent points into the surface equation determined by the triangular element where the intersection point is located. If the obtained value is negative, then the mesh node is the exterior dependent point.

(5) Once all the exterior dependent points are identified, they serve as a shell along with solid boundary, and the mesh nodes inside the shell are marked as exterior nodes by seed filling algorithm from Computer Graphics (Foley J.D.,1996).

Location of surface element

If a non-uniform mesh is used around the immersed body, step (2) in the above procedure of fast algorithm has to be revised, because *dx*, *dy*, *dz* are different for each mesh node. One has to compare the coordinates of mesh node (*x*,*y*,*z*) with *xmin*, *xmax*, *ymin*, *ymax*, *zmin*, *zmax* to identify which mesh cell contains the surface element.

If an AMR refined mesh is adopted in the computational domain, firstly we need to identify the sub-zone of the mesh that contains the immersed object. Usually this

sub-zone of mesh has the finest resolution and has normal Cartesian coordinate stencil (i.e., Type I in Chapter 3, as shown in Figure 3.1) , then we can follow the same procedure as step (2) above.

Intersection point along mesh line

In the three-dimensional case, to find the intersection point between a mesh line and a surface element, one must firstly find the plane equation of the surface element. The standard equation of a plane in 3D space is

$$Ax + By + Cz + D = 0 \tag{6.5}$$

For the triangular element with the coordinates of three given points ($x1,y1,z1$), ($x2,y2,z2$), ($x3,y3,z3$), the coefficients in the plane equation (6.5) are given by the following determinants:

$$A = \begin{vmatrix} 1 & y1 & z1 \\ 1 & y2 & z2 \\ 1 & y3 & z3 \end{vmatrix} \quad B = \begin{vmatrix} x1 & 1 & z1 \\ x2 & 1 & z2 \\ x3 & 1 & z3 \end{vmatrix} \quad C = \begin{vmatrix} x1 & y1 & 1 \\ x2 & y2 & 1 \\ x3 & y3 & 1 \end{vmatrix} \quad D = \begin{vmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{vmatrix} \tag{6.6}$$

As shown in Figure 6.6, to find the intersection point of a plane and a mesh line between two mesh nodes **P1** and **P2** along y direction, firstly we need to find the equation of line through points **P1** (x1,y1,z1) and **P2** (x2,y2,z2) which gives

$$\mathbf{P} = \mathbf{P1} + t\,(\mathbf{P2} - \mathbf{P1}) \tag{6.7}$$

where P(x,y,z) is an arbitrary point along the line. Since the intersection point is on the plane, substituting the coordinates of point ***P*** into the equation of the plane gives

$$A\,(x1 + t\,(x2 - x1)) + B\,(y1 + t\,(y2 - y1)) + C\,(z1 + t\,(z2 - z1)) + D = 0 \tag{6.8}$$

which further gives

$$t = \frac{A * x1 + B * y1 + C * z1 + D}{A * (x1 - x2) + B * (y1 - y2) + C * (z1 - z2)} \tag{6.9}$$

If *t* is between 0 and 1, then there is an intersection point between **P1** and **P2**.

This intersection point is on the plane determined by the triangle ΔQ1Q2Q3, as shown in Figure 6.6. Naturally, a question arises, that is, is the intersection point inside of the triangular element? Take Figure 6.7a as an example. Suppose that, the surface element is ΔABC with the coordinates of three vertices (x_A,y_A,z_A), (x_B,y_B,z_B) (x_C,y_C,z_C). Its maximum and minimum coordinates in *x*, *y*, *z* direction are *xmin*, *xmax*, *ymin*, *ymax*, *zmin*, *zmax*; the intersection point is $D(x_D,y_D,z_D)$ and has $x\min \le x_D \le x\max$, $y\min \le y_D \le y\max$, $z\min \le z_D \le z\max$. It is found that the point D can still possibly be outside of the surface element ΔABC. Hence, some intersection points may not be the truly intersection points.

Therefore, we have to determine if a point is truly located in the triangle. As shown in Figure 6.7b, for a triangle ΔABC in 3D space, if the sum of area of sub-triangles ΔDAB,ΔDBC,ΔDCA equals to the area of triangle ΔABC, then point D is in ΔABC. To calculate the area of the triangle in 3D space, take ΔABC as an example, we have to find the vector AB, AC:

$$\begin{aligned} \overrightarrow{AB} &= (x_B - x_A, y_B - y_A, z_B - z_A) = (a_1, a_2, a_3), \\ \overrightarrow{AC} &= (x_C - x_A, y_C - y_A, z_C - z_A) = (b_1, b_2, b_3) \end{aligned} \tag{6.10}$$

And their cross (vector) product is

$$\overrightarrow{AB} \times \overrightarrow{AC} = (a_2 b_3 - a_3 b_2, a_3 b_1 - a_1 b_3, a_1 b_2 - a_2 b_1) \tag{6.11}$$

The area of ΔABC is the half of the magnitude of resultant vector $\overrightarrow{AB} \times \overrightarrow{AC}$, i.e.:

$$S_{\Delta ABC} = \frac{1}{2}\sqrt{(a_2 b_3 - a_3 b_2)^2 + (a_3 b_1 - a_1 b_3)^2 + (a_1 b_2 - a_2 b_1)^2} \tag{6.12}$$

The areas of triangles ΔDAB,ΔDBC,ΔDCA can be computed in the smae way. If

$$S_{\Delta DAB} + S_{\Delta DBC} + S_{\Delta DCA} = S_{\Delta ABC} \tag{6.13}$$

Then point D is in the triangle ΔABC, which means the intersection point of the triangle element and mesh line is found.

Once the intersection point is found, we can easily determine the pair of exterior dependent point and interior dependent point around the boundary surface. Suppose that **P0** (x0,y0,z0) is one of dependent points, to determine if it is the interior dependent or the exterior dependent point, one can substitute the coordinate of the mesh node into the plane equation of the surface element Eq.(6.5), the value is

$$S_{p0} = Ax0 + By0 + Cz0 + D \tag{6.14}$$

If $S_{p0}>0$, it means that **P0** is on the same direction of the normal outward of the surface element. Therefore, **P0** could be the interior dependent point if the index sequence of the surface elements is arranged in such a way that the normal outward direction of the solid body is directed into the flow domain (most CAD software give this kind of index sequence). The other end node of this dependent points pair is then identified as the exterior dependent point.

6.4 Application to three-dimensional flows around stationary boundaries

6.4.1 Force calculation

In the simulations of the flow around a 3D blunt body, lift and drag coefficients C_L

and C_D are useful flow parameters to quantitatively validate the accuracy of numerical results. The definitions of C_L and C_D can be commonly written as

$$C_D = \frac{\mathbf{F}_D}{\frac{1}{2}\rho U^2 A}, \quad C_L = \frac{\mathbf{F}_L}{\frac{1}{2}\rho U^2 A} \tag{6.15}$$

where ρ is the fluid density; U is the reference velocity and A is the reference area, whose definitions may vary for different cases. In the following 3D simulations, the drag force F_D and the lift force F_L on the immersed body are calculated from the integral of the x- and y-component of the momentum equations over a rectangular control volume Ω enclosing the blunt body as follows :

$$F_D = -\left\{ \frac{d}{dt}\left(\int_\Omega u_1 d\Omega \right) + \int_{\partial\Omega} \left[u_1 \mathbf{u}\cdot\mathbf{n} + p n_1 - \frac{1}{\mathrm{Re}}\left(\frac{\partial u_1}{\partial x_i} + \frac{\partial u_i}{\partial x_1} \right) \right] ds \right\} \tag{6.16}$$

$$F_L = -\left\{ \frac{d}{dt}\left(\int_\Omega u_2 d\Omega \right) + \int_{\partial\Omega} \left[u_2 \mathbf{u}\cdot\mathbf{n} + p n_2 - \frac{1}{\mathrm{Re}}\left(\frac{\partial u_2}{\partial x_i} + \frac{\partial u_i}{\partial x_2} \right) \right] ds \right\} \tag{6.17}$$

where $\partial\Omega$ is the control surface of Ω, $\mathbf{n} = (n_1, n_2)$ is the normal vector to the boundary of the control surface, the subscripts 1 and 2 denote the x-direction and y-direction, respectively.

6.4.2 Numerical validation of flows around a stationary sphere

The 3D solver is firstly validated by simulating the flow over a sphere. The Reynolds number is defined as $\mathrm{Re} = \frac{U_\infty D}{\nu}$. Here U_∞ is the free stream velocity and D is the sphere diameter, and ν is the kinematic viscosity. We consider here the flows with intermediate Reynolds number ($\mathrm{Re} < 300$).

The size of the computational domain is 30D×20D×20D in the x-, y- and z-direction, respectively. The sphere is located at (10D,10D,10D), covered by a surface mesh of 2896 triangular elements. The non-uniform mesh for the flow past a sphere is shown in Figure 6.1, where the uniform mesh is placed around the sphere, and the stretched meshes are applied in the rest of the domain. We use three different meshes in this study:

- Mesh Type 1: 121×101×101 for Re=50 and 100, and the uniform mesh spacing around the sphere is Δ=0.03D;
- Mesh Type 2: 141×121×121 for Re=150 and 200, and the uniform mesh spacing around the sphere is Δ=0.025D;
- Mesh Type 3: 201×181×181 for Re=250, and the uniform mesh spacing around the sphere is Δ=0.0125D.

At the inlet, the velocity is set to the free stream velocity U_∞. Zero-gradient condition is imposed at far field boundaries of computational domain, and non-slip boundary condition is applied on the sphere surface.

Figure 6.8 shows the streamlines on the x-y plane for Re=50, 100, 150, 200, respectively. The numerical results show that the wake behind the sphere is steady and axi-symmetric up to $\mathrm{Re} = 200$, in agreement with previous experimental observations. In this range of Re, both the separation angle and the length of recirculation region (L_s) increase with Re, as shown in Figure 6.9, in which the

results of Johnson and Patel (1999) and Gilmanov et al (2003) are also included for comparison. Good agreement can be seen between our results and theirs.

Figure 6.10 shows the streamlines on the *x*-*y* and *x*-*z* planes at Re=250. It is found that the flow on the x-y plane is still symmetric, but the symmetry is lost on the x-z plane. Hence, the flow is not axi-symmetric, but planar symmetric. These results are consistent with previous findings (Johnson and Patel, 1999; Gilmanov et al. 2003).

Table 6.1 compares the drag coefficients with previous numerical (Johnson and Patel, 1999; Gilmanov et al, 2003; Wu and Shu, 2010) and experimental data (White, 1974). Here, the drag coefficient C_D is defined as

$$C_D = \frac{F_D}{\frac{1}{2}\rho U_\infty^2 S} = \frac{F_D}{\frac{1}{2}\rho U_\infty^2 \pi R^2} \tag{6.18}$$

where $\mathbf{F}_D$ is the drag force determined by Eq.(6.16).

From Table 6.1, it is found that the present results agree well with the data from the literature. This validation indicates that our LDFD-IBM 3D solver can accurately simulate 3D flows with irregular boundary.

6.4.3 Numerical simulation of 3D flow past a torus with small aspect ratio

The shapes like sphere and cylinder can be considered as special cases of the torus (or ring). A single geometric parameter, i.e., aspect ratio (*Ar*), can be used to describe the

bluff torus geometry. The aspect ratio (*Ar*) is defined as $Ar = D/d$, where D is the mean torus diameter and d is the cross-section diameter, as shown in Figure 6.11. By varying *Ar*, the uniform axi-symmetric body changes from sphere at $Ar = 0$, to cylinder in the limit $Ar \to \infty$.

The flow over a torus with a small aspect ratio Ar=2 is considered here. The axis of the torus is placed parallel to the flow. Since there is a hole in the center of torus, the flow can go through the torus from its central portion, which is significantly different from the flow over a sphere. Therefore, it is expected that the wake structure behind the torus with $Ar = 2$ has different flow patterns too.

In the present simulation, Re=40 is considered. Here, the Reynolds number is based on the free stream velocity U_∞ and the cross-section diameter d. The initial conditions and outer boundary conditions are implemented through the same way as used in the sphere case.

Similar to the sphere case, the computational domain is taken as 30D×20D×20D in x,y,z direction, respectively. The torus is located at (10D, 10D, 10D), which is covered by a surface mesh of 4436 triangular elements. A non-uniform mesh with mesh size of 121×131×131, which is fine and uniform around the torus with Δ=0.05D, is taken.

Figure 6.12 shows the streamlines. It is found from the figure that the flow at Re=40 is axi-symmetric. The existence of hole in the center of torus makes the recirculation region on the axis detach from the rear part of torus, which is quite different from the case of flow over a sphere. Due to presence of the hole, the stagnation points of torus are located in cross-section surface. Hence, the new recirculation region appears behind the cross-section of torus.

As suggested by Sheard et al. (2005), the drag coefficient of torus is defined as

$$C_D = \frac{F_D}{\frac{1}{2}\rho U_\infty^2 A_{frontal}} \tag{6.19}$$

where $A_{frontal}$ is the projected frontal area of torus. If the dimensions are scaled by the diameter of torus cross-section d, $A_{frontal}$ can be written as a function of Ar

$$A_{frontal} = \begin{cases} \dfrac{4}{\pi(Ar^2 + 2Ar + 1)} & 0 \le Ar \le 1 \\ \dfrac{1}{\pi Ar} & Ar \ge 1 \end{cases} \tag{6.20}$$

The obtained drag coefficients are listed in Table 6.2, which also includes the results from Sheard et al. (2005) and Wu et al (2010). It can be found that the present numerical results compare well with previous data in the literature.

6.4.4 Numerical simulation of 3D flows over a circular cylinder

6.4.4.1 Background

Although the flow past a circular cylinder has been investigated by a large number of researchers, the computational studies are mainly restricted to two-dimensional flows.

On the other hand, according to Karniadakis and Triantafyllou (1989), three-dimensionality is "an unavoidable state of even nominally 2D wakes, once a certain critical Reynolds number is exceeded". It is found that the flow remains two-dimensional only when the Reynolds number is below 170. Beyond that, 2D vortex structures are unstable to 3D disturbances. Thus, it is conventionally thought that the 2D wake of a cylinder experiences a transition to 3D wake at the critical Reynolds number Re=170 (Zhang & Dalton, 1998).

Besides the Reynolds number, various other parameters, for example, surface roughness, free stream turbulence, blockage, end-effects, aspect ratio, have a significant impact on the flow. A comprehensive overview of vortex dynamics in the cylinder wake can be found in the review paper by Williamson (1996) and the recent book by Zdravkovich (1997).

Various researchers in the past have observed slantwise/oblique vortex shedding past a circular cylinder in 3D configuration. This leads to the concept of oblique modes versus parallel modes of vortex shedding according to the angle between the axis of the cylinder and that of the vortices being shed related to the 3D wake. Oblique shedding is one of the important 3D features that could influence the amplitude, frequency and phase of the flow-induced forces.

It is believed that the end-conditions of the cylinder can trigger different modes of

shedding. Slaouti and Gerrard (1981) studied the dependence of the vortex shedding on the end conditions. Their experiments utilized cylinders of aspect ratio (=L/D, ratio of length to diameter) varying between 25 and 30 in a tow tank at Reynolds number ranging from 60 to 200. They reported a strong dependence of three dimensional structure of the wake on the end conditions of the cylinder.

Williamson (1989) related the phenomenon of oblique vortex shedding to the existence of discontinuities in the Strouhal-Reynolds number relationship. He concluded that both the oblique and parallel modes of vortex shedding are intrinsic to the flow past a circular cylinder and are instigated by the end boundary conditions.

Mittal (2001) and Behara & Mittal (2010) did extensive studies on oblique vortex shedding in the uniform flow for a nominally 2D cylinder. They carried out numerical experiments for cylinders with aspect ratio of 16 and 60 at Re=60~1000. The end conditions are specified to model the effect of a wall. They found that the flow for Re<200 is very organized, devoid of any vortex dislocations.

In the following sections, LDFD-IBM 3D solver is applied to simulate the 3D flow over a single cylinder with two different end conditions at Re=100. The Reynolds number is low enough so that the flow is devoid of the complexities associated with turbulence in the wake. The 3D mesh strategy of combing 2D refined mesh and 1D uniform mesh described in Section 6.2.2 is adopted. The numerical results are compared with other researcher's results and good agreement is obtained. The end

effects on the 3D structure of this kind of flows are discussed.

6.4.4.2 Numerical simulation of 3D flow over a cylinder with periodic boundary condition

Firstly, we apply the 3D LDFD-IBM solver to simulate the 3D flow over a circular cylinder with periodic boundary condition at Re=100. At this Reynolds number, the physical flow exhibits clearly 2D features. The Reynolds number is defined as $\mathrm{Re} = \frac{U_\infty D}{\nu}$, where D is the cylinder diameter, and ν is the kinematic viscosity.

The problem configuration is shown in Fig. 6.13, and the size of the domain is 32D×20D×11D. The center of the cylinder is located at (8D×10D） on the x-y plane. To enforce the periodic boundary condition, the length in the z direction of the computational domain is taken to be the spanwise wavelength or its multiple. As recommended by Zhang and Dalton (1998), the length of the cylinder L is taken as 11R (R is the radius of the circular cylinder) or multiple of 11R.

Uniform flow velocity is prescribed at the inlet and no-slip condition is specified on the cylinder wall. At the downstream boundary, a far-field boundary condition is used. On the boundary in the z direction, a periodic condition is applied.

A refined mesh around the cylinder is used in the x-y plane, similar to the 2D simulation. The initial mesh is 161×101. After refinement around the cylinder by 8

levels, final mesh on the x-y plane has 46791 nodes. There are 41 nodes uniformly distributed along the z direction.

Figure 6.14 shows vorticity distribution on the x-z plane. Figure 6.15 illustrates the iso-surfaces of the vorticity in the flow domain. It is clear that the structure of vorticity exhibits two-dimensional feature. The evolution of C_D, C_L with time in the mid-plane is shown in Figure 6.16. Table 6.3 lists the 2D and 3D results obtained by the LDFD-IBM, in comparison with available data in the literature. It is found that the present 2D results and the 3D results are very close, and agree well with the results of previous studies. This confirms that the flows at Re=100 with periodic boundary condition are essentially 2D flows.

6.4.4.3 Numerical simulation for 3D flow over cylinder with two end-walls boundary condition

In this section, the 3D LDFD-IBM solver is applied to simulate the 3D flow over a cylinder with two walls at Re=100. The wall boundary condition is applied on two ends in the z direction. The 3D flows over the cylinder with aspect ratio of L/D=16 are chosen as the test case, in which the 3D flow features are expected.

The computational domain, mesh size and cylinder location remain the same as the previous one. The boundary conditions and numerical discretization are almost the same, except that the no-slip boundary condition is imposed on the two end walls in

the z direction.

For 3D flows past a cylinder with end walls, the simulation is terminated when the lift coefficient C_L reaches a truly "stationary state". According to So et al (2005), because of the phase transition, the simulation needs to be carried out for a long period of dimensionless time (e.g. t>500) such that the variation of C_L reaches a truly ‘stationary state’. In our case, we terminate the computation at t=700 and the stationary state is considered to be reached.

Figure 6.17 shows the evolution of C_D and C_L with time in the mid-plane of the cylinder for present numerical simulation. It is found that the flow reaches a "quasi-stationary state" when 120<t<200. After a transition period, the flow reaches "stationary state" at around t=400. When t>400, no further variation was observed.

Figure 6.18 shows the distribution of vorticity component ω_z in the Y=0 plane at various time instants during the simulation. It is observed that for the time instant of t=140 (which is in quasi-stationary state according to Figure 6.17), the vortex shedding is parallel, while at t=700 (in stationary state) it becomes oblique shedding. This is in line with the observation of Williamson (1989) and So et al (2005). Unlike the 3D flow over a cylinder with periodic boundary condition, where only parallel vortex shedding can be observed, the wake of flow over a cylinder with two end walls presents clearly 3D features and there is no symmetry along the cylinder in the z

direction. Therefore, it is essentially a 3D flow. Table 6.4 lists the obtained C_D, C_L and *St*, as well as the data from the literature. It is clear that good agreement has been achieved.

Our numerical studies confirm that for a circular cylinder with finite length and low Re, the vortex shedding pattern, either oblique or parallel, strongly depends on the boundary condition at both ends of the cylinder. The end wall conditions may provoke the oblique mode of vortex shedding over the cylinder even if the length of cylinder is relatively large (16D) as compared with its counterpart (11D) with periodic boundary condition. The wake is already three-dimensional at Re=100 with end-wall boundary condition, which is far below the critical Reynolds number of 170 with the periodic boundary condition.

6.5 Concluding remarks

In this chapter, the numerical discretization and fast algorithm to determine the status of mesh nodes in the 2D LDFD-IBM solver are extended to the 3D case. Two mesh strategies are adopted to avoid very costly computation on a 3D uniform Cartesian mesh. The 3D LDFD-IBM solver has been used to study the flows past bluff bodies with curved boundaries, such as the torus family ranging from the sphere (Ar=0) and the torus (Ar=2) to the cylinder (Ar→∞). The results successfully predict the flow patterns for a variety of flow conditions. The quantitative agreement of flow parameters with the literature data validates the LDFD-IBM for 3D flow simulations.

All these indicate that the present LDFD-IBM is a promising tool for the numerical study of flows with complex geometry.

Table 6.1. Comparison of drag coefficient C_D for flows over a sphere at Re=100,200,250

References	Re=100	Re=200	Re=250
Johnson et al(1999)	1.112	0.79	0.70
Gilmanov et al(2003)	1.153	-	-
White (1974)	1.18	0.81	-
Wu and Shu (2010)	1.128	0.8	-
Present	1.134	0.81	0.73

Table 6.2. Comparison of drag coefficient C_D for flows over a torus with Ar=2, Re=40

References	C_D at Re=40
Sheard et al(2005)	1.3
Wu and Shu(2010)	1.335
Present	1.39

Table 6.3 Comparison of $\overline{C}_D$, C_L' at Re=100

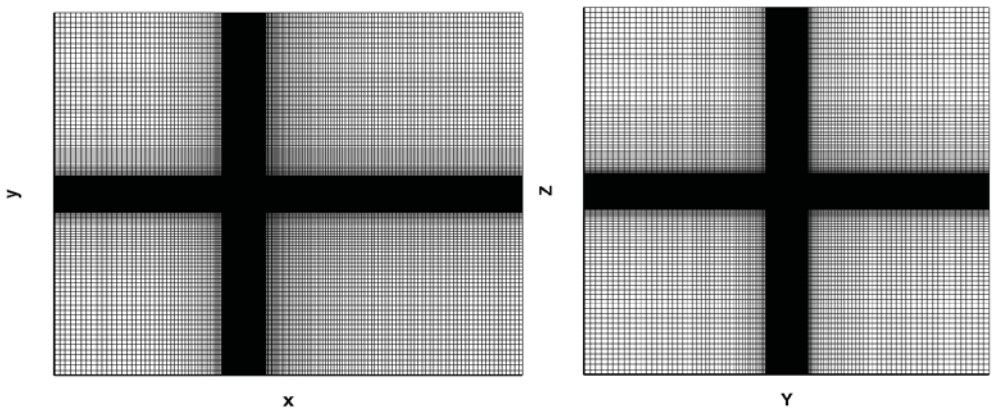
(3D cylinder, cylinder length =11D, periodic boundary condition)

References	$\overline{C}_D$	C_L'	*St*
Lai and Peskin (2000) 2D results	1.4473	±0.323	0.165
Zhang &Dalton (1998) 3D results L/D=11	1.32	±0.23	-
Jordan and Fromm (1972) experiments	1.30	-	0.16-0.17
Present LDFD-IBM 2D results	1.33	±0.34	0.166
Present LDFD-IBM 3D results	1.42	±0.35	0.166

Table 6.4 Numerical and experimental values of $\overline{C}_D$, C_L', *St* at Re=100

(3D cylinder, cylinder length =16D, value at mid-plane and with end walls)

3D results	$\overline{C}_D$	C_L'	*St*
Mittal (2001)	1.34	0.047	0.145
Liu et al (2006)	1.38	0.069	0.145
Tang et al (2003)	1.325	0.075	0.152
Present	1.43	0.071	0.147



(a) X-Y plane (b) Y-Z plane

Figure 6.1 Non-uniform mesh for 3D simulation of the flow past a sphere

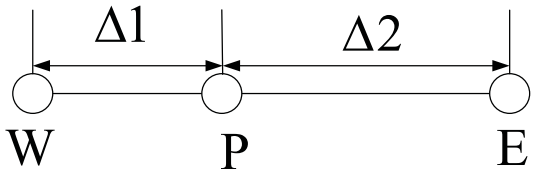
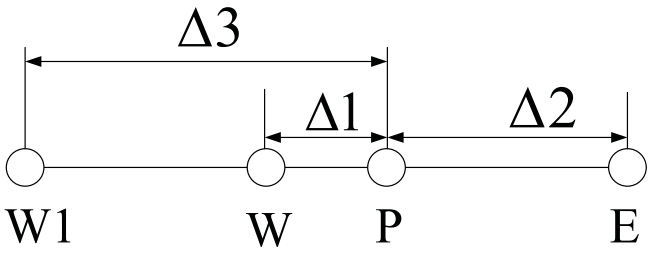
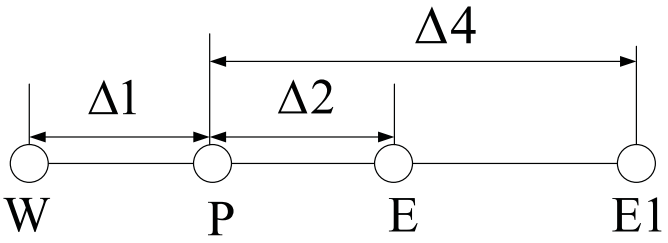


Figure 6.2 Non-uniform mesh: 3-points scheme



(a) backward 4-points scheme



(b) forward 4-points scheme

Figure 6.3 Non-uniform mesh: 4-points scheme

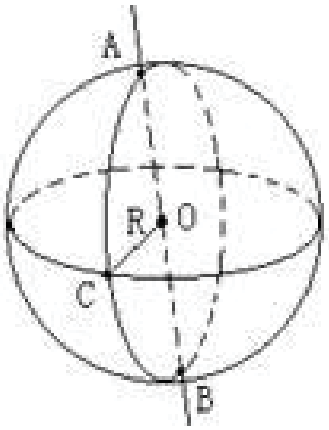


Figure 6.4 Analytical definition of a sphere

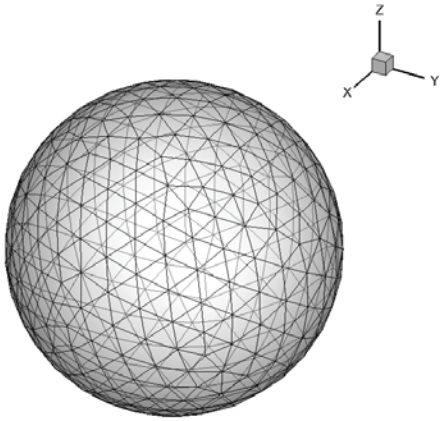


Figure 6.5 Triangular surface mesh covering the surface of the solid body

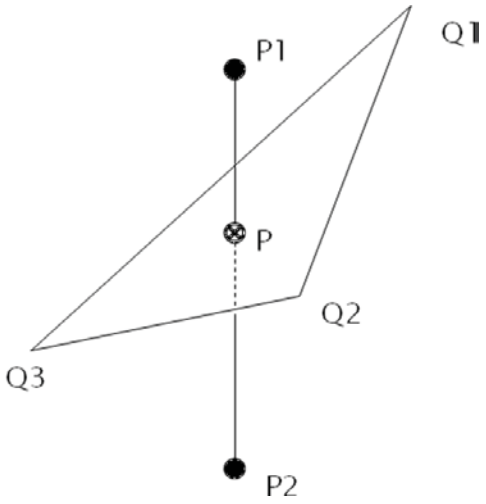
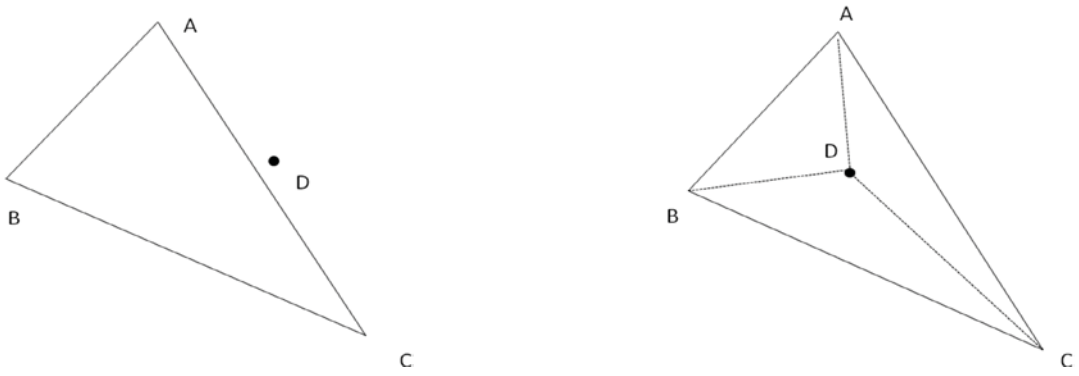
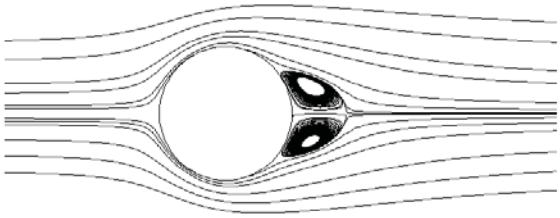


Figure 6.6 Finding the intersection point of a plane with a mesh line

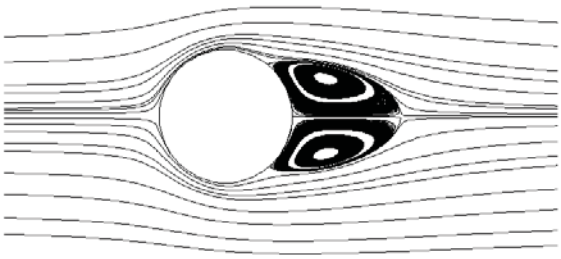


(a) Point D is not inside ΔABC (b) Point D is inside ΔABC

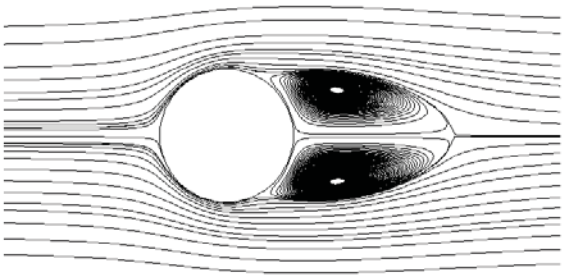
Figure 6.7 Determination of a point being inside a triangular element



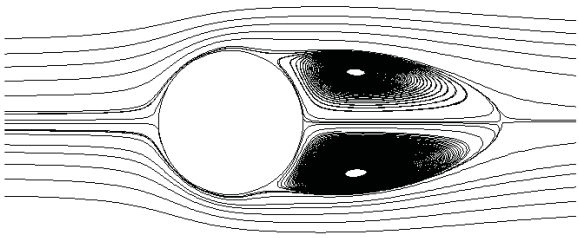
(a) Re=50



(b) Re=100



(c) Re=150



(d) Re=200

Figure 6.8 Streamlines on the x-z plane for flows over a sphere at steady axi-symmetric state

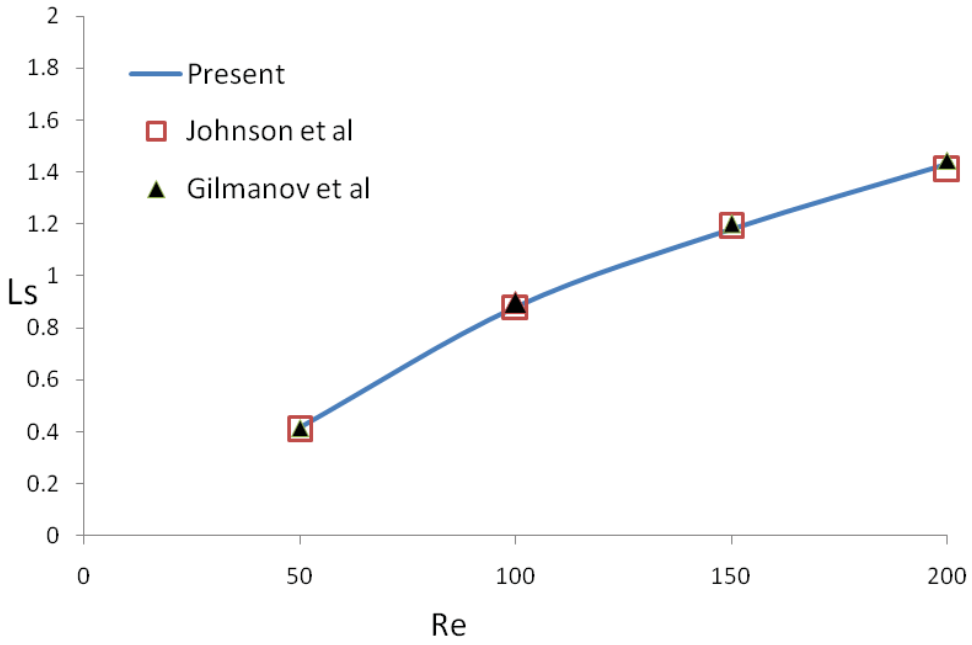
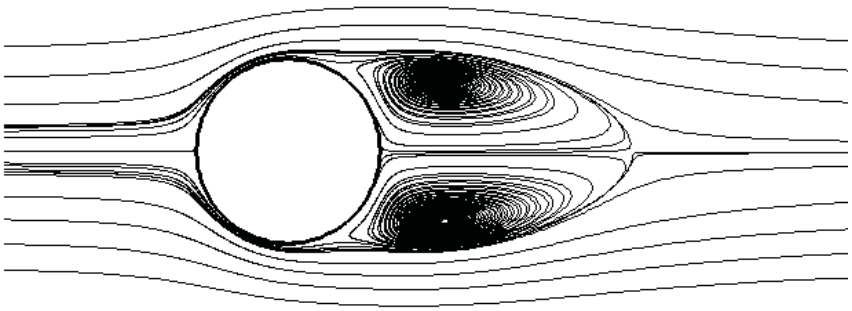
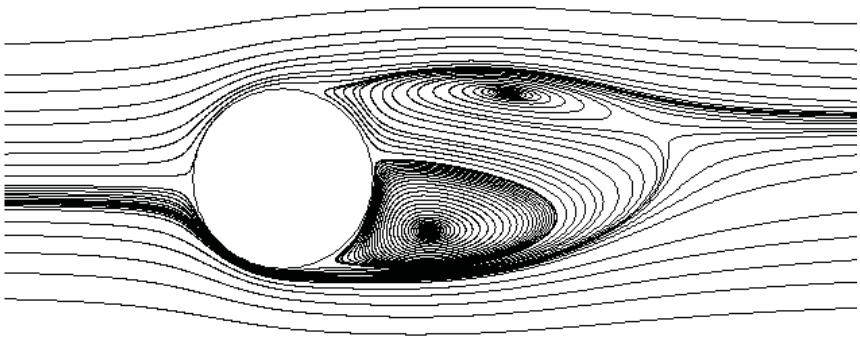


Figure 6.9 Comparison of recirculation length Ls for flow over a sphere at different Re



(a) X-Y plane



(b) X-Z plane

Figure 6.10 Streamlines for flow over a sphere at *Re* = 250 (steady non-axi-symmetric state)

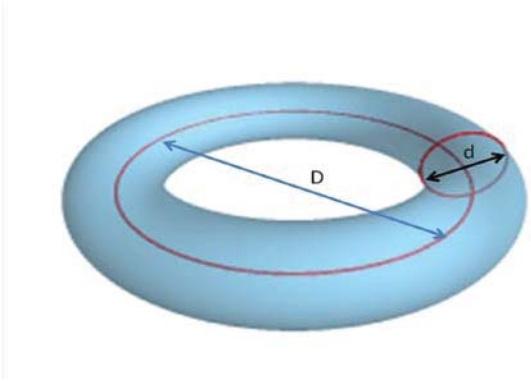


Figure 6.11 Configuration of a torus

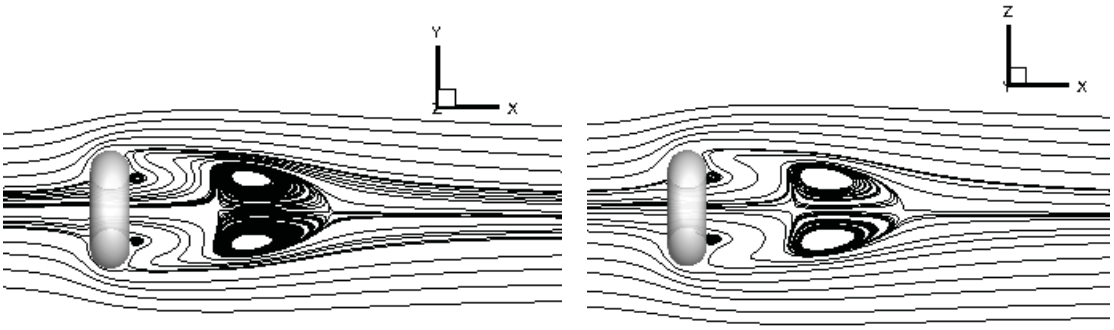


Figure 6.12 Streamlines for flows over a torus with Ar=2, Re=40

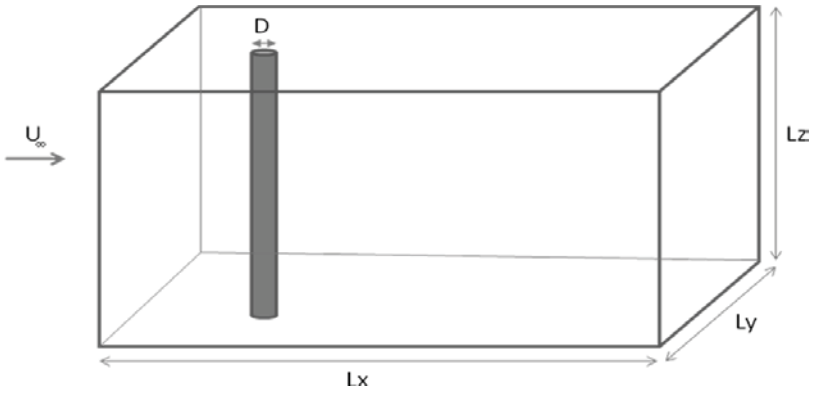


Figure 6.13 Schematic view of the configuration of 3D flow over a cylinder

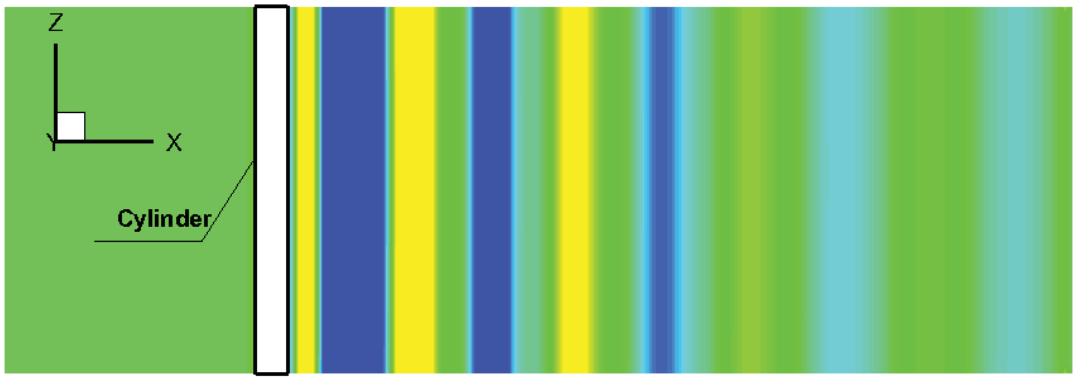


Figure 6.14 The span-wise component of vorticity in the X-Z plane passing through the axis of the cylinder (Re=100, periodic boundary condition)

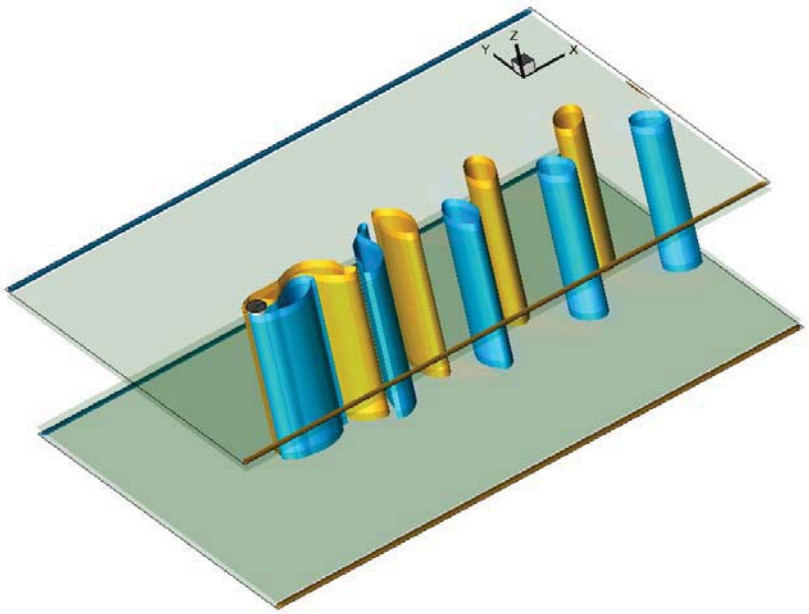


Figure 6.15 The iso-surface of spanwise component of vorticity: flow past a cylinder at Re=100 with periodic boundary condition

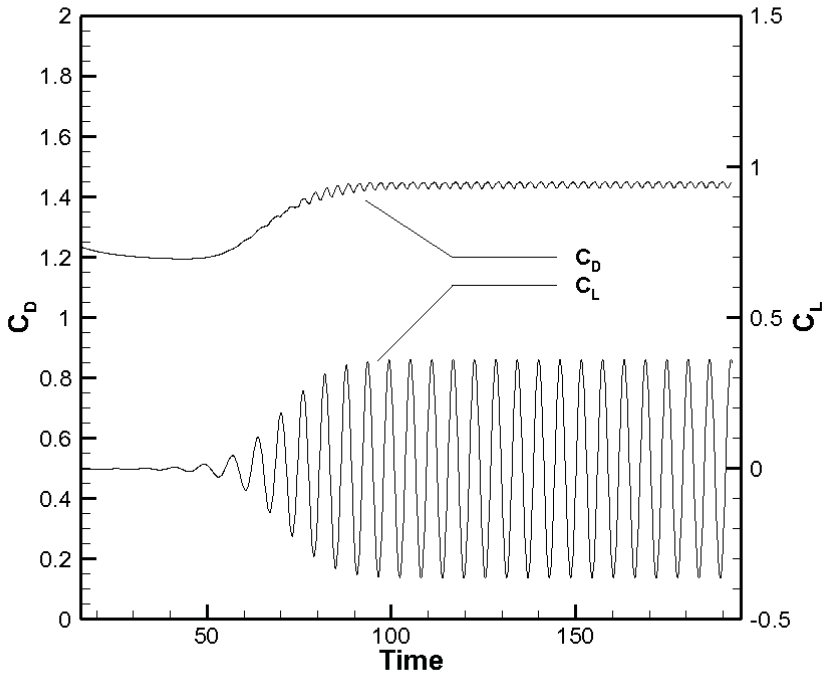


Figure 6.16 Drag and lift coefficients for 3D flow over a cylinder of L/D=11 at Re=100 with periodic boundary condition

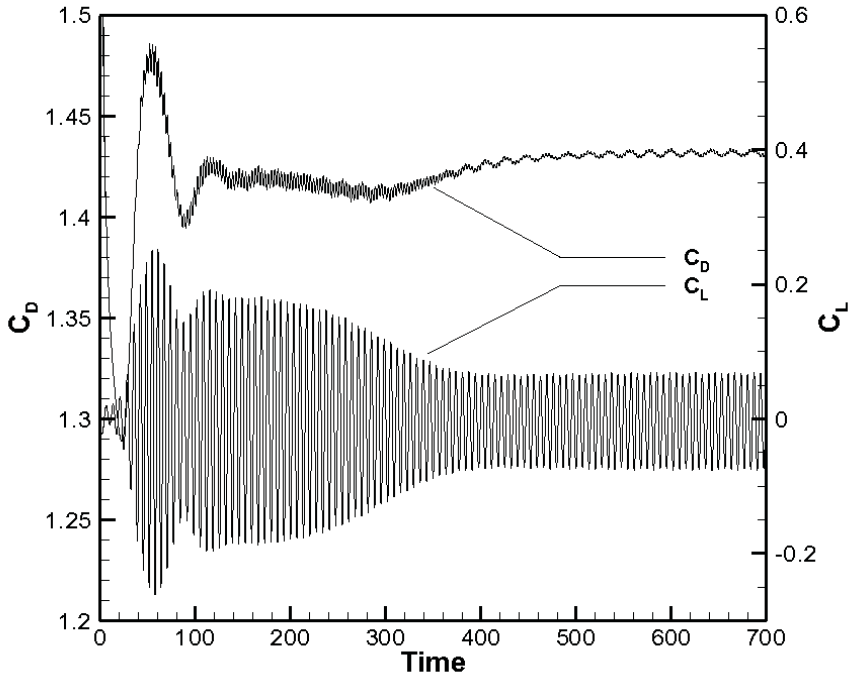


Figure 6.17 Drag and lift coefficients for 3D flow over a cylinder of L/D=16 at Re=100 with two end walls boundary condition

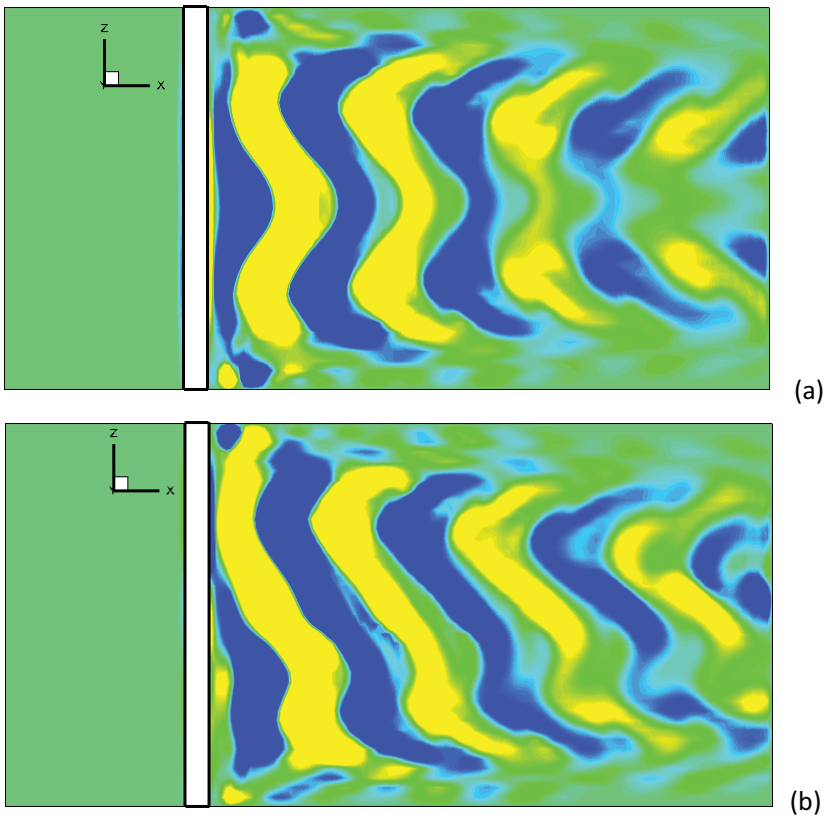


Figure 6.18 Vorticity component ω_z in the Y=0 plane at different time instants:
(a) t=140 in the quasi-stationary state, parallel shedding
(b) t=700 in the stationary state, oblique shedding.

CHAPTER 7

Application of LDFD to Simulate Compressible Inviscid Flows

In this chapter, we apply the LDFD method to simulate compressible inviscid flows, which are governed by Euler equations. As compared to viscous flows, the no-slip condition at the surface of the solid body is replaced by no-penetration boundary condition in compressible inviscid flows, which poses a new challenge for LDFD method. Since the conservation on the discrete level is highly desirable for Euler solvers, the LDFD in the framework of finite volume is presented. The method is applied to a variety of applications, such as inviscid compressible flow past a cylinder, the flow with strong shock in a wedge channel and the flow past a NACA0012 airfoil. Numerical results appear to agree well with those available in the literature.

7.1 Euler equations and numerical discretization

Two-dimensional Euler equations, representing conservation of mass, momentum and energy, in Cartesian coordinates can be written as

$$\frac{\partial}{\partial t}\mathbf{W} + \frac{\partial}{\partial x}\mathbf{F}_1(\mathbf{W}) + \frac{\partial}{\partial y}\mathbf{F}_2(\mathbf{W}) = 0 \tag{7.1}$$

or

$$\frac{\partial}{\partial t}\mathbf{W} + \nabla \bullet \mathbf{F}(\mathbf{W}) = 0 \tag{7.2}$$

where $\mathbf{W}$ is the vector of conservative variables, $\mathbf{F_1}$ and $\mathbf{F_2}$ are the convective flux vectors. They are given by

$$\mathbf{W} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad \mathbf{F_1}(\mathbf{W}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho u(E + p/\rho) \end{pmatrix}, \quad \mathbf{F_2}(\mathbf{W}) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho v(E + p/\rho) \end{pmatrix}, \quad \mathbf{F} = [\mathbf{F_1}, \mathbf{F_2}].$$

where ρ represents the fluid density, u and v are the x and y components of the velocity, E is the total energy, and p is the pressure. For perfect gas, the equation of state can be written as:

$$p = (\gamma - 1)\rho(E - \frac{u^2 + v^2}{2}) \tag{7.3}$$

where γ is the ratio of specific heats.

When using classical numerical methods to solve the Euler equations with discontinuity, the central difference scheme was found to be unconditionally unstable. Therefore, upwind schemes, in which the stencil for spatial discretization is biased according to the sign of the local velocity, are proposed. Examples are the CIR scheme (Courant, Isaacson and Rees, 1952) and Lax-Friedrich scheme. Both of them are first-order accurate. First-order schemes provide stable computations but suffer from heavily smeared discontinuities in the solution due to large numerical dissipation.

Second-order schemes include artificial viscosity schemes such as Lax-Wendroff (1960) and MacCormack (1969) schemes. For this family of methods, an additional "artificial viscosity" term is added to the hyperbolic partial differential equations (Leveque 1992). The additional viscosity term is only needed near discontinuities, therefore the coefficient depends on the behavior of the solution, e.g., large value near

discontinuities and small value in smooth regions. Jameson, Schmidt and Turkel (1981) proposed the central difference scheme with artificial dissipation operator (JST scheme), which has better control of the amount of dissipation introduced.

Many modern CFD methods for Euler equations relate the approximation of fluxes to the direction of local propagation of characteristics. These methods can be generally categorized into Flux Difference Splitting (FDS) methods and Flux Vector Splitting (FVS) methods (Eleuterioi 2009). A recent scheme of Advection Upstream Splitting Method (AUSM) that attempts to combine the features of the FVS and FDS methods is due to Liou and Steffen (1993).

Roe scheme (Roe 1981) is the most popular FDS method. The idea comes from Godunov's scheme (Godunov 1959) which is based on the solution of the Riemann problem. Unlike the Godunov scheme in seeking the exact solution of the piecewise Riemann problem, Roe scheme is based on approximate solution of the Riemann problem. This makes Roe scheme less computationally intensive and generally simpler. In the current LDFD Euler solver, Roe scheme is adopted. Unlike in the LDFD incompressible solver, where the second-order scheme (central difference) is adopted for the approximation of spatial derivatives, LDFD Euler solver with Roe scheme is only first order in space. Therefore, to get sharp resolution of the shock waves, one needs to use very fine mesh.

7.2 LDFD Euler solver

7.2.1 Numerical discretization

Finite Volume method is used for spatial discretization in LDFD Euler solver. The flow variables **W** and the flux quantities $F \bullet S$ (where S is the area of interface of the control volume) are evaluated at the cell center and cell faces, respectively. Therefore, using Gauss' divergence theorem, the governing equations (7.2) can be written in integral form as:

$$\frac{\partial}{\partial t}\int_{\Omega_{i,j}} \mathbf{W} d\Omega + \int_{\partial\Omega_{i,j}} \mathbf{F} \bullet \mathbf{n} ds = 0 \tag{7.4}$$

for a control volume $\Omega_{i,j}$ with boundary $\partial\Omega_{i,j}$. Here, **n** denotes the unit outward normal to $\partial\Omega_{i,j}$ and **F** is the tensor of flux density. The boundary $\partial\Omega_{i,j}$ is given by

$$\partial\Omega_{i,j} = d_{i,j-\frac{1}{2}} + d_{i+\frac{1}{2},j} + d_{i,j+\frac{1}{2}} + d_{i-\frac{1}{2},j} \tag{7.5}$$

where $d_{i,j-\frac{1}{2}}, d_{i+\frac{1}{2},j}, d_{i,j+\frac{1}{2}}, d_{i-\frac{1}{2},j}$ are the four cell faces of the control volume. Suppose that the flow quantities **W** are taken to be volume averaged of the control volume $\Omega_{i,j}$, hence:

$$\mathbf{W}_{i,j} = \frac{1}{S_{i,j}}\int_{\Omega_{i,j}} \mathbf{W} d\Omega \tag{7.6}$$

where $S_{i,j}$ is the cell area of $\Omega_{i,j}$. The discrete form of equation (7.4) is written as:

$$S_{i,j}\left(\frac{d}{dt}\mathbf{W}_{i,j}\right) + \mathbf{Q}_{i,j} = 0 \tag{7.7}$$

where $\mathbf{Q}_{i,j}$ represents the net flux out of a cell. The flux term $\mathbf{Q}_{i,j}$ is given by

$$\mathbf{Q}_{i,j} = \mathbf{F}\cdot d\Big|_{i,j+\frac{1}{2}} - \mathbf{F}\cdot d\Big|_{i,j-\frac{1}{2}} + \mathbf{F}\cdot d\Big|_{i+\frac{1}{2},j} - \mathbf{F}\cdot d\Big|_{i-\frac{1}{2},j} \tag{7.8}$$

The value of **F** is calculated using the Roe scheme. For example, the value of $\mathbf{F}_{i,j+\frac{1}{2}}$ can be evaluated by the formula below:

$$\mathbf{F}_{i,j+\frac{1}{2}} = \frac{1}{2}\left[\mathbf{F}(\mathbf{W}_L) + \mathbf{F}(\mathbf{W}_R)\right] - \frac{1}{2}\mathbf{L}^{-1}\left|\tilde{\mathbf{\Lambda}}\right|\mathbf{L}(\mathbf{W}_R - \mathbf{W}_L) \tag{7.9}$$

where subscripts *L* and *R* denote the two sides of cell face, $\tilde{\mathbf{\Lambda}}$ is the eigenvalue matrix of $\tilde{\mathbf{A}}$. Then $\tilde{\mathbf{A}}$ can be written as:

$$\tilde{\mathbf{A}} = \mathbf{R}(\tilde{\mathbf{W}})\tilde{\mathbf{\Lambda}}\mathbf{L}(\tilde{\mathbf{W}}) \tag{7.10}$$

where vectors **L, R** are the left and right eigenvectors of $\tilde{\mathbf{A}}$. $\tilde{\mathbf{W}}$ can be obtained via following so called *Roe-average variables*.

$$\begin{aligned}
\tilde{\rho} &= \sqrt{\rho_L \rho_R} \\
\tilde{u} &= \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \\
\tilde{v} &= \frac{\sqrt{\rho_L}v_L + \sqrt{\rho_R}v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{H} &= \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\
\tilde{a}^2 &= (\gamma - 1)\left[\tilde{H} - \frac{1}{2}(\tilde{u}^2 + \tilde{v}^2)\right]
\end{aligned} \tag{7.11}$$

In Roe scheme, $\tilde{\mathbf{A}} = \tilde{\mathbf{A}}(\mathbf{W}_L, \mathbf{W}_R)$ is a constant Jacobian matrix, which is used to replace the original Jacobian matrix $\mathbf{A} = \mathbf{A}(\mathbf{W})$. By setting $\Delta\mathbf{W} = \mathbf{W_R} - \mathbf{W}_L$, and using **RL=I**, we have

$$\mathbf{L^{-1}}\left|\tilde{\mathbf{\Lambda}}\right|\mathbf{L(W_R - W_L)} = \mathbf{RLL^{-1}}\left|\tilde{\mathbf{\Lambda}}\right|\mathbf{L}\Delta\mathbf{W} = \mathbf{R}\left|\tilde{\mathbf{\Lambda}}\right|\mathbf{L}\Delta\mathbf{W} = \sum_{j=1}^{m}\tilde{\alpha}_j\left|\tilde{\lambda}^{(j)}\right|r_{(j)} \tag{7.12}$$

In Eq. (7.12),

$\mathbf{R} = (r_{(1)}, r_{(2)}, r_{(3)}, r_{(4)}), \tilde{\alpha}_j = \begin{bmatrix} l_{(1)} \\ l_{(2)} \\ l_{(3)} \\ l_{(4)} \end{bmatrix}\Delta\mathbf{W}, \lambda^{(j)}$ is the eigenvalue of $\tilde{\mathbf{A}}$,

where

$$\tilde{\lambda}^{(1)} = \tilde{u} - \tilde{a}, \tilde{\lambda}^{(2)} = \tilde{\lambda}^{(3)} = \tilde{u}, \tilde{\lambda}^{(4)} = \tilde{u} + \tilde{a} \tag{7.13}$$

For the two-dimensional problem, we have

$$
\begin{aligned}
r_{(1)} &= (1, \tilde{u} - \tilde{a}n_x, \tilde{v} - \tilde{a}n_y, \tilde{H} - \tilde{u}\cdot\vec{n}\tilde{a})^T \\
r_{(2)} &= (1, \tilde{u}, \tilde{v}, \frac{1}{2}(\tilde{u}^2 + \tilde{v}^2))^T \\
r_{(3)} &= (0, n_y, -n_x, \tilde{u}n_y - \tilde{v}n_x)^T \\
r_{(4)} &= (1, \tilde{u} + \tilde{a}n_x, \tilde{v} + \tilde{a}n_y, \tilde{H} + \tilde{u}\cdot\vec{n}\tilde{a})^T
\end{aligned}
\tag{7.14}
$$

$$
\tilde{\alpha}_j = \begin{bmatrix} l_{(1)} \\ l_{(2)} \\ l_{(3)} \\ l_{(4)} \end{bmatrix} \mathbf{\Delta W} = \begin{bmatrix} \frac{1}{2\tilde{a}^2}(\Delta p - \tilde{\rho}\tilde{a}\Delta u_n) \\ \frac{1}{\tilde{a}^2}(\tilde{a}^2\Delta\rho - \Delta p) \\ \frac{1}{\tilde{a}}\tilde{\rho}\Delta u_t \\ \frac{1}{2\tilde{a}^2}(\Delta p + \tilde{\rho}\tilde{a}\Delta u_n) \end{bmatrix} \tag{7.15}
$$

where

$$
\begin{aligned}
u_n &= \mathbf{u}\cdot\mathbf{n} = un_x + vn_y \\
u_t &= \mathbf{u}\cdot\mathbf{t} = un_y - vn_x
\end{aligned}
\tag{7.16}
$$

After getting the flux term, Eq. (7.7) can be solved by time marching scheme.

7.2.2 Implementation of boundary condition

Solid boundary condition

In the presence of solid bodies, the LDFD treatment of boundary conditions generally includes:

1) No-penetration wall condition;
2) Satisfaction of local momentum equation in the normal direction to the wall;
3) Isentropic condition at the wall;
4) Conservation of total enthalpy along the normal direction to the wall.

In Figure 7.1, the interior mesh nodes are represented by the solid circle (such as B1,

B2), the exterior nodes are represented by shade circles, and open circles represent exterior dependent nodes (such as E, H). The open squares represent interior dependent nodes (such as A1, A2). The solid squares on the boundary are points intersected by the lines along the normal direction to the wall through the exterior dependent node and boundary curve. The “mirror point” of a given exterior dependent node is the interior point that has the equal normal distance to the wall as that of the exterior dependent node. They are represented by the solid squares in the fluid domain as shown in Figure 7.1, such as E' and H', which are the mirror points of E and H, respectively.

To implement boundary conditions at the solid wall, one can take advantage of the information at the mirror point (Zhou et al 2007). Thus, to get the functional values at a mirror point, an interpolation scheme is needed, and bilinear interpolation scheme normally serves this purpose well. However, it is found that some mirror points are located so near to the solid wall, occasionally leading to failure of bilinear interpolation. Taking the exterior dependent point E in Figure 7.2 as an example, the line along the normal direction linking E and E' intersects the wall at the point W. Because EW=E'W is very small, the mirror point E’ falls into the cell $\overline{EA1\,D1A2}$ in which not all the vertices are interior points (E is the exterior dependent point whose variable values are unknown).

Therefore, instead of finding the mirror point E’, we turn to find the image point F for

E along the normal direction, where $|FW| = \alpha \min(dx, dy)$, dx and dy denote the nodal space in the x or y direction. α is the coefficient which makes the point F fall into the cell whose vertices are the interior points (such as cell $\overline{A1B1C1D1}$). Here we take α=1.5.

If F falls into the cell consisting of $A1$ (x_1, y_1),B1 (x_2, y_1), $C1$ (x_2, y_2) and $D1$ (x_1, y_2), then bilinear interpolation for $u_F(x, y)$ can be written as

$$u_F(x,y) \approx \frac{1}{(x_1 - x_2)(y_2 - y_1)}[u_{A1}(x - x_2)(y_2 - y) + u_{B1}(x_1 - x)(y_2 - y) + u_{C1}(x_1 - x)(y - y_1) + u_{D1}(x - x_2)(y - y_1)] \tag{7.17}$$

Similarly, the variable values at the image point F such as pressure p_F and density ρ_F,velocity v_F can also be obtained in the same way like Eq. (7.17).

After getting the variable values at image point, we can find the variable values at the exterior dependent points. Suppose that, we want to get variable values at the exterior dependent point E, i.e., pressure, density, and velocity p_E, ρ_E, u_E, v_E. As shown in Figure 7.2, $\vec{n}$ is the normal direction of the boundary, $\vec{t}$ is the tangential direction of the boundary. The velocity components of $\mathbf{u}_F$ at $\vec{n}$ and $\vec{t}$ direction can be expressed as

$$\begin{aligned} u_{nF} &= u_F n_x + v_F n_y \\ u_{tF} &= u_F t_x + v_F t_y \end{aligned} \tag{7.18}$$

The solid boundary condition in the LDFD solver can be implemented by the following steps:

1) Applying the no-penetration wall condition ($u_{nW} = 0$) at the mirror point E', we

have

$$u_{nE} = -u_{nE'} \tag{7.19}$$

Since $|E'W| = |EW| = \Delta s_E$, $|FW| = \Delta s_F$ are all known, we can easily compute

$$\begin{aligned} u_{nE'} &= \frac{\Delta s_E}{\Delta s_F} u_{nF} \\ u_{nE} &= -\frac{\Delta s_E}{\Delta s_F} u_{nF} \end{aligned} \tag{7.20}$$

2) Local momentum equation at the boundary point W along the normal direction $\vec{n}$ can be written as

$$\left(\frac{\partial p}{\partial n}\right)_W == \frac{p_F - p_E}{\Delta s_F + \Delta s_E} = \frac{\rho_W (u_{tW})^2}{R_W} \tag{7.21}$$

Applying no shear stress boundary condition at the wall, that is

$$u_{tW} = u_{tF} \tag{7.22}$$

Substituting Eq. (7.22) and $\rho_W = \rho_F$ into Eq.(7.21), we have

$$p_E = C p_F \tag{7.23}$$

where

$$C = 1 - (\Delta s_F + \Delta s_E) \frac{\rho_F}{p_F} \frac{(u_{tF})^2}{R_W} \tag{7.24}$$

3) Using the isentropic condition at the wall

$$\frac{p_E}{\rho_E^\kappa} = \frac{p_F}{\rho_f^\kappa} \tag{7.25}$$

together with Eq.(7.23), we can get

$$\rho_E = C^{\frac{1}{\kappa}} \rho_F \tag{7.26}$$

4) Applying total enthalpy conservation along the normal direction (Dadone et al, 2004) gives

$$\frac{u_{nE}^2+u_{tE}^2}{2}+\frac{\kappa}{\kappa-1}\frac{p_E}{\rho_E}=\frac{u_{nF}^2+u_{tF}^2}{2}+\frac{\kappa}{\kappa-1}\frac{p_F}{\rho_F} \tag{7.27}$$

Then we have

$$u_{tE}=\sqrt{2\left[\frac{u_{nF}^2+u_{tF}^2}{2}+\frac{\kappa}{\kappa-1}\left(\frac{p_F}{\rho_F}-\frac{p_E}{\rho_E}\right)\right]-u_{nE}^2} \tag{7.28}$$

After obtaining $u_{nE'}, u_{tE}$, the velocity at *E* can be obtained by

$$\begin{aligned} u_E &= u_{nE}n_x + u_{tE}t_x \\ v_E &= u_{nE}n_y + u_{tE}t_y \end{aligned} \tag{7.29}$$

Through the above process, the solid boundary conditions are implemented, and variables (pressure p_E, density ρ_E,velocity u_E v_E) at the exterior dependent point *E* are computed.

Far field boundary condition

In this study, the local one-dimensional characteristic boundary conditions are adopted at the far field boundary (Morinishi, 1992). The Riemann invariants are introduced corresponding to the incoming and outgoing characteristic as:

$$R_\infty=\mathbf{u}_f\cdot\mathbf{n}-\frac{2a_f}{\gamma-1},\quad R_e=\mathbf{u}_e\cdot\mathbf{n}+\frac{2a_e}{\gamma-1} \tag{7.30}$$

where *a* is the speed of sound, **n** is the unit vector normal to the boundary. The subscripts *f* and *e* denote the free stream values and the values extrapolated from neighboring interior nodes, respectively. The actual normal velocity component and sound speed at the boundary are given by

$$u_n=\frac{1}{2}(R_e+R_\infty),\quad a=\frac{\gamma-1}{4}(R_e-R_\infty) \tag{7.31}$$

Two other conditions for two dimensional flow problems are needed. For the inflow

boundary, the tangential velocity component and entropy can be specified at their free stream values. For the outflow boundary, they are extrapolated from the interior points.

7.3 Numerical examples

For the application of the present LDFD Euler solver to different test cases, we use perfect gas as the fluid, i.e., $\gamma = 1.4$.

7.3.1 Inviscid flow past a 2D circular cylinder

The inviscid flow past a circular cylinder is a classic flow problem in fluid dynamics. The configuration is shown in Figure 7.3. In the Cartesian coordinate system, the flow past a circular cylinder is a flow problem with curved boundary. Therefore it serves as a good test case for the LDFD Euler solver.

A square of 10D×10D (*D* is the diameter of the cylinder) is taken as the computational domain with uniform mesh with Δx=Δy=0.005D. In this case, $U_\infty = 0.1$, $V_\infty = 0$, $p_\infty = 0.7143$, $\rho_\infty = 1$.

The analytical solution for this case can be obtained by superposition of a doublet and uniform flow. The pressure coefficient distribution is predicted by the theory, given by the expression of

$$C_p = 1 - 4\sin^2\theta \tag{7.32}$$

where θ is angle measured from the back of the cylinder, as shown in Figure 7.3. In

the numerical simulation, C_p can be defined as

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty U_\infty^2} \tag{7.33}$$

Figure 7.4 compares the present surface pressure coefficient with the theoretical solution, and it is clear that a good agreement has been achieved.

Figure 7.5 shows the streamlines around the cylinder. It is found that the streamlines are symmetrical, which is consistent with the theoretical analysis.

7.3.2 Supersonic flow in a wedge channel

To test the shock-capturing capability of LDFD Euler solver, we choose the supersonic flow in a wedge channel as the test problem. A supersonic flow with Ma=2.0 in a wedge channel was considered. The configuration of the channel is shown in Figure 7.6, where there is 15° wedge on the lower wall.

The length of computational domain in the x and y direction is XL=3; YL=1, respectively, and the Cartesian mesh with $\Delta x=\Delta y=0.00625$ is used. For this case, the flow parameters in the free stream are $U_\infty = 2$, $V_\infty = 0$, $p_\infty = 1$, $\rho_\infty = 1.4$.

The numerical results are visualized by the Mach number distribution, as shown in Figure 7.7. It is found that an oblique shock wave is produced by the wedge. The shock reflects from the upper wall and interacts with the expansion wave. The expansion wave also reflects from the upper wall slightly downstream of the

shock-reflection point.

Some flow parameters as shown in Figure 7.8 are compared with the theoretical solutions. The analytical calculations predict a post shock Mach number of 1.454, while present computation gives 1.456. The angle of wedge shock (β as shown in Fig. 7.8) emitting from the front wedge point is 45.38° by theoretical analysis, and our simulation gives 44.27°.

Clearly, our results are in good agreement with those from theoretical analysis. It shows that the LDFD method is able to capture the discontinuities such as shock in the compressible flow simulations.

7.3.3 Compressible flow over NACA0012 airfoil

The inviscid subsonic or transonic flow over a NACA0012 airfoil is the thin object problem (referred in Chapter 2) as the airfoil has a sharp trailing edge.

The outer boundary of the computational domain is 10 chords away from the airfoil surface. The mesh spacing is $\Delta x=\Delta y=0.005$. In the following two cases, we use $p_\infty = 0.7143$, $\rho_\infty = 1$.

Case 1: Subsonic flow over NACA0012 with free stream Ma=0.3 at the attack angle AOA=0°

This case is primarily used for validation of LDFD method for the subsonic flows.

Figure 7.9 shows the pressure contours, where the flow appears to be symmetric due to the zero angle of attack. Hence there is no lift being produced. It is also found that at such a low Mach number, the flow does not incur any shock.

The pressure coefficient C_p distribution on the surface of airfoil is given in Figure 7.10. The comparison of present results with the experimental data from Ladson et al (1987) shows good agreement.

Case 2: Transonic flow over NACA0012 at Ma=0.8, AOA=*0°* and 1.25°

We consider the transonic flows over a NACA0012 airfoil at Ma=0.8 and AOA=*0°* and 1.25°, in which shocks are known to occur. The same mesh used in the previous subsonic flow simulation is adopted again. Figures 7.11 and 7.13 show the pressure contours for both cases with different AOA. Unlike in the subsonic flow case, in this transonic flow case, there are two shocks which can be clearly recognized on both sides of airfoil surface. For zero angle of attack, the shocks appear symmetrically near the mid-chord. For AOA=1.25°, the strong upper shock wave is located at about 0.626 of length of chord, and the weak lower shock wave occurs at about 0.364 of length of chord. Both are well captured by the LDFD Euler solver. Figures 7.12 and 7.14 show the comparison of surface pressure coefficient C_p between the present results and those in the literature. Good agreements have been achieved.

7.4 Concluding remarks

In this study, the two-dimensional LDFD Euler solver is presented. It is developed based on Roe scheme and framework of Finite Volume Method. Unlike the cut cell Cartesian mesh solvers, the LDFD Euler solver treats the solid boundary in a simpler and straightforward way. The LDFD Euler solver is then validated by applying to simulate inviscid, subsonic, transonic and supersonic flows with different geometries. Numerical experiments show that it can capture main features of the flow properly, such as shock position and strength. The good agreement with data in the literature indicates that the LDFD Euler solver can work quite accurately and robustly.

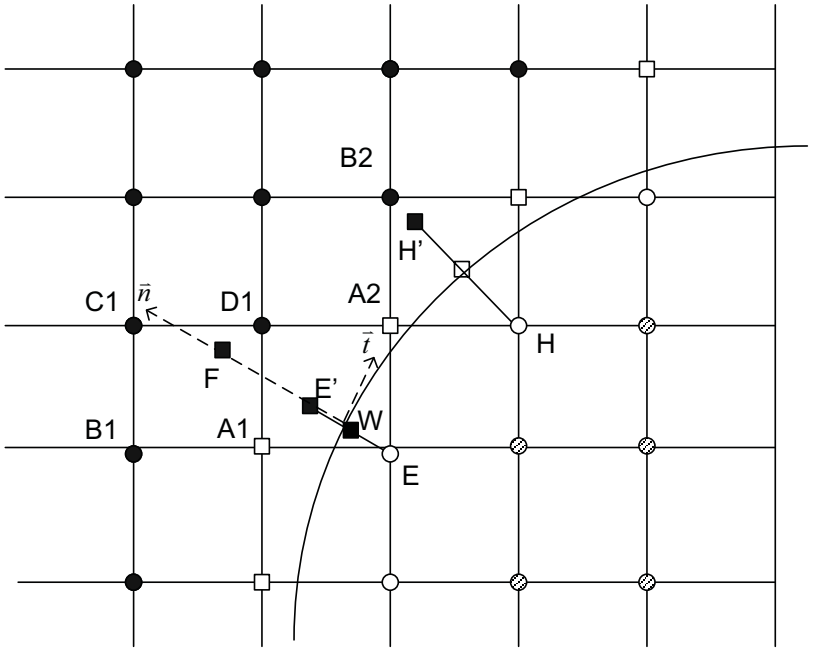


Figure 7.1 Treatment of solid boundary condition

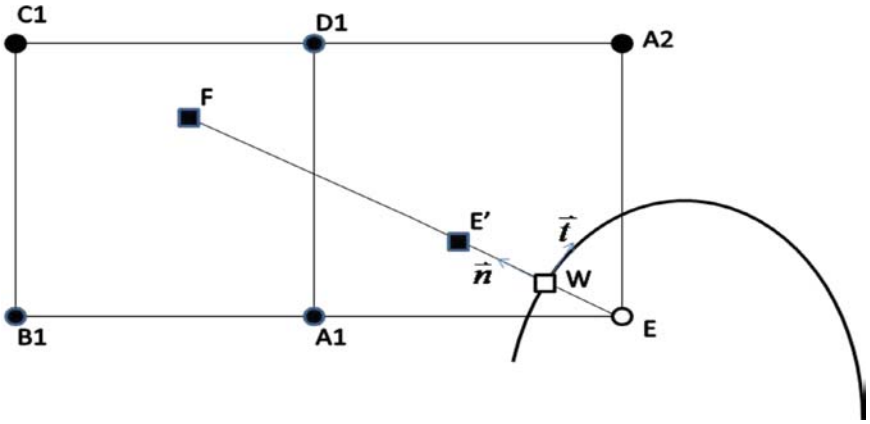


Figure 7.2 Mirror point, image point and its interpolation domain

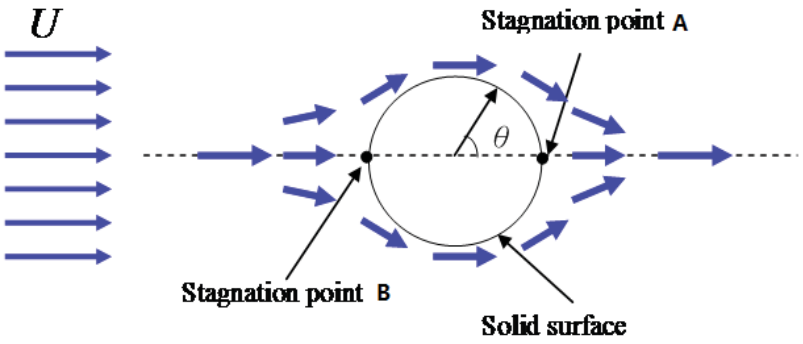


Figure 7.3 Inviscid flow past a circular cylinder

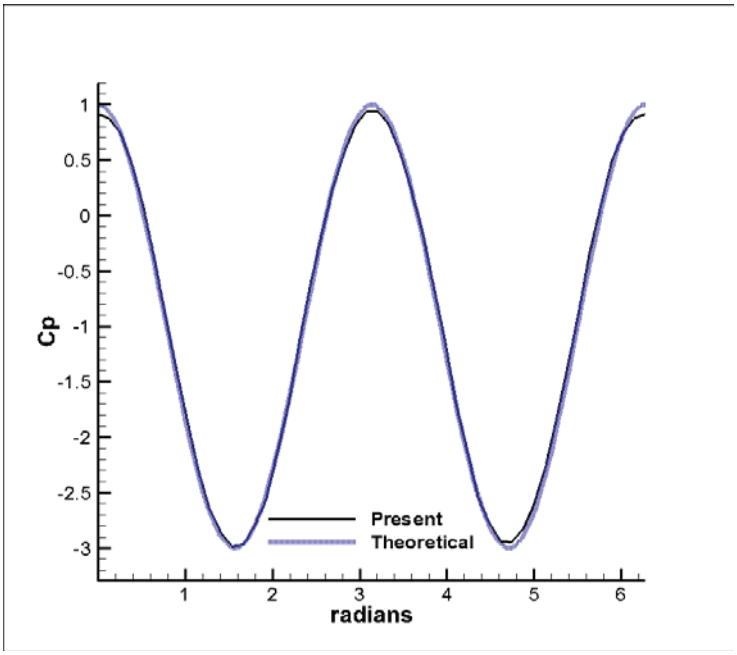


Figure 7.4 Pressure coefficient distribution along the surface of the cylinder

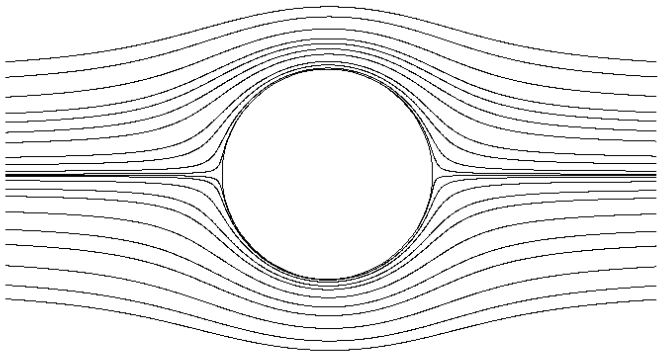


Figure 7.5 Streamlines around the cylinder by the present simulation

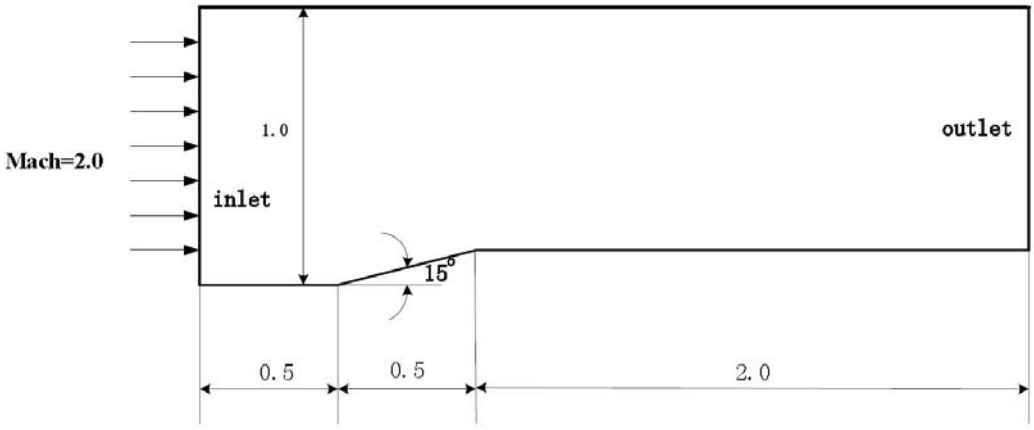


Figure 7.6 Configuration for the supersonic flow in a wedge channel

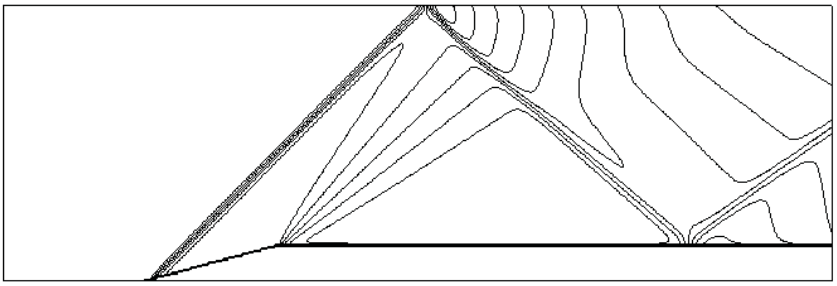


Figure 7.7 Mach number distribution for supersonic flow in a wedge channel

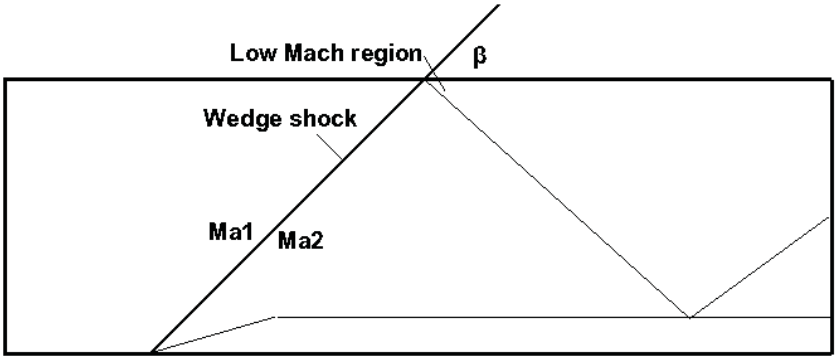


Figure 7.8 Illustration of some flow parameters in a supersonic flow in wedge channel

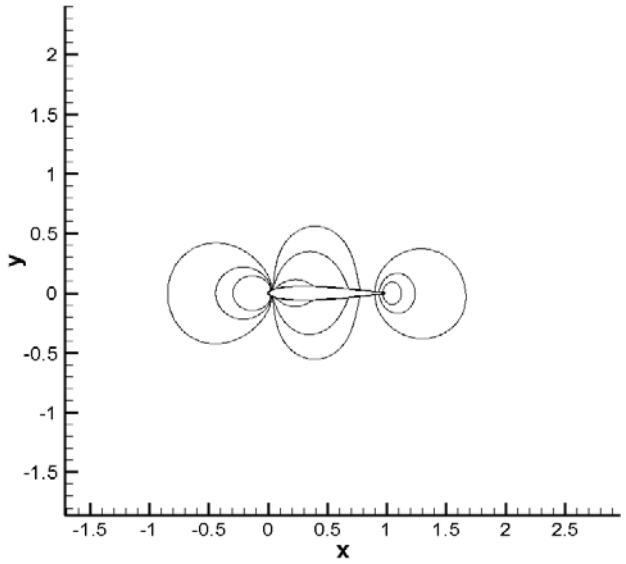


Figure 7.9 Pressure contours for the subsonic flow over NACA0012 （$M_\infty = 0.3$，$\alpha = 0^o$）

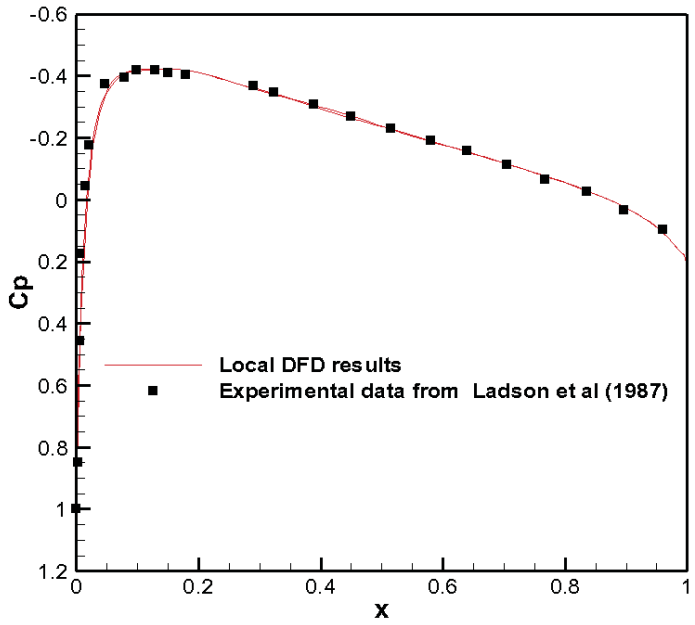


Figure 7.10 Present numerical solution for pressure coefficient distribution, C_p, compared with Experimental data.

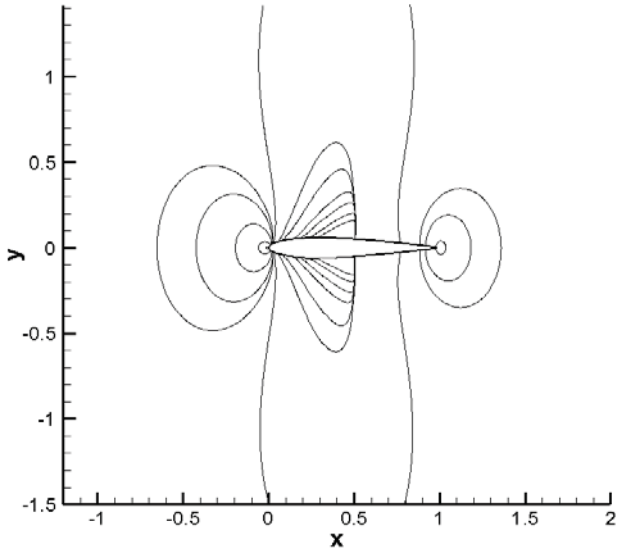


Figure 7.11 Pressure contours for the transonic flow over NACA0012（$M_\infty = 0.8$，$\alpha = 0^\circ$）

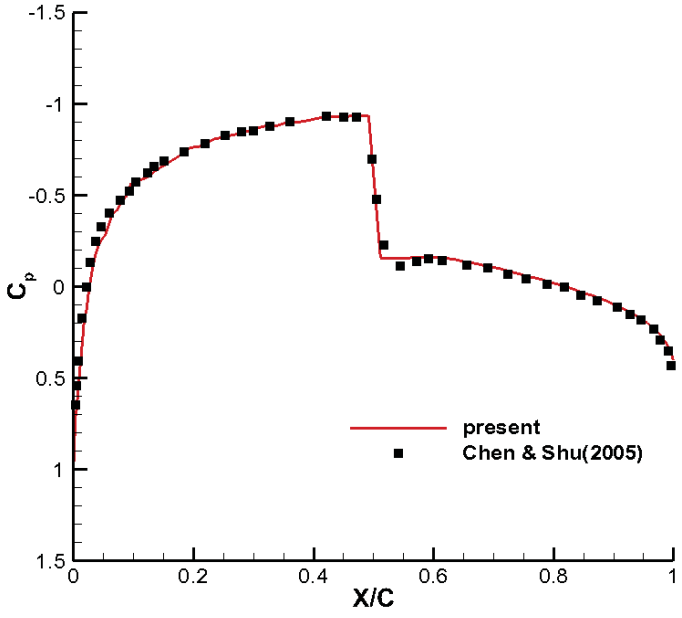


Figure 7.12　Pressure coefficients C_p on airfoil surface　（$M_\infty = 0.8$，$\alpha = 0^\circ$）

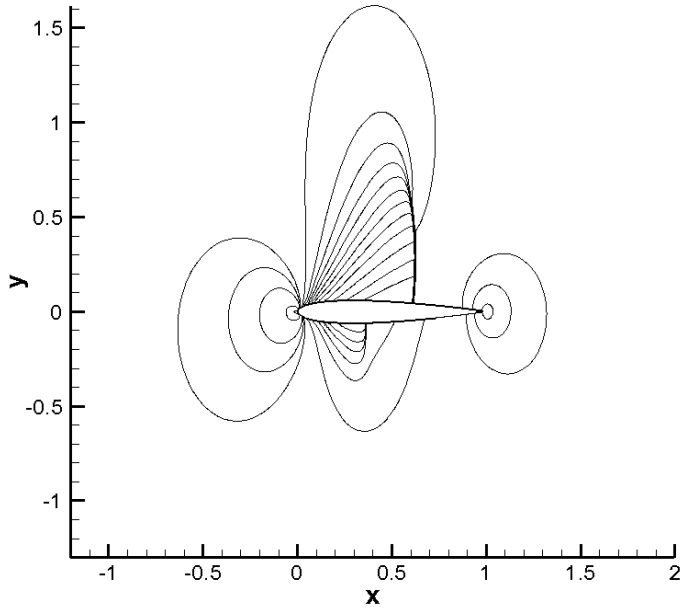


Figure 7.13 Pressure contours for the transonic flow over NACA0012（$M_\infty = 0.8$，$\alpha = 1.25^\circ$）

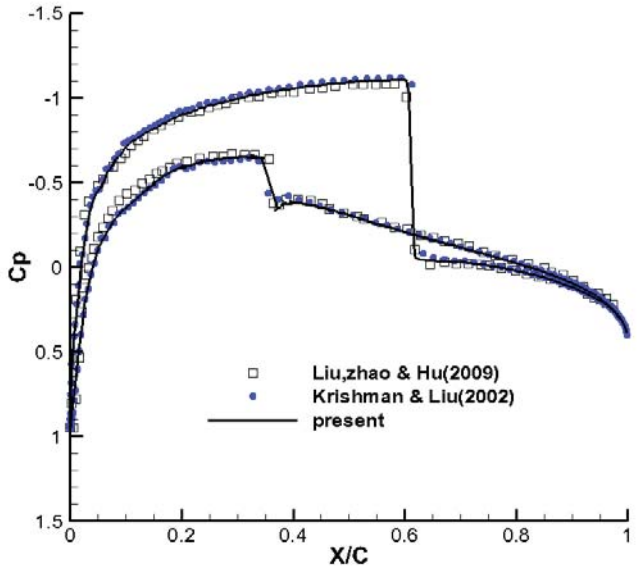


Figure 7.14 Pressure coefficients C_p on airfoil surface （$M_\infty = 0.8$，$\alpha = 1.25^\circ$）

CHAPTER 8

Conclusions and Recommendations

8.1 Conclusions

In this thesis, two non-mesh-conforming methods, LDFD and LDFD-IBM, are presented. Since they are implemented on a Cartesian mesh, they can be considered among the family of Cartesian mesh solvers. The benefits of a Cartesian mesh solver can be summarized as follows.

(1) A lot of effort in generating the mesh is saved, especially for flow problems with complex geometries or moving boundaries;

(2) The well-developed computational techniques for Cartesian mesh can be easily adopted. For example, the recently developed adaptive stencil refinement approach has been introduced in LDFD and LDFD-IBM, which significantly enhances the computational efficiency.

For both LDFD and LDFD-IBM, it is a crucial step to identify the status of mesh nodes, which not only determines where the governing equations should be solved, but also indicates the position of the solid boundaries. A fast algorithm for the status identification has been proposed based on the Cartesian mesh. Once the statuses of all the nodes are known, both LDFD and LDFD-IBM can provide simple and efficient solvers for the flow problem with stationary and moving boundaries.

Unlike the Global DFD method that uses polynomials of very high order to approximate the functions and derivatives, the LDFD method uses low order

polynomials, and thus the numerical discretization and the extrapolation scheme only involve a few neighboring nodes around the target node. This not only makes the treatment of solid boundary more flexible, but also reduces the risk of numerical instability due to extrapolation errors. In addition to incompressible flow simulations, the LDFD method has also been shown to have the capability of accurately simulating inviscid compressible flows with discontinuities such as shocks, when it is coupled with the Roe scheme under the Finite Volume framework. These features suggest that the LDFD method is a powerful tool for a wide range of flow problems with complex geometries, either incompressible or compressible.

The LDFD-IBM is a hybrid method which combines the LDFD method and the Immersed Boundary Method (IBM). Compared to the LDFD method, the LDFD-IBM does not need to enforce the pressure boundary condition at the embedded boundaries, and thus greatly simplifies the implementation of the method. This is particularly true in the study of moving boundary problems and three-dimensional flows with complex geometry. Compared to the conventional IBM, the LDFD-IBM avoids the problem of flow penetration into the solid wall, which arises from the fact that the no-slip boundary condition is not strictly satisfied in the conventional IBM. In this sense, LDFD-IBM inherits the merits from both the LDFD and the IBM.

One of the most attractive features of LDFD and LDFD-IBM is that they are able to simulate moving boundary problems. Compared to the simulation of flows over stationary embedded objects, the major difference lies in the regular update of the

node status and providing the flow variables at the nodes where the status change occurs. In LDFD, the solution procedure for moving boundary problems is almost the same as that for the stationary boundary problems because the variable values are readily available at the nodes where the status changes from "solid" to "fluid" if we control the time interval small enough to avoid the jump of the node status. In LDFD-IBM, a two-steps forcing procedure, which takes account of both the status-changing effect and acceleration effect of the moving objects, is presented. The two methods are used to study the moving boundary problems with irregular boundaries. Numerical results indicate that both methods have good performance on this type of problems.

Because of its easy implementation, the LDFD-IBM is primarily used to study the three-dimensional flow problems with complex boundaries. To maximize the computational efficiency, two meshing strategies are adopted. One is to use non-uniform mesh scheme, and the other is to use the AMR algorithm. The former is applied to three-dimensional flow past a sphere and flow over a torus. The latter is used to simulate the three-dimensional flow past a circular cylinder. Both give good agreement with the literature data.

All the numerical examples demonstrate that the two methods have the potential to simulate practical flow problems with moving complex objects encountered in engineering applications.

8.2 Recommendations

The further development of LDFD and LDFD-IBM is recommended to focus on the following topics:

1) Multi-Grid acceleration

Solving the pressure Poisson equation is the most time-consuming part in the LDFD and LDFD-IBM solvers when the solution of Navier-Stokes equations in the primitive variable form is pursued. Multi-grid acceleration is known for its fast convergence in solving Poisson type of equations. Moreover, implementation of Multi-Grid acceleration technique on a Cartesian mesh is supposed to be easy and straightforward.

2) Implementation of two methods on the other type of meshes

In principle, LDFD and LDFD-IBM can be used in any kind of mesh, although the generation of a Cartesian mesh is rather simple and efficient. However, for practical flow problems, the Cartesian mesh may not be the most efficient one. We may use these two methods on the other types of mesh (such as unstructured mesh, overset mesh or hybrid mesh), to capture the flow features more accurately and efficiently.

3) Development of high-order LDFD or LDFD-IBM schemes

In this study, the overall discretization accuracy of LDFD and LDFD-IBM is second order. It may not be sufficient when turbulent flows are considered. In this regard, high-order LDFD or LDFD-IBM schemes are more suitable and should be developed.

4) Development of LDFD-IBM solver for the compressible flow

In this work, the LDFD-IBM is restricted to the simulation of incompressible flows. It will be interesting to explore the possibility of development of LDFD-IBM solver for the compressible flow.

References

[1] Arad, N., N.Dyn, and D.Reisfeld, (1994): Image warping by radial basis function: applications to facial expressions, *CVGIP: Graphical models and image processing* 56:161-172

[2] Bao Y., Zhou D. and Huang C. (2010): "Numerical Simulation of flow over three circular cylinders in equilateral arrangements at low Reynolds number by a second-order characteristic-based split finite element method", *Computers & Fluids* 39: 882-889.

[3] Bearman P.W. and Wadcock A.J. (1973): "The interaction between a pair of circular cylinders normal to a stream", *J. Fluid Mech. 61*, 499-511

[4] Behara S and Mittal S. (2010): Flow past a circular cylinder at low Reynolds number: Oblique vortex shedding, Physics of Fluids 22(5):1-14

[5] Bellman RE, Casti J (1971): Differential Quadrature and long-term integration, *J. Math. Anal. Appl*, Vol 34:235-238

[6] Bellman RE, Kashef B.G., and Casti J. (1972): Differential quadrature: a technique for the rapid solution of nonlinear partial differential equations, *J. Comput. Phys.,* 10:40-52.

[7] Berger M.J. and Oliger J. (1984): Adaptive mesh refinement for hyperbolic partial differential equations, *J. Comput. Phys.* 53, 484-512.

[8] Berger M.J. and LeVeque R. (1998): Adaptive mesh refinement for two-dimensional hyperbolic systems and the AMRCLAW software, *SIAM J. Numer. Anal.* 35, 2298-2316.

[9] Braza M, Chassaing P, Ha Minh H. (1986): Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *J. Fluid Mech*, 165:79

[10] Calhoun D. (2002), A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equatins in irregular regions. *J. Comput.Physics*: 176:231-275

[11] Canuto, C. (1988): *Spectral methods in fluid dynamics*, Springer-Verlag, New York

[12] Chai J.C., Lee H.S., and Patankar S.V. (1994), Treatment of Irregular Geometries using a Cartesian Coordiantes finite-volume radiation heat transfer procedure, *Numer. Heat Transfer B*, Vol 26 : 179~197.

[13] Chang K.S. and Song C.J. (1990): Interactive vortex shedding from a pair of circular cylinders in a transverse arrangement, *Int. J. numer. Methods Fluids*, 11:317-329

[14] Chen HQ and Shu C (2005): "An efficient Implicit mesh free method to solve two-dimensional compressible Euler equations", *International Journal of Modern Physics* C Vol 16,No.3 439-454

[15] Chen S, Doolen GD. (1998): Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*; 30:329-364

[16] Choi H. and Moin P. (1994): Effects of the computational time step on numerical solutions of turbulent flow, *Journal of Computational Physics* 113, 1-4.

[17] Colella P, Graves DT, Keen BJ, Modiano D. (2006): A Cartesian grid embedded boundary method for hyperbolic conservation laws. *Journal of Computational Physics* 211:347–366

[18] Corier W. and Powell K.G.(1995): An accuracy assessment of Cartesian-Mesh Approaches for the Euler Equations. *Journal of Computational Physics*, 117: 121-131

[19] Courant R., Isaacson E., and Rees M.. (1952): On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences. Comm. Pure. Appl. Math., 5:243–255.

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

[21] Dadone A. and Grossman B.(2007): Ghost-cell method for inviscid three-dimensional flows on Cartesian grids, *Comput. Fluids*, 36:1513–1528.

[22] Dennis S.C.R., Chang G.Z. (1970): Numerical solutions for steady flow past a circular cylinder at Reynolds number up to 100, *J. Fluid. Mech*. 42: 471

[23] Ding H. and Shu C. (2006): A Stencil Adaptive Algorithm for Finite Difference Solution of Incompressible Viscous Flows, *J. Comput. Phys*., 214:397-420

[24] Ding H, Shu C, Cai Q.D. (2007): Applications of stencil-adaptive finite difference method to incompressible viscous flows with curved boundary, *Computers & Fluids* 36:786-793

[25] Ding H, Shu C, Yeo K.S., and Xu D. (2004): Simulation of incompressible viscous flows past circular cylinder by hybrid FD scheme and meshless least square-based finite difference method, *Comput. Methods Appl. Mech. Engrg.* 193:727-744

[26] Ding, H., Shu, C., Yeo, K. S., and Xu, D.(2007): "Numerical Simulation of Flows Around Two Circular Cylinders by Mesh-Free Least Square Based Finite Difference Methods," *Int. J. Numer. Methods Fluids*, 53, pp. 305–332.

[27] Durbin P.A. and Iaccarino G.(2002): An approach to local refinement of structured grids, *J. Comput. Phys.* 181, 639-653.

[28] Eleuterioi F. Toro (2009): *Riemann solvers and numerical methods for fluid dynamics: A Practical Introduction*, Springer.

[29] Fadlun E.A, Verzicco R., Orlandi P, Mohd-Yusof J. (2000): Combined Immersed boundary finite difference methods for three dimensional complex flow simulations, *Journal of Computational Physics* 161: 35-60

[30] Fan, L.F. (2000): Domain-free discretization method and its applications, *Master Thesis*, National University of Singapore.

[31] Farrant T., Tan M., and Price W.G. (2000): A cell boundary element method applied to laminar vortex-shedding from arrays of cylinders in various arrangements, *J. of Fluids and Structures*, 14: 375-402.

[32] Farrant T., Tan M., and Price W.G. (2001): A cell boundary element method applied to laminar vortex shedding from circular cylinders, *Computers and Fluids* 30: 211-236

[33] Fasshauer G.E. (2002) Newton Iteration with Multiquadrics for the solution of nonlinear PDEs. *Comput. Math. Applic.* 43(16):423-438

[34] Fasshauer G.E. (1997) Solving Partial Differential Equations by Collocation with Radial Basis Functions: Surface Fitting and Multiresolution Methods, Vanderbilt University Press, 131-138.

[35]Fedikw R.P., Aslam T., Merriman B., Osher S. (1999): A Non-oscillatory Eulerian Approach to interfaces in Multimaterial Flows (the Ghost Fluid Method), *Journal of Computational Physics* 152, 457-492.

[36]Feng Z.G. and Michaelides E.E. (2005): Proteus: a direct forcing method in the simulations of particulate flow, *Journal of Computational Physics*, 202: 20

[37]Ferziger J.H. and Peric M (2002): *Computational Methods for Fluid Dynamics*, Springer-Verlag, Berlin.

[38] Foley J.D.(1996): *Computer Graphics: Principles and Practices, 2nd Ed.in C*, Addison-Wesley Systems Programming Series.

[39] Fromm J. E. (1968): A method for reducing dispersion in convective difference schemes, *J. Comput. Phys.* 3:176 – 189.

[40]Ghaddar N.K.(1992): Natural convection heat transfer between a uniformly heated cylindrical element and its rectangular enclosure, *Int.J.Heat Mass Transfer* 35(10): 2327-2334

[41] Gilmanov A, Sotiropoulos F, and Balarasf E (2003): A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids, *J.Comp. Phys.* 191, 660-669.

[42]Godunov S.K. (1959): Finite difference methods for numerical computation of discontinuous solutions of the equations of fluid dynamics, *Mat. Sborn*. 47, 271

[43]Goldstein D., Hadler R., Sirovich L. (1993): Modeling a no-slip flow boundary with an external force field, *J. Comput. Phys*.105:354-378

[44] Guilmineau E. and Queutey P. (2002): A numerical simulation of vortex shedding from an oscillating circular cylinder, *Journal of Fluids and Structures*, 16, 773-794.

[45] Guj G. and Stella F., (1995): Natural convection in horizontal eccentric annuli: Numerical study, *Numerical Heat Transfer,Part A,* 27:89-105

[46] He X.Y. and Doolen G.D. (1997): Lattice Boltzmann method on a curvilinear coordinate system: Vortex shedding behind a circular cylinder, *Physical Review*, 56:434-440.

[47] Hirsch C.(1990): *Numerical Computation of Internal and External Flows,* , John Wiley & Sons.

[48] Imamura T., Suzuki K., Nakamura T., Yoshida M.,(2004): Flow simulation around an airfoil using lattice Boltzmann method on generalized coordinates, *AIAA 2004-0244*.

[49] Jameson A, Schmidt W, Turkel E (1981): Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes, *AIAA paper 81-1259*.

[50] Ji H, Lien FS, Yee E (2008): A robust and efficient hybrid cut-cell/ghost-cell method with adaptive mesh refinement for moving boundaries on irregular domains, *Comput. Methods Appl. Mech. Engrg*. 198 : 432 – 448

[51] Johnson TA and Patel VC (1999): Flow past a sphere up to a Reynolds number of 300, *J.Fluid Mech.* 378, 19-70.

[52] Johnson AA and Tezduyar TE (1994): Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces, *Comput .Meth. Appl. Mech. Eng.* **119**, 73-94.

[53] Jordan, F., Fromm, J. (1972): Oscillatory drag, lift and torque on a circular cylinder in a uniform flow. *Physics of Fluids* 15, 371.

[54] Karniadakis G.E. and Triantafyllou G.S. ,(1989), 'Frequency selection and asymptotic states in laminar wakes', *J. Fluid Mech*., 199, 441–469

[55] Khokhlov A.M. (1998): Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations, *J. Comput. Phys.* 143, 519-543.

[56] Kim J., Kim D., and Choi H.(2001): An immersed-boundary finite-volume method for simulations of flow in complex geometries, *J.Comput.Phys.* 171:132-150.

[57] Kirshman D.J. and Liu F. (2002): Gridless boundary condition treatment for a non-body-conforming mesh, *AIAA paper 2002-3285*.

[58] Kuehn T.H. and Goldstein R.J (1976): An experimental and theoretical study of natural convection in the annulus between horizontal concentric cylinders, *J. Fluid Mech.,* 74:695-716

[59] Kuehn T.H. and Goldstein R.J.(1978): An experimental study of natural convection in the concentric and eccentric horizontal cylindrical annuli, *ASME J.Heat Transfer*, 100:635-640.

[60] Ladson, C. L., Hill, A. S., and Johnson, Jr., W. G. (1987): "Pressure Distributions from High Reynolds Number Transonic Tests of an NACA 0012 Airfoil in the Langley 0.3-Meter Transonic Cryogenic Tunnel," *NASA TM 100526,* December 1987

[61]Lai M., Peskin C.S. (2000): An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, *J.Comput.Phys.* 160:705-734

[62] Lax P. D. and Wendroff B. (1960): Systems of Conservation Laws. *Comm. Pure Appl.Math.*, 13:217 – 237,.

[63] Lee K, Yang K.S., and Yoon D.H. (2009) "Flow-induced forces on two circular cylinders in proximity", *Computers & Fluids* 38: 111-120

[64] Leung M.Y.Y., Hung Yen Hui, and King, I.(1996): Facial expression synthesis by radial basis function network and image warping, *ICNN 96. The 1996 IEEE International Conference on Neural Networks ,*Vol 3: 1400-1405

[65] LeVeque R.J. (1992): Numerical methods for conservation laws, Birkhauser.

[66] Leveque RJ, Li Z. (1994): The immersed interface method for elliptic equations with discontinuous coefficients and singular sources.*SIAM J. Numer. Anal.* 31:1001–25

[67] Lin W. L., Carlson K. D., and Chen C. J. (1998), Diagonal Cartesian Method for Numerical Simulation of Incompressible Flow over Complex Boundary, *Numer. Heat Transfer A*, 33 (5): 181-213.

[68] Linnick M.N., Fasel H.F. (2005): A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, *J.Comput.Phys.* 204: 157-128

[69] Liou, M.-S. and Steffen, C. (1993): "A New Flux Splitting Scheme," *J. Comput. Phys.,* Vol. 107, 23-39.

[70] Liu C, Zheng X.,Sung C.H. (1998), Preconditioned multigrid metrhods for unsteady incompressible flows, *J.Comput.Phys.* 139

[71] Liu J., Zhao N., and Hu O.(2009): The ghost cell method and its applications for inviscid compressible flow on adaptive tree Cartesian grids, Advances in Applied Mathematics and Mechanics, 1 (5): 664-682.

[72] Liu Y, Nan, P.T, Kemp R.(1996): Coupled conduction-convection problem for a cylinder in an enclosure, *Comput. Mech*. 18:429-443.

[73] Liu Y., So R.M.C., Cui Z.X.(2006): Bluff body flow simulation using lattice Boltzmann equation with multiple relaxation time, *Computers.&Fluids*, 35:951-956

[74] Lu X. Y. and Dalton C., (1996): Calculation of the timing of vortex formation from an oscillating cylinder, *Journal of Fluids and Structures*, 10,527-541

[75] MacCormack R.W. (1969):The effect of viscosity in hypervelocity impact cratering, *AIAA Paper 69-354*.

[76] Madych W.R. and Nelson S.A. (1989), Multivariate interpolation and conditionally positive definite function, *Approx. Theory Appl*. 4:77-89

[77] Menhart M., Wengle H. (1994): Large eddy simulation of turbulent boundary layer over a hemishpere, In P.Voke, L.Kleiser, J.P. Choolet (eds), *Proc. 1st ERCOFTAC Workshop on Direct and Large Eddy Simulatin*, 299-310, Kluwer Acedemic Publishers, Dordrecht

[78] Mittal Rajat and Iaccarino Gianluca (2005): Immersed Boundary Methods, *Annu. Rev.Fluid Mech*. 37:239-61

[79] Mittal S. (2001): Computation of three-dimensional flows past circular cylinder of low aspect ratio, Physics of Fluids, 13(1): 177-191

[80] Mittal S., Kumar V. and Raghuvanshi A. (1997): Unsteady incompressible flows past two cylinders in tandem and staggered arrangements, *Int. J. numer. Methods Fluids*, 25: 1315-1344.

[81] Mittal S and Tezduyar TE (1994), Massively parallel finite element computation of incompressible flows involving fluid-body interactions, *Comput. Meth. Appl. Mech. Eng.* 112, 253-282.

[82] Morinishi Koji, (1992)"A finite difference solution of the Euler equations on non-body-fitted Cartesian Grid", Computers Fluids Vol.21.No.3, PP: 331-344

[83] Moukalled F. and Acharya S., (1996): Natural Convection in the Annulus Between Concentric Horizontal Circular and Square Cylinders, *Journal of Thermophysics and Heat Transfer*, Vol.10, No.3,July-September:524-531

[84] Niu X. D., Shu C., Chew Y. T. and Peng Y. (2006): A Momentum Exchange-based Immersed Boundary-Lattice Boltzmann Method for Simulating Incompressible Viscous Flows, *Physics Letters A*, 354, 173-182.

[85] Park, J., Kwon, K. and Choi, H. (1998): "Numerical Solutions of Flow Past a Circular Cylinder at Reynolds Number up to 160," *KSME International Journal*, Vol.12, p. 1200.

[86] Pember R.B., Bell J.B., Colella P., Curtchfield W.Y. and Welcome M.L.(1995): An adaptive Cartesian grid method for unsteady compressible flow in irregular regions, *J. Comput. Phys.* 120, 278-304

[87] Peng Y., Shu C., Chew Y. T., Niu X. D. and Lu X. Y. (2006): Application of Multi-block Approach in the Immersed Boundary-Lattice Boltzmann Method for Viscous Fluid Flows, *Journal of Computational Physics*, 218: 460-478

[88] Peskin CS. (1972): *Flow patterns around heart valves: a digital computer method for solving the equations of motion.* PhD thesis. Physiol., Albert Einstein Coll. Med., Univ.

[89] Peskin C.S. (1977): Numerical analysis of blood flow in the heart, *J.Comput.Phys*.25: 220

[90] Rippa S. (1999): An algorithm for selecting a good value for the parameter *c* in radial basis function interpolation, *Advances in Computational Mathematics* 11:193-210

[91] Rockwell D. (1998): Vortex-body interactions, Ann.Rev.Fluids Mech. 30:199-229

[92] Roe P.L. (1981): Approximate Riemann solvers, parameter vectors and difference schemes, *Journal of Compu. Physics*, 43:357-372

[93]Russell D, Wang Z.J.(2003): A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow, *Journal of Compu. Physics* 191: 177-205.

[94] Sheard GJ, Hourigan K, and Thompson MC (2005): Computations of the drag coefficients for low-Reynolds-number flow past rings, *J. Fluid Mech.* 526, 257-275, 2005.

[95] Slaouti A. and Gerrard J.H. (1981): " An experimental investigation of the end effects on the wake of a circular cylinder towed through water at low Reynolds numbers", *J. Fluid Mech.* 112, 297

[96] Sheard GJ, Hourigan K, and Thompson MC (2005): Computations of the drag coefficients for low-Reynolds-number flow past rings, *J. Fluid Mech*. 526:257-275.

97] Shu, C. (2000): *Differential quadrature and its application in engineering*, Springer-Verlag, London.

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

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

[100] Shu C. and Wu Y. L. (2002), Domain-free discretization method for doubly connected domain and its application to simulate natural convection in eccentric annuli, *Comput. Methods Appl. Mech. Engrg.,* **191**, 1827–1841.

[101] Shu C., Xue H., and Zhu Y.D. (2001): Numerical study of natural convection in an eccentric annulus between a square outer cylinder and a circular inner cylinder using DQ method, *International Journal of Heat and Mass Transfer*, 44: 3321-3333

[102] Slaouti A. and Gerrard J. H. (1981): "An experimental investigation of the effects on the wake of a circular cylinder towed through the water at low Reynolds numbers," *J. Fluid Mech.* 112, 297

[103] So R.M.C., Liu Y., Cui Z.X., Zhang C.H., Wang X.Q. (2005): Three dimensional wake effects on flow-induced forces, *Journal of Fluids and Structures* 20:373-402

[104] Strikewerda J.C.(2004): *Finite Difference Schemes and Partial Differential Equations*, SIAM, Philadelphia

[105] Tang H.S., Jones S C., Sotiropoulos F.(2003): An overset-grid method for 3D unsteady incompressible flows, *J. Comput. Phys. 191* :567-600

[106] Tezduyar T.E., Behr M. and Liou J. (1992): A new strategy for finite element computations involving moving boundaries and interfaces- The deforming-spatial-domain /space-time procedure: I.The concept and the preliminary numerical tests, *Computer Methods in Applied Mechanics and Engineering* 94, 353.

[107] Tseng Y.H., Ferziger J.H.(2003): A ghost-cell immersed boundary method for flow in complex geometry, *J. Comput. Phys. 192* :593.

[108] Tuann S.Y., Olson M.D. (1978): Numerical studies of the flow around a circular cylinder by a finite element method, *Comput. Fluid*, 6: 219

[109] Udaykumar HS, Mittal R, Rampunggoon P,Khanna A.(2001): A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. *J. Comput. Phys.*174:345–80

[110] Uhlmann M. (2005): An immersed boundary method with direct forcing for the simulation of particulate flows, *J. Comput. Phys.*209:448-476.

[111] Warming R. F. and Beam R. W. (1976): Upwind Second Order Difference Schemes with Applications i n Aerodynamic Flows. *AIAA Journal*, 24:1241－1249.

[112] White F.W. (1974): Viscous Fluid Flow, *McGraw-Hill*, New York.

[113] Williamson C.H.K. (1985): "Evolution of a single wake behind a pair of bluff bodies", *J.Fluid Mech.* 159, 1-18

[114] Williamson C.H.K. (1989):"Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers", *J.Fluid Mech*.206,579

[115] Williamson,C.H.K. (1996): Three dimensional wake transition. *Journal of Fluid Mechanics* 328:345–407.

[116] Williamson C.H.K. (1996) "Vortex dynamics in the cylinder wake," *Annu. Rev.Fluid Mech*. 28, 477

[117] Wu J. (2010): Development of a novel immersed boundary-lattice boltzmann method and its applications, Ph.D thesis, National University of Singapore.

[118] Wu J., Shu C. (2009): Implicit velocity correction-based immersed boundary-Lattice Boltzmann method and its applications, *J. Comput. Phys.* 228: 1963-1979

[119] Wu J., Shu C. (2010): An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows, *J. Comput. Phys.* 229: 5022-5042

[120] Wu Y.L., Shu C.(2002): Development of RBF-DQ method for derivative approximation and its application to simulate natural convection in concentric annuli, *Computational Mechanics* 29: 477-485.

[121]Wu Y.L., Shu C., Chen H.Q. and Zhao N. (2004): Radial Basis Function-Enhanced Domain-Free Discretization method and its Applications, *Numerical Heat Transfer, Part B*, 46:269-282

[122] Wu Y.L., Shu C., Qiu J, Tani J.(2003): Implementing Multi-Grid approach in Domain-Free Discretization method to speed up convergence, *Computer Methods in Applied Mechanics and Engineering*, 192: 2425-2438

[123] Xu S., Wang Z.J. (2006): An immersed interface method for simulating the interaction of a fluid with moving boundaries. *Journal of Computational Physics* 216: 454-493

124] Ye T, Mittal R, Udaykumar HS, Shyy W.(1999): An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *J. Comput. Phys.*156:209–40

[125] Young D.L., Huang J.L., Eldho T.I. (2001): Simulation of laminar vortex shedding flow past cylinders using a coupled BEM and FEM model, *Computer Methods in Applied Mechanics and Engineering*, 190: 5975-5998

[126] Zdravkovich M.M. (1977): "Review of flow interference between two circular cylinders in various arrangements", *ASME Journal of fluids engineering* 99: 618-633

[127] Zdravkovich M.M. (1985): Flow induced oscillations of two interfering circular cylinders, *J. Sound Vib.* 101: 511-521

[128] Zdravkovich M.M. (1997): "Flow around circular cylinders", Vol.1, Oxford University Press, New York.

[129] Zhang J. and Dalton C.(1998): A three-dimensional simulation of a steady approach flow past a circular cylinder at low Reynolds number, *Int. J. Numer. Meth. Fluids* 26: 1003-1022

[130] Zhou C.H, Shu C and Wu Y.Z.(2007): Extension of Domain-Free Discretization method to simulate compressible flows over fixed and moving bodies . *Int. J. Numer. Meth. Fluids*. 53: 175-199.

[131]Zienkiewicz O.C.(1977): *The finite element method*, McGraw-Hill, New York

List of Journal Papers based on the thesis

1. C.Shu, **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*,51: 897-912, 2006
2. **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,* 58 (11):1223-1236, 2008
3. **Y.L.Wu**, C.Shu, and H.Ding: 'Simulation of Incompressible Viscous Flows by Local DFD-Immersed Boundary Method', *Advances in Applied Mathematics and Mechanics*, 4 (3): 311-324,2012