Doctor of Philosophy

Cartesian Mesh-based Incompressible Flow Simulation over Arbitrarily Complex Geometries using a Multigrid Method

The Graduate School
of the University of Ulsan

School of Naval Architecture and
Ocean Engineering

Go, Gwangsoo

Cartesian Mesh-based Incompressible Flow Simulation over Arbitrarily Complex Geometries using a Multigrid Method

Supervisor : Hyung Taek Ahn

A Dissertation

Submitted to
the Graduate School of the University of Ulsan
In partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Go, Gwangsoo

School of Naval Architecture and Ocean Engineering
Ulsan, Korea
August 2019

Cartesian Mesh-based Incompressible Flow Simulation over Arbitrarily Complex Geometries using a Multigrid Method

This certifies that the dissertation of Gwangsoo Go is approved.

[illegible]

Committee Chair Dr. Yoon Rak Choi

[illegible]

Committee Member Dr. Hyoung-Seock Seo

[illegible]

Committee Member Dr. Jung-Il Choi

[illegible]

Committee Member Dr. Jae Hwa Lee

[illegible]

Committee Member Dr. Hyung Taek Ahn

School of Naval Architecture and Ocean Engineering
Ulsan, Korea
August 2019

Acknowledgement

I would like to express my gratitude to those who supported me throughout my Ph.D course. I would not have been able to finish Ph.D course without their help.

I would like to express my sincere appreciation to my advisor, Prof. Hyung Taek Ahn, for his advices, comments, patience, and encouragement. His guidance helped me in all the time of research and writing of this dissertation. I could not have imagined having a better advisor and mentor for my Ph.D study. Besides my advisor, I would like to thank the rest of my committee: Prof. Yoon Rak Choi, Prof. Hyoung-Seock Seo, Prof. Jung-Il Choi, and Prof. Jae Hwa Lee for their insightful comments and encouragement. I also would like to thank the my family, parents and brother, for supporting me spiritually throughout the Ph.D course and my friends of Bldg. 42 of UOU for their friendship and assistance.

My research for this dissertation was supported by a National Research Foundation of Korea (NRF) Grant funded by the Korean Government (NRF-2017-Global Ph.D. Fellowship Program), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2013R1A1A2007924), and the National Supercomputing Center with supercomputing resources including technical support (KSC-2018-CHA-0006).

Abstract

In this dissertation, two algorithms for efficient Cartesian mesh-based flow simulations over arbitrarily complex objects are presented. As a first category of this dissertation, a geometric multigrid (MG) algorithm using the Heaviside function restriction is presented. This algorithm is simple to implement, readily parallelizable, and can be applied to any irregular domain problem. The validity of the presented algorithm is demonstrated by solving an analytically defined Poisson problem on an irregular domain. Furthermore, the optimal performance of the MG method is also demonstrated herein. As a second category of this dissertation, a novel efficient and reliable determination procedure of the signed distance function (SDF) based on an adaptive mesh refinement (AMR) strategy is described. Our motivation is that the SDF near the solid boundary is accurately required in the fluid simulation, whereas the SDF of the rest region can be considered as arbitrarily large values if the sign is correct. We employ the AMR procedure in order to efficiently define interface cells containing the solid boundary. By focusing on interface cells in hierarchical grids, the number of operations for computing the SDF can be reduced significantly. The efficiency and accuracy of the proposed method is demonstrated by volume convergence tests.

By combining above two algorithms with an existing solution algorithm for the fluid, an in-house Cartesian mesh-based incompressible flow solver is developed. The hybrid MPI/OpenMP parallelization is applied to this solver, and parallel computation is conducted on the KISTI`s *Nurion* supercomputer. To demonstrate applicability of the current approach, various well-known benchmark problems in both 2D and 3D are simulated, and all results are validated with previous experimental and numerical data. Furthermore, as the most challenging test case, simulations for flow over an underwater walking robot, namely Crabster, are presented. Finally, golf ball wake simulations with and without back-spin are presented as a very large-scale problem, which consists of ten-billions order cells.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Background, motivation, and objective

Flow simulation over a complex geometry could be classified into two major categories depending on techniques of the body representation, namely the method based on body-fitted meshes, either structured [9, 41] or unstructured meshes [1, 2, 53, 66], and the one based on Cartesian meshes with a special method for representing the existence of the body within the domain, such as immersed/embedded boundary method [76, 67, 29, 68, 28, 97, 52, 98, 14, 17, 21, 33, 72, 36, 37, 22, 34, 35, 43, 58, 62, 63, 78, 106, 107]. Compared to the immersed/embedded boundary approach on Cartesian meshes, the body-fitted mesh approach is a more intuitive in terms of body representation, boundary condition imposition, and resolving boundary layer. The Cartesian mesh-based approach with immersed/ embedded boundary method has been regaining popularity with introduction of massively parallel computing architecture.

There are two main advantages of the Cartesian mesh-based method with immersed/embedded body compared to the body-fitted ones. First, the Cartesian mesh-based method does not require a complicated mesh generation process prior to the actual computation. Instead, because generally Cartesian grids are not fitted to a geometry interface, it rather requires determination of a simple marker function, which is usually a signed distance function (SDF) [75, 38, 21, 25, 84, 36] or Heaviside function [33, 36] with respect to a given geometry. Second, the method is efficient in the linear solver stage, which is the most time consuming part of the fluid simulation, as it can employ an optimally fast iterative linear solver such as geometric multigrid (MG) [13, 16, 26, 101, 89, 48, 59, 36], which is readily applicable by utilizing the structured nature of the mesh connectivity of the Cartesian mesh system.

Both of previously mentioned processes, namely *a-priori* SDF computation and a fast-iterative linear solver, ironically become potential obstacles against the wide application of the Cartesian mesh-based method for more challenging flow simulations involving complex geometries. This dissertation focuses on these two contents.

1.1.1. Multigrid method on an irregular domain

The current solution algorithm for incompressible Navier-Stokes equations is based on the projection method, originally introduced by Chorin [18]. The projection method involves the procedure to solve the pressure Poisson equation, where most of the CPU time associated with the flow simulation is spent. The success of the projection method heavily depends on the efficiency of the Poisson solver. Due to the regular connectivity of the Cartesian mesh, various fast iterative solvers can be applied to the Poisson solver stage, including Krylov subspace methods [51] with pre-conditioning techniques and/or MG method [13, 16, 26, 101, 89, 48, 59, 36].

It is well known that the Poisson-type equations on a regular domain (for example a rectangular domain without any obstacles) can be most efficiently solved by a geometric MG with the fixed number of cycles regardless of mesh size or resolution [13, 16, 26, 101]. This property is the unique feature of the MG method. When the MG algorithm is optimally constructed, the total computational cost of the MG solver is $\mathcal{O}(N)$, where N is the number of unknowns or N_{cells} for cell-based schemes in the discrete Poisson problem. It is natural that the total computation cost of the optimal MG solver grows linearly with respect to N because the cost of the single MG cycle increases linearly with respect to N.

However, if irregular domain problems are considered, the application of the MG method becomes rather unclear. To achieve the optimal performance of the MG method for irregular domain problems, the irregularity of the domain shape has to be effectively transferred down to successively coarsened grids, and the restriction and prolongation algorithms have to be consistently constructed. Although there have been a few successful previous studies applying the MG method to irregular domain problems [98,110,113,116], the actual implementation is rather complicated [98, 110, 113] and the optimal performance of the $\mathcal{O}(N)$ complexity has not been demonstrated clearly [116].

The first objective of this paper is to present an easy-to-implement geometric MG algorithm for solving the Poisson equation on irregular domains. The key idea of the current MG algorithm is to use the efficient *edge/face-wise Heaviside function restriction* in order to effectively represent the domain irregularities on successively coarsened grids. It should be noted that a similar idea of the edge/face-wise restriction concept was also presented by Weber et al. [116]. However, we employ the Heaviside function [33], which is the well-designed edge/face-wise function that can efficiently represent domain irregularities, and their actual implementation is different from ours.

In the current study, the MG algorithm is more simply implemented by using the recursive function, not iteratively. Furthermore, we clearly demonstrate the optimal performance of the MG method in the irregular domain problem. By using the Heaviside function restriction, the detailed geometric information on the finest grid is effectively transferred down to the coarsest grid without any additional effort, and this makes the actual implementation of the MG method become trivially simple. By using the multilevel algorithm, the presented MG algorithm is applied to parallel computation. The scalability of the current MG method is demonstrated on the many-core CPU architecture using the grid resolution of $\mathcal{O}(10^9)$ cells.

1.1.2. Signed distance function computation

A special indicator function has to be pre-computed prior to the actual flow simulation to represent a complex shaped body within a Cartesian domain. Unless the body shape is represented by a simple analytical formula, such as a circle, ellipse, polygon, or any combination of those in higher spatial dimensions, the marker function computation involves distance computation and complicated in/out tests to determine the sign with respect to given geometric information. For most engineering design processes, the final body shape is represented by an explicit surface definition from a computer-aided-design (CAD) model. Perhaps the most popular definition of the body surface would be a stereolithography (STL) file format, which is a directed triangular surface mesh. After considering all these processes, Cartesian mesh-based flow simulation around complex engineering objects involves SDF computations with respect to an arbitrary body shape, and this process perhaps corresponds to the stages of the volume mesh generation in body-fitted mesh methods.

The second objective of this dissertation is to introduce a novel efficient, accurate, and robust computation method for the SDF in the Cartesian domain by using the adaptive mesh refinement (AMR) [54, 56, 7, 36] strategy. The AMR technique is utilized to optimize the number of operations for the SDF. The ray-casting algorithm [85, 43, 36], which is known to be the most accurate and reliable algorithm of the in/out test, is employed to determine the sign of the distance function.

The reasons that most of the Cartesian mesh-based methods have been applied to simple geometries could be due to the lack of efficient and reliable algorithms for the SDF computation. Thus, the method is limited to simple shaped-body simulations in which complex SDF computation can be avoided. Nevertheless, there are a few recent algorithms that aim to extend SDF computations for complex body shapes. The representative example is the one based on the pseudo-normal vector estimation [15, 21].

The pseudo-normal approach can be successfully applied to a relatively complex moving body [21]. However, to the best of the authors' knowledge, the algorithm is not guaranteed to deliver the correct results for arbitrarily complex geometries. In fact, it is mentioned that, in the representative reference of the pseudo-normal applications [21], the algorithm could fail to resolve small features such as gaps and holes. To guarantee the applicability of the Cartesian mesh-based immersed/embedded body method, the SDF computation process should be improved to apply to an any arbitrary complex geometry.

Here, a novel AMR-based SDF computation method with the ray-casting algorithm is presented to overcome the limitations of the previous algorithms. This new AMR-based approach presented in this paper can guarantee the exact SDF computation regardless of the complexity of the geometry by using the intelligent refinement criterion, which ensures that any small and sharp feature of the object is accurately resolved. Furthermore, current algorithm is an order of magnitude faster than a naive SDF computation on the uniform mesh, thus the number of operations in an AMR-based method is $O\left(n^{d-1}\right)$ where n is the number of cells along a spatial direction, and d is the number of spatial dimensions.

As an extension research of the AMR-based SDF computation algorithm toward flow simulation over a moving body, an efficient and accurate SDF transformation strategy is presented. Compared to the body-fitted mesh approach, the Cartesian mesh method also has an advantage for flow simulation over a moving body in terms of a greater efficiency of handling a moving body without any mesh deformation. However, to take this advantage, it is essential that the time-varying SDF of a moving body must be update as efficiently as possible.

There are several trials in which rigid body motion is successfully simulated by the Cartesian mesh-based immersed boundary method [14, 17, 21, 22, 31, 58, 62, 63, 98, 106, 107]. In most of these cases, the geometry of the moving body is simple enough such that the time-varying SDF can be updated instantly by an analytical expression, such as those involving a cylinder or a sphere. For more complex geometries, the SDFs are re-evaluated based on the location of the body at each time step [21]. There is a nice trial for avoiding the redundant computation of the SDF for a rigid body and slender deformable body although their SDF representations are explained by bilinear interpolation based on a regular Cartesian mesh [63].

In this dissertation, an efficient and accurate moving body representation technique which is essential for flow simulations over a rigid moving-body is presented. The motivation of the idea is that an accurate computation of the initial SDF would suffice for representation of the body, and any additional SDF computation should be avoided regardless of the body motion. This objective can be accomplished by an efficient transformation of the SDF onto the arbitrarily moving frame attached to the body. Based on the accurate determination of the SDF around a moving body, the new SDF can be simply transformed from the initial SDF field according to the mechanics of rigid body motion. In other words, the timely update of the SDF field is nothing but a combined rotational and translational motion of the originally "frozen" SDF field.

1.1.3. Fluid/rigid-body coupling method

The existence of a body inside of the flow field has to be properly represented in a manner consistent with the solution algorithm for the governing equations, namely the momentum and continuity equations. The presence of the body should be appropriately reflected to the Navier-Stokes equations to be imposed with correct boundary condition, i.e. the no-slip velocity condition for the momentum equations and the Neumann pressure condition for the Poisson equation which is the result of the continuity equation. These two boundary conditions must be properly implemented to correctly represent the body with the Cartesian meshes.

There are several classes of techniques that can implement those boundary conditions. The immersed boundary method originally proposed by Peskin [76] is perhaps the most popular approach to represent an irregular body shape on a Cartesian mesh. The key idea of the immersed boundary method is enforcing the desired boundary condition for the velocity indirectly using forcing functions. Depending on types of forcing functions, various immersed boundary methods [29, 68, 28, 52, 97, 78] have been reported. However, among these studies, there are no straightforward ways to satisfy the Neumann boundary condition of a pressure Poisson equation in the projection stage.

Ng et al. [72] introduced a novel discretization of the projection method on an irregular domain to enforce the Neumann condition in the pressure Poisson equation. Then, Gibou and Min [33] proposed a more generalized form by introducing the Heaviside function, which can be directly computed from the SDF with respect to a given body shape. In their study, the pressure Poisson equation on the irregular domain is represented by this Heaviside function, which is 0 in the body region and 1 in the fluid region.

Heaviside functions can be readily sampled at cell edges/faces and approximated by the length fraction in 2D and area fraction in 3D using SDF. The pressure Poisson equation on the irregular domain with Neumann boundary condition can be straightforwardly derived by multiplying the original equation of the projection step by the Heaviside function. The algorithm presented in Gibou and Min [33] is employed in this study to impose the two types of boundary conditions, i.e. the Dirichlet condition for the velocity and Neumann condition for the pressure, in a manner consistent with equation discretization.

1.2. Major contributions

As the main contributions of the current research, two algorithms for efficient Cartesian mesh-based incompressible flow simulations over complex geometries are introduced towards solving actual engineering problems and described as follows.

1.2.1. An easy-to-implement multigrid method for an irregular domain

In the current study, the MG method is directly applied to an irregular domain problem, and the original contributions can be summarized as follows:

1. Application of the MG method to an irregular domain problem using Heaviside function restriction

2. Verification of the optimal performance for the MG method on an irregular domain problem

It is well known that the Poisson-type equations on a regular domain can be most efficiently solved by the MG method with the fixed number of cycles, namely the optimal performance, regardless of mesh size or resolution. For an irregular domain problem, however, it is rather unclear what is the optimal iterative linear solver for the Poisson problem. In the current research, a direct application of the MG solver for the arbitrarily irregular domain using Heaviside function restriction is introduced, and its optimal performance is demonstrated by solving the analytically defined projection problem.

1.2.2. An efficient and accurate SDF computation

In the part of signed distance function (SDF) computation, the original contributions of the current research can be listed as follows:

1. First application of a ray-casting algorithm on an adaptively refined mesh system

2. First introduction of a mesh refinement criterion based on geometric information

3. An efficient and accurate SDF transformation for representing moving bodies

The first category of the contributions in the part of SDF computation is to develop a novel efficient and robust computation method for the SDF in the Cartesian domain by combining a ray-casting algorithm and the AMR strategy. The AMR technique is utilized to optimize the number of operations for the SDF computation by tracing interface cells. The ray-casting algorithm, which is known to be the most accurate and reliable algorithm of the in/out test, is employed to determine the sign of the distance function on an adaptively refined mesh system.

As the second category, a mesh refined criterion based on geometric information, namely edge/face-based criterion, is introduced in this dissertation. The AMR technique is utilized to efficiently define the finest interface cells, where accurate sign determination is required, by continuously refining the coarser interface cells. Therefore, a robust refinement criterion which can be detect arbitrarily complex geometries is highly desired. In the edge/face-based method, a cell is refined if an intersection between a cell edge/face and body interface is detected. This criterion guarantees the detection of arbitrarily complex geometries.

An efficient and accurate SDF transformation is the last category of the contributions in the part of SDF computation. The objective is that any additional SDF computation, such as re-initialization of the SDF, should be avoided regardless of the body motion. This objective can be accomplished by a transformation of the SDF using two different frames: one is global frames for the fluid simulation, and the other is local frames attached to bodies for the original SDF. The new SDF field of the next time step can be simply transformed from the original SDF field by coordinate transformation from global to local frame and interpolation.

1.3. Outline of the dissertation

The rest of this dissertation is organized as follows.

In Chapter 2, the efficient and stable projection method for the present incompressible flow simulation on an irregular domain, originally proposed by Gibou and Min [33], is presented. First, semi-Lagrangian approach dealing with non-linear convection terms of incompressible Navier-Stokes equations is described. Next, a standard projection method introduced by Chorin [18] is explained. Then, how to treat an irregular domain in projection method, i.e. velocity Helmholtz equation for the intermediate state and pressure Poisson equation for the projection step, is presented.

In Chapter 3, a detailed description about how the MG method can be applied to the irregular domain is presented together with basic concepts of the MG method. To demonstrate the accuracy and efficiency of the proposed MG algorithm for an irregular domain problem, an analytically defined projection problem is solved using the current MG method. The convergence speed of the current MG solver is compared with a standard CG method at different grid levels, and the optimal performance of the MG method is demonstrated for an irregular domain problem.

In Chapter 4, an efficient and accurate way to compute SDF utilizing AMR and the ray-casting algorithm is presented. First, the mesh refinement algorithm, including the refinement decision making criterion, namely edge/face-based criterion, is introduced. Then, the entire process for the global SDF computation is explained. Verification studies for a circle and sphere are presented to demonstrate the efficiency and validity of the current algorithm. Furthermore, several examples are presented for both 2D and 3D geometries to show applicability of the current algorithm towards engineering problem. Finally, SDF transformation strategy based on high-order interpolation for representing moving bodies on Cartesian meshes is described.

In Chapter 5, strategies for the parallel computation using hybrid MPI/OpenMP programming are presented. First, basics of MPI and OpenMP are described, and the concept of hybrid programming is introduced. Then, the parallel MG method using the multi-level V-cycle algorithm [74] is described, and specification of *Nurion* supercomputer, where current parallel computation is conducted, is presented. Furthermore, strong and weak scalabilities of the current incompressible flow solver are presented.

In Chapters 6 and 7, in order to show the accuracy, efficiency, and robustness of the current method, various simulation results for both in 2D and 3D domains are presented and discussed. Finally, concluding remarks of the current research and recommended future works are presented in Chapter 8.

Chapter 2

Numerical Methods for the Fluid

2.1. Governing Equation

The governing equations, composed of momentum and continuity equations, for unsteady incompressible flow without body forces can be written as

$$\rho \frac{D\boldsymbol{U}}{Dt} = -\nabla p + \mu \nabla^2 \boldsymbol{U}, \tag{2.1}$$

$$\nabla \cdot \boldsymbol{U} = 0, \tag{2.2}$$

where $\boldsymbol{U} = (u, v, w)$ and p respectively refer to the fluid velocity vector and pressure (the unknowns of the equations) and given data ρ and μ are the fluid density and the fluid viscosity, respectively. The left-hand-side of the momentum equation refers to the acceleration of the fluid, which is expressed with the total derivative D/Dt of the fluid velocity with respect to time.

The total derivative term can be expanded as the nonlinear advection form if the flow field is described in a purely Eulerian frame of reference. One of the major difficulties of the momentum equation comes from the non-linearity of the advection term. Several elegant algorithms have been proposed and successfully applied to solve this nonlinear convection term, for example a flux-difference splitting [80] scheme, a weighted essentially non-oscillatory (WENO) [45] scheme, and a constrained interpolation profile/conservative semi-Lagrangian (CIP-CSL) [103] method. The common aspect of these schemes is about how to efficiently reflect the upstream information of the advection phenomenon.

In this study, the semi-Lagrangian time discretization method [77, 104, 33, 71, 72, 88, 36, 37] is implemented based on a Cartesian grid system. To discretize the acceleration term, the departure points $\boldsymbol{x}_d^n$ of fluid particles arriving at grid points $\boldsymbol{x}^{n+1}$ should track backward along the characteristic lines $\boldsymbol{x}$ where $D\boldsymbol{x}/Dt = \boldsymbol{U}(\boldsymbol{x}, t)$ within a time step. The second-order backward differentiation formula (BDF2) is utilized to discretize the total derivative term of Eq. (2.1). Therefore, momentum equation can be implicitly discretized as follows:

$$\rho\frac{\frac{3}{2}\boldsymbol{U}^{n+1} - 2\boldsymbol{U}_d^n + \frac{1}{2}\boldsymbol{U}_d^{n-1}}{\Delta t} = (-\nabla p + \mu\nabla^2\boldsymbol{U})^{n+1}, \tag{2.3}$$

where variables which have the superscript $n+1$ are defined at grid points, and $\boldsymbol{U}_d^n$ and $\boldsymbol{U}_d^{n-1}$ are fluid velocities of departure points defined at time t^n and t^{n-1}, respectively. Using a second-order Runge-Kutta method, a departure point of time t^n can be computed as

$$\boldsymbol{x}_d^{n+\frac{1}{2}} = \boldsymbol{x}^{n+1} - \frac{\Delta t}{2}\boldsymbol{U}^n(\boldsymbol{x}^{n+1}), \tag{2.4}$$

$$\boldsymbol{x}_d^n = \boldsymbol{x}^{n+1} - \Delta t\boldsymbol{U}^{n+\frac{1}{2}}\left(\boldsymbol{x}_d^{n+\frac{1}{2}}\right). \tag{2.5}$$

Here, $\boldsymbol{U}^{n+1/2}$ is estimated by linear extrapolation using the two previous time steps, i.e. $\boldsymbol{U}^{n+1/2} = 3/2\,\boldsymbol{U}^n - 1/2\,\boldsymbol{U}^{n-1}$. A departure point of time $n-1$ can be similarly computed as

$$\boldsymbol{x}_d^n = \boldsymbol{x}^{n+1} - \Delta t\boldsymbol{U}^n(\boldsymbol{x}^{n+1}), \tag{2.6}$$

$$\boldsymbol{x}_d^{n-1} = \boldsymbol{x}^{n+1} - 2\Delta t\boldsymbol{U}^n(\boldsymbol{x}_d^n). \tag{2.7}$$

Notice that the nonlinearity of the left-hand-side acceleration term is still included and incorporated into the backtracking process.

Generally, the departure point does not coincide with a grid point. In this case, bilinear interpolation (2D) or trilinear interpolation (3D) is employed to determine the values of a departure point. For a 2D unit cell $[0,1]^2$, bilinear interpolation with variable ξ can be defined as

$$\xi(x,y) = \xi(0,0)(1-x)(1-y) + \xi(0,1)(1-x)y + \xi(1,0)x(1-y) + \xi(1,1)xy. \tag{2.8}$$

The beauty of the semi-Lagrangian algorithm is in its superior stability with respect to the time step. Eq. (2.3) is unconditionally stable. This means that the size of time step can be chosen only by the accuracy requirement, not by the stability. In other words, the limitation of CFL number can be ignored, thus a large time step can be utilized as long as it is allowed by the temporal accuracy requirement.

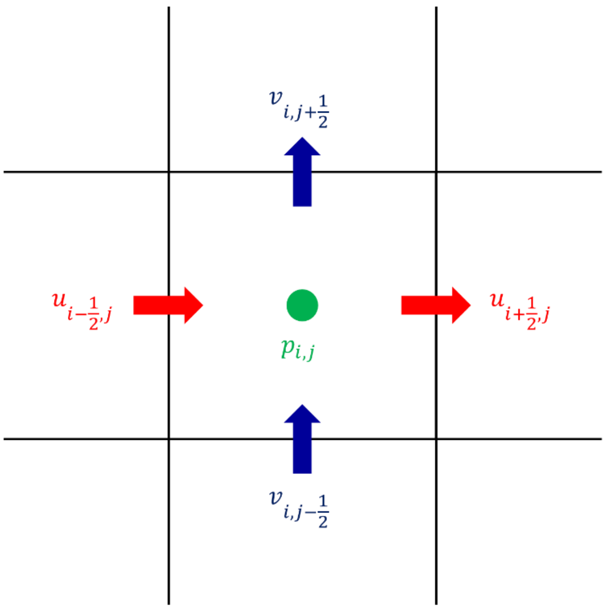


Fig. 2.1. Staggered grid system. In two-spatial dimensions, the horizontal and vertical velocity are respectively sampled at the vertical and horizontal surface, and the pressure is defined at the center of the cell.

A standard projection method introduced by Chorin [18] is employed together with a staggered grid system [39] (see Fig. 2.1) to solve the rest of the implicitly discretized momentum equation. In this grid system, the horizontal velocity ($u_{i\pm1/2,j}$) is positioned at the center of the vertical plane ($\boldsymbol{x}_{i\pm1/2,j}$), while the vertical velocity ($v_{i,j\pm1/2}$) is located at the center of the horizontal plane ($\boldsymbol{x}_{i,j\pm1/2}$), and pressure ($p_{i,j}$) is defined at the center of the grid ($\boldsymbol{x}_{i,j}$). This grid system has an advantage in terms of effectively satisfying the incompressibility.

2.2. Projection Method

In the projection method utilized in this study, a single physical time evolution is decomposed into two sub-stages (see Fig. 2.2): prediction of the intermediate non-solenoidal velocity field and its subsequent projection step down to the divergence-free velocity field. First, the momentum equation at the intermediate state (*) are given by

$$\rho\frac{\frac{3}{2}\boldsymbol{U}^{*}-2\boldsymbol{U}_{d}^{n}+\frac{1}{2}\boldsymbol{U}_{d}^{n-1}}{\Delta t}=\mu\nabla^{2}\boldsymbol{U}^{*}. \tag{2.9}$$

The fluid velocity at intermediate state $\boldsymbol{U}^{*}$ is sought by solving this implicit Helmholtz equation with a constant source term defined by the velocities at previous time steps. The predicted intermediate velocity $\boldsymbol{U}^{*}$ does not satisfy the incompressibility ($\nabla\cdot\boldsymbol{U}=0$), thus it has to be corrected by removing the non-solenoidal contribution by the following projection. The equations for identifying the non-solenoidal contribution can be defined by the pressure Poisson equation as follows:

$$\nabla\cdot\left(\frac{\boldsymbol{U}^{n+1}-\boldsymbol{U}^{*}}{\Delta t}=-\frac{1}{\rho}\nabla q^{n+1}\right)\Rightarrow-\frac{1}{\Delta t}\nabla\cdot\boldsymbol{U}^{*}=-\nabla\cdot\left(\frac{1}{\rho}\nabla q^{n+1}\right), \tag{2.10}$$

where q is the scalar quantity so called *pseudo pressure* to enforce the incompressibility of the fluid, and the pressure is also updated to the next time step by using q. The actual velocity projection step can be defined as

$$\boldsymbol{U}^{n+1}=\boldsymbol{U}^{*}-\frac{\Delta t}{\rho}\nabla q^{n+1}. \tag{2.11}$$

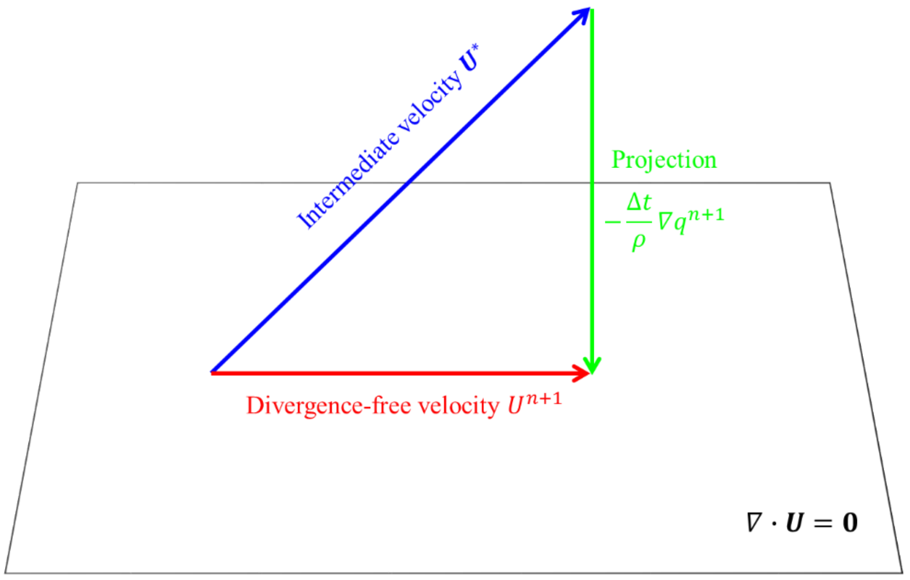

Intermediate velocity $\boldsymbol{U}^*$
Projection
$-\frac{\Delta t}{\rho}\nabla q^{n+1}$
Divergence-free velocity U^{n+1}
$\nabla \cdot \boldsymbol{U} = \boldsymbol{0}$

Fig. 2.2. The concept of the projection method for the incompressible flow

Using this projection process, the intermediate non-solenoidal velocity $\boldsymbol{U}^*$ can be filtered to the divergence-free velocity $\boldsymbol{U}^{n+1}$. If Eq. (2.11) is arranged for $\boldsymbol{U}^*$ and then substituted into Eq. (2.9), it should recover the original momentum equation shown Eq. (2.3). Based on this relation, the pressure, which is in accordance with the divergence-free velocity, can be updated to the time step $n+1$:

$$p^{n+1} = \frac{3}{2}q^{n+1} - \mu\nabla \cdot \boldsymbol{U}^*. \tag{2.12}$$

The previously mentioned procedure for the projection method can only be applied for regular domain problems. If a body domain is submerged in the computational domain, the solution algorithm should be modified to represent an irregular fluid domain excluded by the body. The treatment of the irregular domain problem in the projection method is presented in the following chapters including: 1) velocity Helmholtz equation on an irregular domain; 2) pressure Poisson equation on an irregular domain.

2.3. Velocity Helmholtz equation on an irregular domain

In the intermediate state of an irregular fluid domain problem, $\boldsymbol{U}^*$ should satisfy the no-slip boundary condition provided that the Neumann condition holds for q on the body surface. Therefore, the equation of the intermediate state can be considered as a Helmholtz equation on an irregular domain with a Dirichlet boundary condition. From Eq. (2.9), the Helmholtz equation for the intermediate velocity can be written by

$$\nabla^2 \boldsymbol{U}^* = f(\boldsymbol{U}^*), \tag{2.13}$$

where $\boldsymbol{U}^* = (u^*, v^*, w^*)$ is the intermediate velocity vector and $f(\boldsymbol{U}^*)$ is defined as the left-hand-side of Eq. (2.9) multiplied by $1/\mu$.

A symmetric discretization method for the Poisson equation on an irregular domain proposed by Gibou et al. [30] is employed to solve Eq. (2.13) along with the additional diagonal contribution from the unsteady source, i.e. the leading term of the left-hand-side of Eq. (2.9). This discretization is performed in a dimension by dimension fashion. Thus, only a one-dimensional case is described in this paper without loss of generality. The discretization of other spatial directions is straightforward and can be expressed simply by changing the index in accordance with the direction of interest.

Using the central difference approximation, the one-dimensional Helmholtz equation $\nabla^2 u^* = f(u^*)$ can be discretized as

$$\frac{\left(\dfrac{u^*_{i+3/2} - u^*_{i+1/2}}{\Delta x}\right) - \left(\dfrac{u^*_{i+1/2} - u^*_{i-1/2}}{\Delta x}\right)}{\Delta x} = f_{i+1/2}(u^*), \tag{2.14}$$

where u^* is the x-directional intermediate velocity. If the interface is located between cell boundary $i + 1/2$ and $i + 3/2$ (see Fig. 2.3), the special treatment to satisfy the Dirichlet boundary condition at the interface should be applied. The ghost cell approach is used to deal with the irregular domain, and this approach utilizes the linearly extrapolated values as the solution of the ghost cell in order to enforce the Dirichlet boundary condition. As depicted in Fig. 2.3, the extrapolated value can be computed as

$$u^G_{i+3/2} = \frac{\Delta x}{\Delta \tilde{x}}\left(u_b - u^*_{i+1/2}\right) + u^*_{i+1/2}. \tag{2.15}$$

Based on Gibou et al. [30], the one-dimensional Helmholtz equation for the intermediate velocity can be discretized with the Dirichlet boundary condition on the irregular domain:

$$\frac{\left(\frac{u^G_{i+3/2}-u^*_{i+1/2}}{\Delta x}\right)-\left(\frac{u^*_{i+1/2}-u^*_{i-1/2}}{\Delta x}\right)}{\Delta x}$$
$$=\frac{\left(\frac{u_b-u^*_{i+1/2}}{\Delta\tilde{x}}\right)-\left(\frac{u^*_{i+1/2}-u^*_{i-1/2}}{\Delta x}\right)}{\Delta x}=f_{i+1/2}(u^*). \tag{2.16}$$

This discretization still has second-order accuracy and symmetricity. Since the computational overhead for the intermediate velocity prediction is relatively insignificant compared to the following pressure Poisson equation, any iterative method could be utilized. Here, a conjugate gradient method is used to solve this linear system.

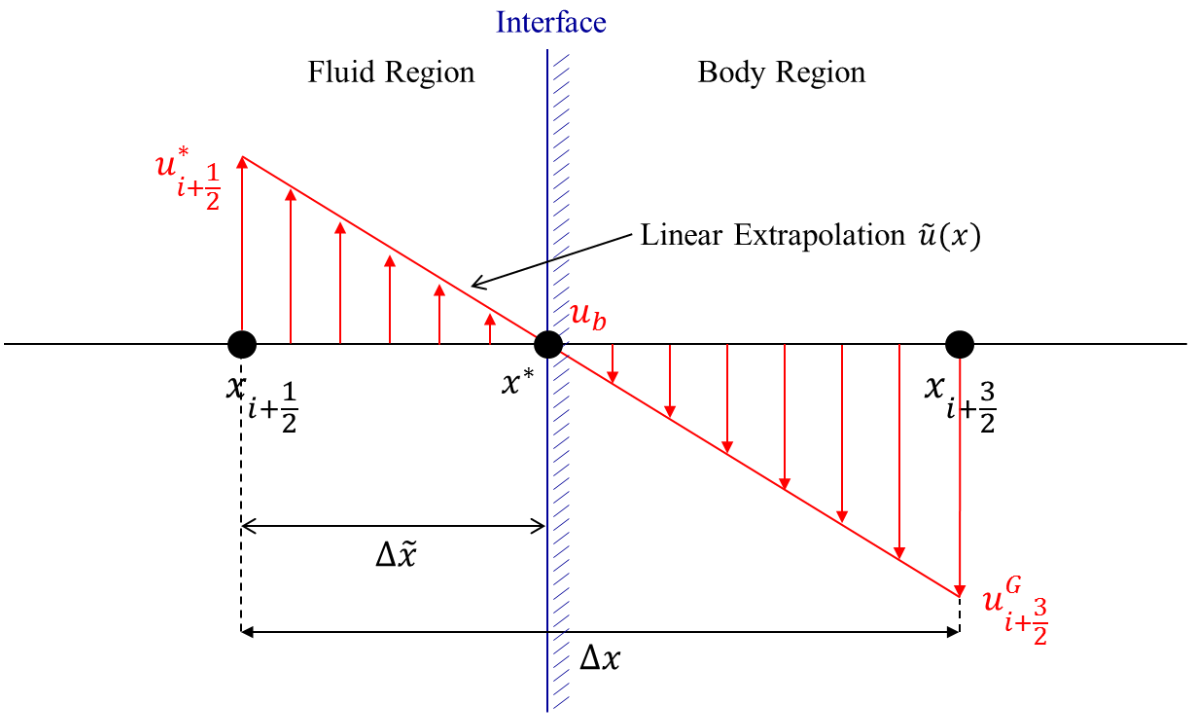


Fig. 2.3. The idea for imposing the Dirichlet boundary condition by assuming the linearly varying velocity across the body interface. $u^G_{i+3/2}$ is the velocity at the interior ghost cell $x_{i+3/2}$ that produced the intended linear velocity profile. u_b indicates a boundary value, and $\tilde{u}(x)$ refers to a linear equation derived by $u^*_{i+1/2}$ and u_b, and $\Delta\tilde{x}$ is the distance between the interface and the real cell point $x_{i+1/2}$.

As illustrated in Fig. 2.3, $\Delta\tilde{x}$ is the distance between the grid point involved in fluid region and the actual solid interface. The location of the solid interface is numerically computed by detecting x^* that satisfies $\phi(x^*) = 0$. Here, ϕ is quadratically reconstructed using the pre-computed SDF near the interface [72]:

$$\phi(x) = \phi(x_{i+1/2}) + \phi_x(x_{i+1/2})x + \frac{1}{2}\phi_{xx}(x_{i+1/2})x^2, \tag{2.17}$$

where ϕ_x and ϕ_{xx} are determined by the central difference approximation at $x_{i+1/2}$. Note that the location of solid interface is approximated quadratically, and the velocity near the solid surface is approximated linearly. Any order elevation for the velocity approximation can be sought, but the current linear approximation of the velocity boundary condition turns out to be consistent with our global solution accuracy, which is indeed second-order.

As illustrated in Fig. 2.3, $\Delta\tilde{x}$ is the distance between the grid point involved in fluid region and the actual solid interface. The location of the solid interface is numerically computed by detecting x^* that satisfies $\phi(x^*) = 0$. Here, ϕ is quadratically reconstructed using the pre-computed SDF near the interface [72]:

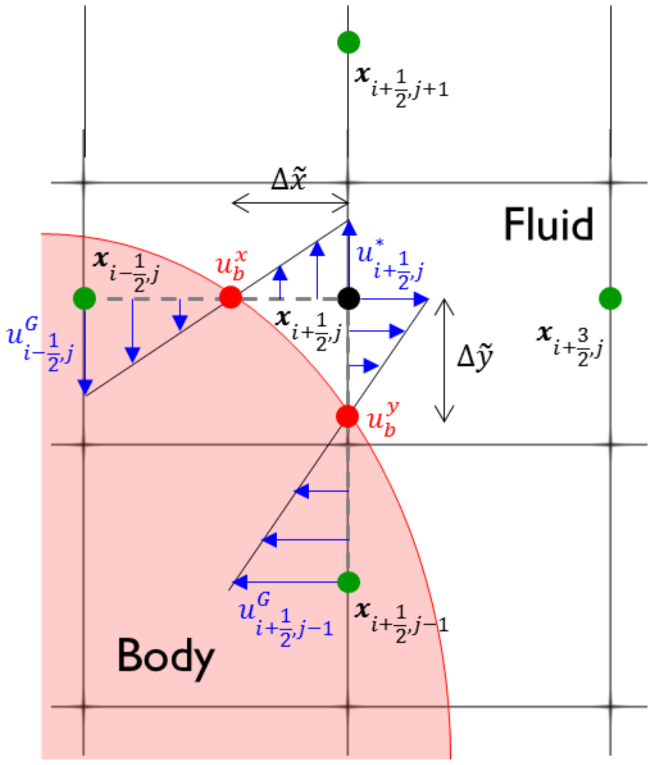

$\boldsymbol{x}_{i+\frac{1}{2},j+1}$
$\Delta\tilde{x}$
Fluid
$\boldsymbol{x}_{i-\frac{1}{2},j}$
u_b^x
$u^*_{i+\frac{1}{2},j}$
$u^G_{i-\frac{1}{2},j}$
$\boldsymbol{x}_{i+\frac{1}{2},j}$
$\Delta\tilde{y}$
$\boldsymbol{x}_{i+\frac{3}{2},j}$
u_b^y
$u^G_{i+\frac{1}{2},j-1}$
$\boldsymbol{x}_{i+\frac{1}{2},j-1}$
Body

Fig. 2.4. Imposition of the Dirichlet boundary condition by assuming the linearly varying velocity across the body interface in the two-spatial dimension.

By using this relation, based on Fig. 2.4., the Helmholtz equation of the x-directional velocity in two-spatial dimension can be discretized as

$$\frac{\rho}{\mu}\left(\frac{\frac{3}{2}u^*_{i+1/2,j} - 2u^n_d + \frac{1}{2}u^{n-1}_d}{\Delta t}\right)$$

$$= \frac{\left(\frac{u^*_{i+3/2,j} - u^*_{i+1/2,j}}{\Delta x}\right) - \left(\frac{u^*_{i+1/2,j} - u^x_b}{\Delta \tilde{x}}\right)}{\Delta x} + \frac{\left(\frac{u^*_{i+1/2,j+1} - u^*_{i+1/2,j}}{\Delta y}\right) - \left(\frac{u^*_{i+1/2,j} - u^y_b}{\Delta \tilde{y}}\right)}{\Delta y} \tag{2.18}$$

2.4. Pressure Poisson equation on an irregular domain

The pressure Poisson equation, actually the equation for q, on an irregular domain is difficult to solve because there are no obvious ways to satisfy the Neumann boundary condition at the interface. A straightforward method of effective enforcement of the Neumann boundary condition was first proposed by Ng et al. [72] and was further generalized by Gibou and Min [33]. The main idea of the Gibou and Min approach is to use the Heaviside function directly for the discretization of the pressure Poisson equation. The Heaviside function is numerically approximated as the length fraction of the cell edge in 2D and the area fraction of the cell face in 3D.

In the case of two-spatial dimensions, see Fig. 2.5, the edge-averaged Heaviside function scaled by the length fraction at a cell interface $x_{i+1/2} \times [y_{i-1/2}, y_{i+1/2}]$ is computed as

$$H_{i+\frac{1}{2},j} = \frac{\phi^+_{i+\frac{1}{2},j+\frac{1}{2}} - \phi^+_{i+\frac{1}{2},j-\frac{1}{2}}}{\phi_{i+\frac{1}{2},j+\frac{1}{2}} - \phi_{i+\frac{1}{2},j-\frac{1}{2}}}, \tag{2.19}$$

where $\phi^+ = \max(\phi, 0)$. If both $\phi_{i+1/2,j+1/2}$ and $\phi_{i+1/2,j-1/2}$ are positive, the Heaviside function is 1. In contrast, if both are negative, then the Heaviside function is 0. If the solid interface intersects the cell edge, then the edge-averaged Heaviside function will be between 0 and 1, i.e. $H_{i+1/2,j} \in (0,1)$. The Heaviside function at horizontal surface can be similarly computed.

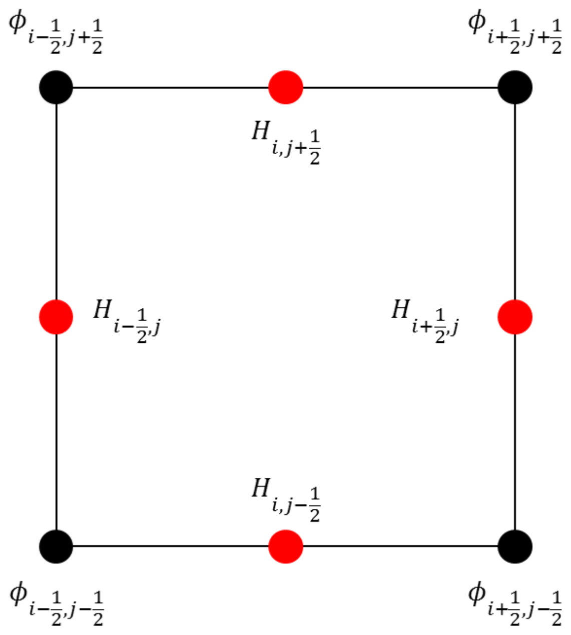

$\phi_{i-\frac{1}{2},j+\frac{1}{2}}$
$\phi_{i+\frac{1}{2},j+\frac{1}{2}}$
$H_{i,j+\frac{1}{2}}$
$H_{i-\frac{1}{2},j}$
$H_{i+\frac{1}{2},j}$
$H_{i,j-\frac{1}{2}}$
$\phi_{i-\frac{1}{2},j-\frac{1}{2}}$
$\phi_{i+\frac{1}{2},j-\frac{1}{2}}$

Fig. 2.5. The layout of the 2D grid cell with the signed distance function and the Heaviside function

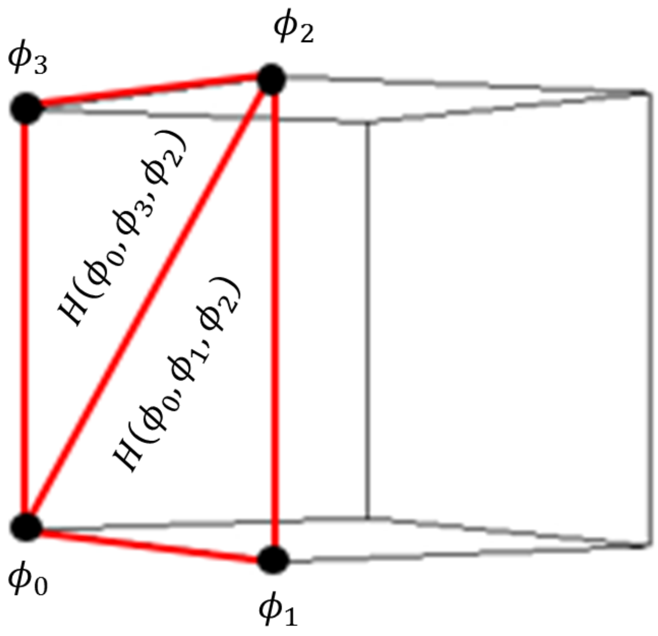


Fig. 2.6. The layout of the 3D grid cell with the signed distance function and the Heaviside function

In the case of three-spatial dimensions, the Heaviside function scaled by the area fraction is defined as the sum of the area fractions of two sub-triangles composing the original cell face. If $\phi_0, \phi_1, \phi_2, \phi_3$ are considered as the SDF of vertices of the cell face, see Fig. 2.6, the area fraction of triangle $\Delta P_0 P_1 P_2$ is computed as

$$H(\phi_0, \phi_1, \phi_2) = \begin{cases} \dfrac{\phi_0^+ - \phi_1^+}{\phi_0 - \phi_1} \cdot \dfrac{\phi_0^+ - \phi_2^+}{\phi_0 - \phi_2} & \text{if } \phi_0 > 0, \phi_1 < 0 \text{ ,and } \phi_2 < 0, \\ 1 - \dfrac{\phi_0^- - \phi_1^-}{\phi_0 - \phi_1} \cdot \dfrac{\phi_0^- - \phi_2^-}{\phi_0 - \phi_2} & \text{if } \phi_0 < 0, \phi_1 > 0 \text{ ,and } \phi_2 > 0, \end{cases} \tag{2.20}$$

where $\phi^- = \min(\phi, 0)$. Note that there were typographical errors in the original formula [33] for the 3D Heaviside function, and those are corrected in Eq. (2.20). Likewise, the Heaviside functions for the rest of the sub-triangle can be calculated by symmetry. Finally, the Heaviside function of the cell surface is defined simply by averaging as follows:

$$H(\phi_0, \phi_1, \phi_2, \phi_3) = \frac{1}{2}\big(H(\phi_0, \phi_1, \phi_2) + H(\phi_0, \phi_3, \phi_2)\big). \tag{2.21}$$

As described in Gibou and Min [33], the previously mentioned approximation of the Heaviside function in two-spatial dimensions produces about sub-second-order ($\cong 1.5$) accuracy. However, discretization in three-spatial dimension has second-order accuracy.

Using the Heaviside function, the pressure Poisson equation on the irregular domain with the Neumann boundary condition can be derived by

$$\nabla \cdot \left\{ H \left(\frac{\boldsymbol{U}^{n+1} - \boldsymbol{U}^*}{\Delta t} = -\frac{1}{\rho} \nabla q^{n+1} \right) \right\} \Rightarrow -\frac{1}{\Delta t} \nabla \cdot (H \boldsymbol{U}^*) = -\nabla \cdot \left(\frac{H}{\rho} \nabla q^{n+1} \right). \tag{2.22}$$

Note that $\nabla H = \boldsymbol{n}\delta_\Gamma$, where $\boldsymbol{n}$ is the outward-positive normal vector with respect to the solid body, and δ_Γ is the Dirac delta function on the interface. Based on this relation, the above equation can enforce the Neumann boundary condition automatically. In two-spatial dimensions, this equation can be discretized with the central difference approximation as follows:

$$-\frac{\dfrac{H_{i+\frac{1}{2},j}\left(q_{i+1,j}^{n+1}-q_{i,j}^{n+1}\right)}{\Delta x}-\dfrac{H_{i-\frac{1}{2},j}\left(q_{i,j}^{n+1}-q_{i-1,j}^{n+1}\right)}{\Delta x}}{\rho\Delta x}-\frac{\dfrac{H_{i,j+\frac{1}{2}}\left(q_{i,j+1}^{n+1}-q_{i,j}^{n+1}\right)}{\Delta y}-\dfrac{H_{i,j-\frac{1}{2}}\left(q_{i,j}^{n+1}-q_{i,j-1}^{n+1}\right)}{\Delta y}}{\rho\Delta y}$$

$$=-\frac{\left(H_{i+\frac{1}{2},j}u^*_{i+\frac{1}{2},j}-H_{i-\frac{1}{2},j}u^*_{i-\frac{1}{2},j}\right)}{\Delta t\Delta x}-\frac{\left(H_{i,j+\frac{1}{2}}v^*_{i,j+\frac{1}{2}}-H_{i,j-\frac{1}{2}}v^*_{i,j-\frac{1}{2}}\right)}{\Delta t\Delta y}. \tag{2.23}$$

The above discretization produces the symmetric-positive-definite system, and iterative methods in conjunction with convergence acceleration methods are strongly encouraged. In the following chapter, the multigrid method, known as the fastest linear solver for Poisson-type equations, is described and extended to the irregular domain problem.

Based on Gibou and Min algorithm, the pressure Poisson equation for a forced moving body can be written as

$$\nabla\cdot\left\{H\left(\frac{\boldsymbol{U}^{n+1}-\boldsymbol{U}^*}{\Delta t}=-\frac{1}{\rho}\nabla q^{n+1}\right)\right\}\Rightarrow\frac{1}{\Delta t}\left(\nabla\cdot(H\boldsymbol{U}^*)-\boldsymbol{U}_b\nabla H\right)=\nabla\cdot\left(\frac{H}{\rho}\nabla q^{n+1}\right), \tag{2.24}$$

since

$$\nabla\cdot(H\boldsymbol{U}^{n+1})=H\nabla\cdot\boldsymbol{U}^{n+1}+\boldsymbol{U}^{n+1}\nabla H=\boldsymbol{U}_b\nabla H, \tag{2.25}$$

where q is the scalar quantity, known as the *pseudo pressure*, H is the Heaviside function of the next time step, and $\boldsymbol{U}_b$ is the prescribed body velocity of the next time step. This approach works because $\nabla H = \delta_\Gamma\boldsymbol{n}$, where δ_Γ is the Dirac delta function on the interface and $\boldsymbol{n}$ is the outward-positive normal vector with respect to the body, and because $\boldsymbol{U}^{n+1}$, which is multiplied to ∇H, can be substituted by $\boldsymbol{U}_b$. The pressure Poisson equation on a moving irregular domain can be also solved by the multigrid method described in following chapter, which can be directly applied to an irregular domain problem.

Chapter 3

Multigrid Method for an Irregular Domain

In this chapter, a geometric multigrid (MG) algorithm using the edge/face-wise Heaviside function restriction is presented. This geometric MG algorithm is simple to implement, readily parallelizable, and can be applied to any irregular domain problem. The presented method shares many features with Weber et al. [116]. However, we employ the Heaviside function [33], which is a well-designed edge/face-wise function that can effectively represent domain irregularities, and the MG algorithm is implemented not *iteratively* but *recursively*. The optimal performance is demonstrated by solving the model Poisson problem exhibiting an analytical solution.

3.1. Standard multigrid method

It is known that iterative linear solvers such as Jacobi and Gauss-Seidel method can produce the smooth residual. High-frequency components of residuals are rapidly reduced in a few iterations using these iterative solvers, whereas low-frequency residuals converge very slowly. Low-frequency parts act like high-frequency components in a coarse grid. MG is based on this property. In other words, the high-frequency residual is eliminated on a fine grid in a few iterations, and then the remaining low-frequency components are projected to a coarse grid as source terms.

Since the low-frequency residual can be considered as a high-frequency part on a coarse grid, it can be successfully eliminated by iterative solvers with a small number of iterations. This procedure is consecutively performed until the coarsest grid level is reached. This process is known as coarse grid correction or restriction. In contrast, solutions computed at the coarse grid are successively interpolated to the fine grid until they reach the finest (original) grid, and this process is called as prolongation. Therefore, the MG solves the linear system using a single restriction-prolongation loop, called as the V-cycle, as depicted in Fig. 3.1. The remarkable property of MG is that once the MG algorithm is optimally constructed, the total number of MG cycles required for the convergence is nearly constant regardless of the grid resolution.

A Gauss-Seidel iterative solver is employed as a smoother in this study. The smoother is applied to both the restriction and prolongation procedures. The smoothers applied at restriction and prolongation are called to pre- and post-smoothers, respectively. In this study, the number of pre-smoothing is 10, and the number of post-smoothing is 2. The bottom-most, coarsest grid for the restriction is set to 2×2 in a 2D domain and $2 \times 2 \times 2$ in a 3D domain regardless of the top-most, initial, finest mesh resolution.

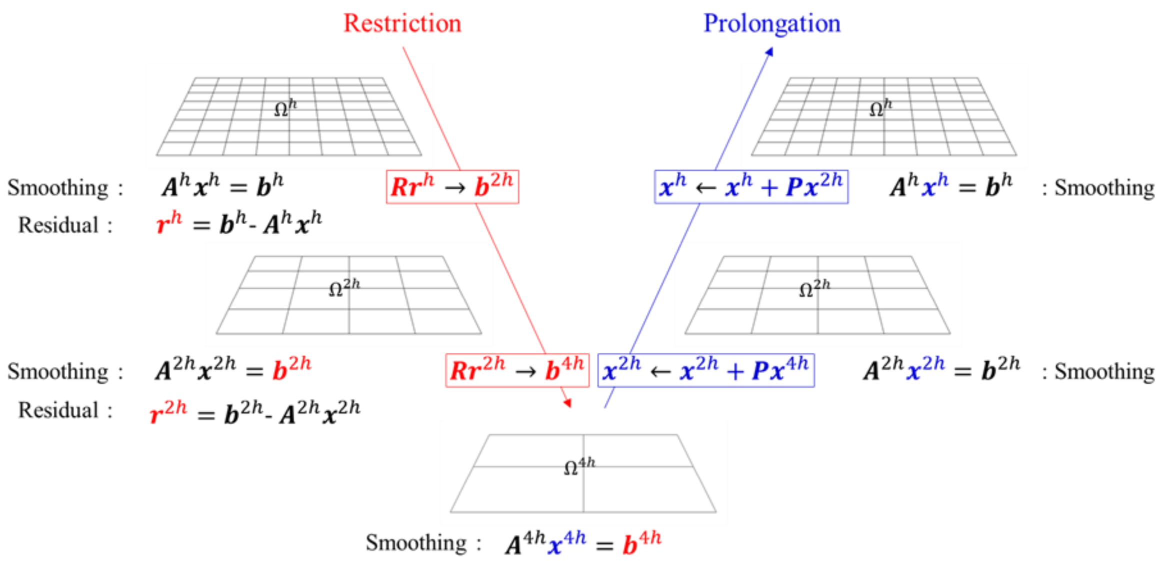

Restriction
Prolongation
Ω^h
Smoothing : $A^h x^h = b^h$
Residual : $r^h = b^h - A^h x^h$
$Rr^h \rightarrow b^{2h}$
$x^h \leftarrow x^h + Px^{2h}$
$A^h x^h = b^h$: Smoothing
Ω^{2h}
Smoothing : $A^{2h} x^{2h} = b^{2h}$
Residual : $r^{2h} = b^{2h} - A^{2h} x^{2h}$
$Rr^{2h} \rightarrow b^{4h}$
$x^{2h} \leftarrow x^{2h} + Px^{4h}$
$A^{2h} x^{2h} = b^{2h}$: Smoothing
Ω^{4h}
Smoothing : $A^{4h} x^{4h} = b^{4h}$

Fig. 3.1. Illustration of the V-cycle with three levels, where $\boldsymbol{R}$ and $\boldsymbol{P}$ refer to restriction and prolongation operators, respectively.

The cell-centered scheme is employed in the current pressure Poisson equation, where the unknowns are sampled at the center of each grid cell. The one-dimensional restriction and prolongation procedures for a cell-centered scheme are simply performed as shown in Fig. 3.2.

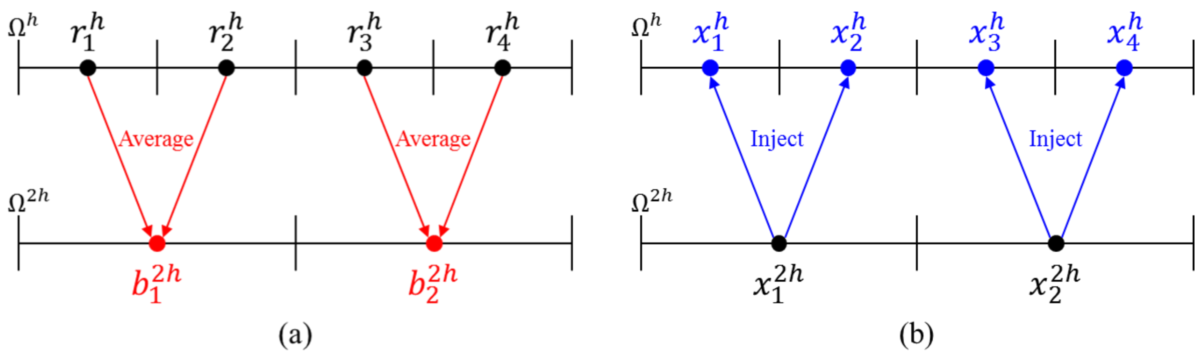


Fig. 3.2. One-dimensional (a) restriction and (b) prolongation procedure for the cell-centered scheme

In the restriction process, the source term in the coarse grid $\boldsymbol{b}^{2h}$ is computed by averaging the residuals of the fine grid $\boldsymbol{r}^h$ whose centers are involved in the corresponding coarse grid. In the case of prolongation, the coarse solution $\boldsymbol{x}^{2h}$ is equally injected to the fine solutions $\boldsymbol{x}^h$ which are contained in the corresponding coarse grid. Based on Fig. 3.2, the restriction and prolongation procedures can be written by

$$\underbrace{\begin{Bmatrix} b_1^{2h} \\ b_2^{2h} \end{Bmatrix}}_{\boldsymbol{b}^{2h}} = \underbrace{\frac{1}{2}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\boldsymbol{R}} \underbrace{\begin{Bmatrix} r_1^h \\ r_2^h \\ r_3^h \\ r_4^h \end{Bmatrix}}_{\boldsymbol{r}^h} \Rightarrow \boldsymbol{b}^{2h} = \boldsymbol{R}\boldsymbol{r}^h, \tag{3.1}$$

$$\underbrace{\begin{Bmatrix} x_1^h \\ x_2^h \\ x_3^h \\ x_4^h \end{Bmatrix}}_{\boldsymbol{x}^h} = \underbrace{\begin{Bmatrix} x_1^h \\ x_2^h \\ x_3^h \\ x_4^h \end{Bmatrix}}_{\boldsymbol{x}^h} + \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{P}} \underbrace{\begin{Bmatrix} x_1^{2h} \\ x_2^{2h} \end{Bmatrix}}_{\boldsymbol{x}^{2h}} \Rightarrow \boldsymbol{x}^h = \boldsymbol{x}^h + \boldsymbol{P}\boldsymbol{x}^{2h}, \tag{3.2}$$

where $\boldsymbol{R}$ and $\boldsymbol{P}$ refer to the restriction and prolongation operators, respectively. Note that these two operators satisfy the variational property [13], i.e. $P = cR^T$ where c is a constant real value. It means that the prolongation operator and the restriction operator are transposes of each other up to a constant. This restriction and prolongation procedures can be easily extended to the 2D case as shown in Fig. 3.3.

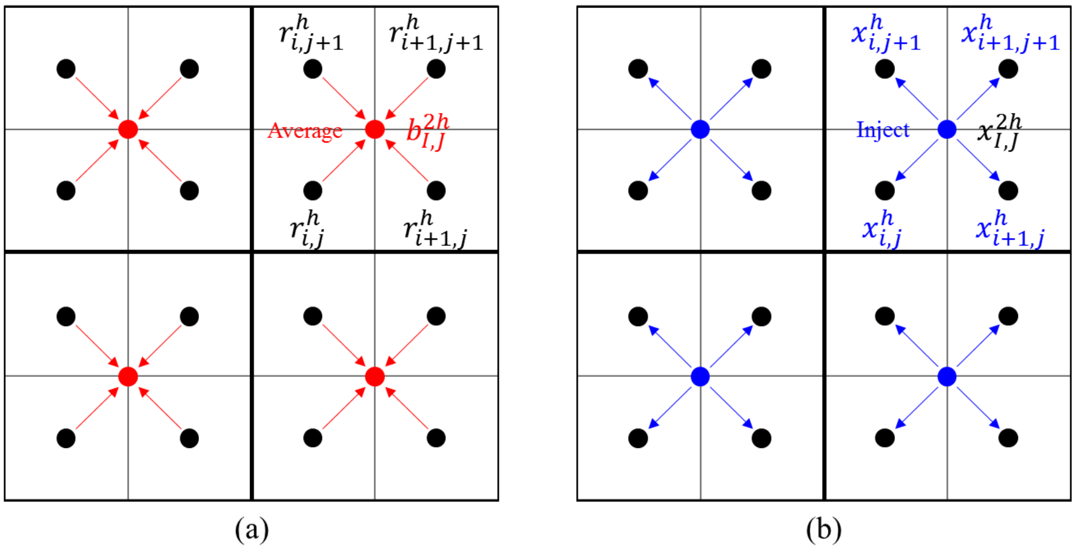


Fig. 3.3. Two-dimensional (a) restriction and (b) prolongation procedures for the cell-centered scheme, where $i = 2I - 1$ and $j = 2J - 1$.

The entire procedure of a single V-cycle for MG can be summarized as follows:

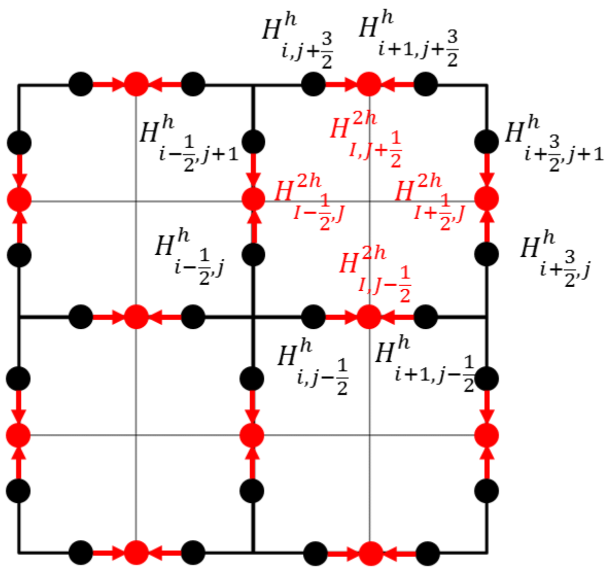
1. Relax the linear system on a fine grid as the pre-smoothing operation.
2. Compute the residual : $\boldsymbol{r}^h = \boldsymbol{b}^h - \boldsymbol{A}^h\boldsymbol{x}^h$.
3. Restrict the residual $\boldsymbol{r}^h$ to the source term $\boldsymbol{b}^{2h}$ on a coarse grid.
4. Repeat 1-3 until reaching the coarsest grid.
5. Relax the linear system on a coarse grid as a post-smoothing process.
6. Prolongate the coarse solution to fine solutions.
7. Repeat 5, 6 until reaching the finest (original) grid.

3.2. Multigrid method on an irregular domain

The current pressure Poisson equation for the irregular domain can be solved as in the case of the regular domain since the domain irregularity is automatically represented by the Heaviside function contained in the Poisson equation. Based on this fact, if the Heaviside function is consistently defined at a successively coarsened grid, the standard MG algorithm can be similarly applied to the current Poisson equation.

Considering the spatial discretization of the pressure Poisson equation as written in Eq. (2.23), the system matrix ($\boldsymbol{A}$) of the Poisson equation involves the Heaviside function. Therefore, in the restriction process, $\boldsymbol{A}$ should be re-constructed at a successively coarsened grid. Here, we employ the edge/face-wise restriction algorithm. Based on this algorithm, the Heaviside function on the edge/face of the coarse grid can be defined by averaging the Heaviside functions on its fine grid edges/faces. This procedure for two-spatial dimensions is depicted in Fig. 3.4.

The Heaviside function restriction algorithm can be extended to 3D cases in a similar manner. Instead of edge-wise averaging of the Heaviside function in 2D, face-wise averaging is applied to the Heaviside function restriction in 3D. More specifically, the Heaviside function on the cell face of the coarse grid can be defined simply by averaging the Heaviside functions at the corresponding four faces of its fine grids.



$H^h_{i,j+\frac{3}{2}}$
$H^h_{i+1,j+\frac{3}{2}}$
$H^h_{i-\frac{1}{2},j+1}$
$H^{2h}_{I,J+\frac{1}{2}}$
$H^h_{i+\frac{3}{2},j+1}$
$H^{2h}_{I-\frac{1}{2},J}$
$H^{2h}_{I+\frac{1}{2},J}$
$H^h_{i-\frac{1}{2},j}$
$H^{2h}_{I,J-\frac{1}{2}}$
$H^h_{i+\frac{3}{2},j}$
$H^h_{i,j-\frac{1}{2}}$
$H^h_{i+1,j-\frac{1}{2}}$

Fig. 3.4. The edge/face-wise Heaviside function restriction procedure in 2D, where $i = 2I - 1$ and $j = 2J - 1$.

The current edge/face-wise restriction algorithm for the Heaviside function is strongly recommended because of the following reasons:

- By using the pre-computed Heaviside function, any redundant computation of the Heaviside function can be eliminated, thus saving computational time and memory.

- If the Heaviside function is re-computed at each coarse grid, the geometric information could be easily lost during the restriction procedure.

Using the algorithm of the Heaviside function restriction, the MG algorithm is implemented recursively as shown in Algorithm 1. In this algorithm, the recursive function “multigrid_v_cycle” is defined. The sub-functions stated as "pre_smooth" and "post_smooth” are implemented by the red-black Gauss-Seidel algorithm. The sub-function "system_matrix" follows the discretization of the pressure Poisson equation as written in Eq. (2.23). The rest sub-functions, i.e., "restrict_cell_center", "prolongate", and "restrict_cell_edgeface", correspond to Fig. 3.3-(a), Fig. 3.3-(b), and Fig. 3.4, respectively. The current MG algorithm using the Heaviside function restriction can be extended to 3D cases in a similar manner.

Algorithm 1. Recursive function of multigrid V-cycle using the Heaviside function restriction

```
function multigrid_v_cycle($\boldsymbol{x}, \boldsymbol{b}, \boldsymbol{H}, \boldsymbol{n}$)
  $\boldsymbol{A} := $ system_matrix($\boldsymbol{H}$)
  $\boldsymbol{x} := $ pre_smooth($\boldsymbol{A}, \boldsymbol{x}, \boldsymbol{b}$)
  if ( min($\boldsymbol{n}$) $>$ 2 )
    $\boldsymbol{r} := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
    $\boldsymbol{b}^{2h} := $ restrict_cell_center($\boldsymbol{r}$)
    $\boldsymbol{H}^{2h} := $ restrict_cell_edgeface($\boldsymbol{H}$)
    multigrid_v_cycle($\boldsymbol{x}^{2h}, \boldsymbol{b}^{2h}, \boldsymbol{H}^{2h}, \boldsymbol{n}/2$)
    $\boldsymbol{x} := $ prolongate$\left(\boldsymbol{x}, \boldsymbol{x}^{2h}\right)$
  end
  $\boldsymbol{x} := $ post_smooth($\boldsymbol{A}, \boldsymbol{x}, \boldsymbol{b}$)
end
```

3.3. Verification study

To demonstrate the validity and performance of the current MG algorithm using the Heaviside function restriction, an analytically defined projection problem on the irregular domain [72] is considered. In this problem, the intermediate vector field $\boldsymbol{\mathcal{U}}^* = (\mathcal{u}^*, \mathcal{v}^*)$ that does not satisfy incompressibility is given to solve the following Poisson equation with irregular boundary:

$$\nabla \cdot (H \nabla \mathcal{q}) = \nabla \cdot (H \boldsymbol{\mathcal{U}}^*),$$

where $\mathcal{q}$ is the unknown of the Poisson equation. Next, $\boldsymbol{\mathcal{U}}^*$ is projected to a divergence-free vector field $\boldsymbol{\mathcal{U}} = (\mathcal{u}, \mathcal{v})$ by

$$\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{U}}^* - \nabla \mathcal{q}.$$

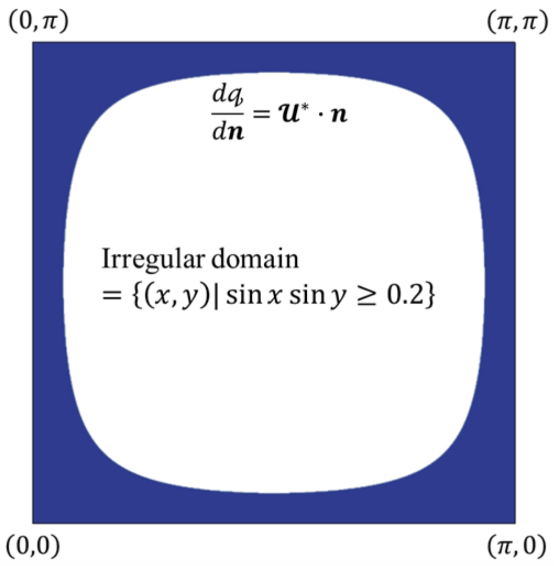

$(0,\pi)$
(π,π)
$\frac{dq}{d\boldsymbol{n}} = \boldsymbol{\mathcal{U}}^* \cdot \boldsymbol{n}$
Irregular domain
$= \{(x,y)|\sin x \sin y \geq 0.2\}$
$(0,0)$
$(\pi,0)$

Fig. 3.5. Configuration of the irregular domain problem with the Neumann boundary condition.

The irregular domain (see Fig. 3.5) is defined as $\{(x,y)|\sin x \sin y \geq 0.2 \text{ and } 0 \leq x,y \leq \pi\}$. Taking non-penetration condition $\boldsymbol{\mathcal{U}} \cdot \boldsymbol{n}$ on the irregular boundary, the Neumann boundary condition, i.e., $dq/d\boldsymbol{n} = \boldsymbol{\mathcal{U}}^* \cdot \boldsymbol{n}$, for the Poisson equation is imposed at the irregular boundary.

The intermediate vector field is given by

$$\boldsymbol{\mathcal{U}}^* = \left(\sin(x)\cos(y) + (x^2 - x)\left(\frac{y^3}{3} - \frac{y^2}{2}\right), -\cos(x)\sin(y) + (y^2 - y)\left(\frac{x^3}{3} - \frac{x^2}{2}\right)\right), \tag{3.3}$$

and it is projected to

$$\boldsymbol{\mathcal{U}} = (\sin(x)\cos(y), -\cos(x)\sin(y)). \tag{3.4}$$

Here, the analytical solution of the Poisson equation is given by

$$q = \left(\frac{x^3}{3} - \frac{x^2}{2}\right)\left(\frac{y^3}{3} - \frac{y^2}{2}\right). \tag{3.5}$$

The results of the current MG method using the Heaviside function restriction are presented together with those of the standard conjugate gradient (CG) method. First, the residual history is considered. Fig. 3.6 shows the L_2-norm of the residual versus the number of cycles (MG) or iterations (CG). In the current MG method, the solutions at each grid system are converged within the nearly fixed number of cycles. In contrast, the number of iterations of the CG method for the convergence increases as the problem size grows. This result implies that the current MG method optimally performs on the irregular domain, and this behavior is the distinctive feature of the MG method. Next, actual computation time is presented using linear and logarithmic scales as shown in Fig. 3.7. Here, the computation time at each grid system is normalized by dividing the computation time of the MG method at the coarsest system (64×64). In this figure, the superiority of the MG method is clearly confirmed, and the optimal performance, whose complexity is $O(N)$, is definitely demonstrated. Finally, Fig. 3.8 shows the convergence history of the error using the logarithmic scale. The numerical errors of the Poisson equation completely depends on the discretization scheme. Therefore, the results of both methods are identical and converge to the analytical solution with the desired second-order rate. Through the rigorous verification study, it is confirmed that the algorithm for the Heaviside function restriction allows the MG method to optimally and correctly perform for the Poisson equation on the irregular domain.

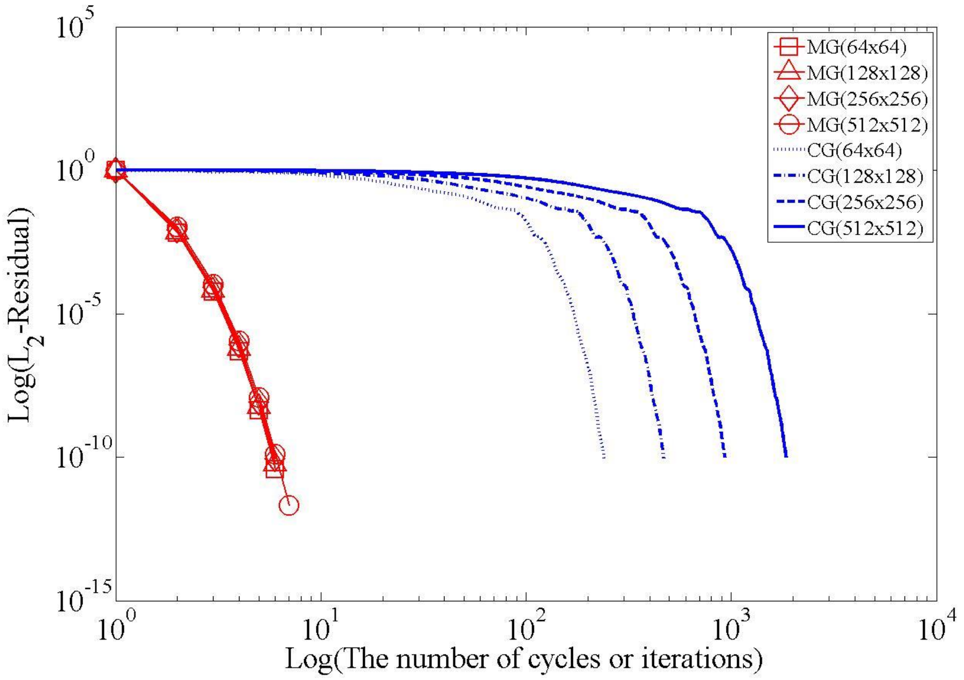


Fig. 3.6. The L_2-norm of residual versus the number of cycles (MG) or iterations (CG) using the logarithmic scale.

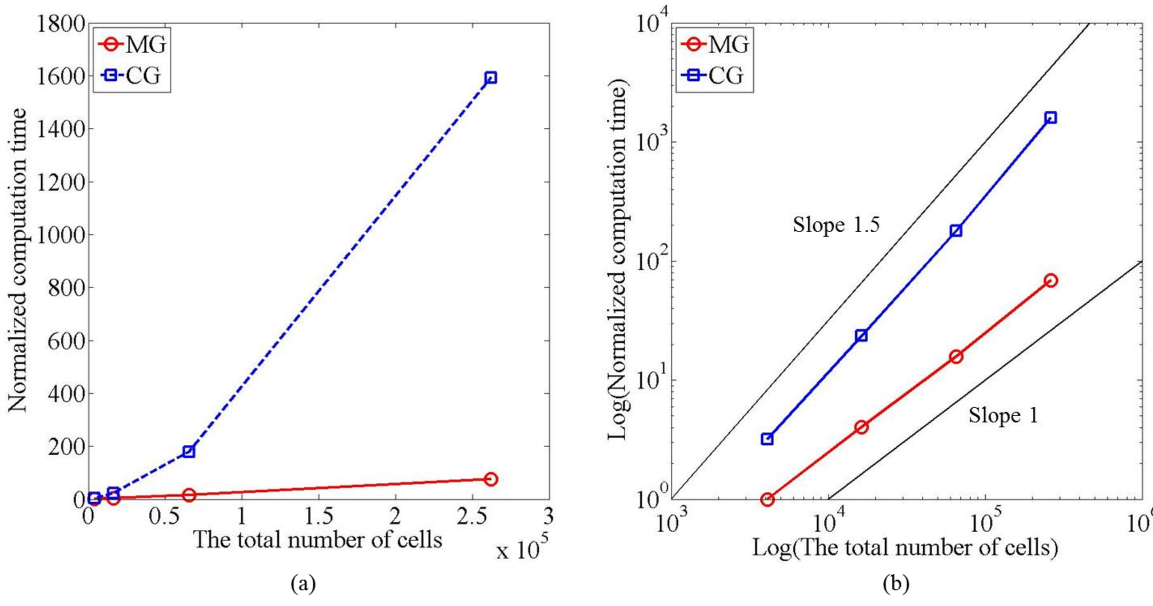


Fig. 3.7. Normalized computation time versus the total number of cells (N): (a) linear scale; (b) logarithmic scale. Computation time is normalized by dividing the computation time of the MG method at the 64^2 grid system. The complexity of the current MG method on the irregular domain is $\mathcal{O}(N)$, which is optimal.

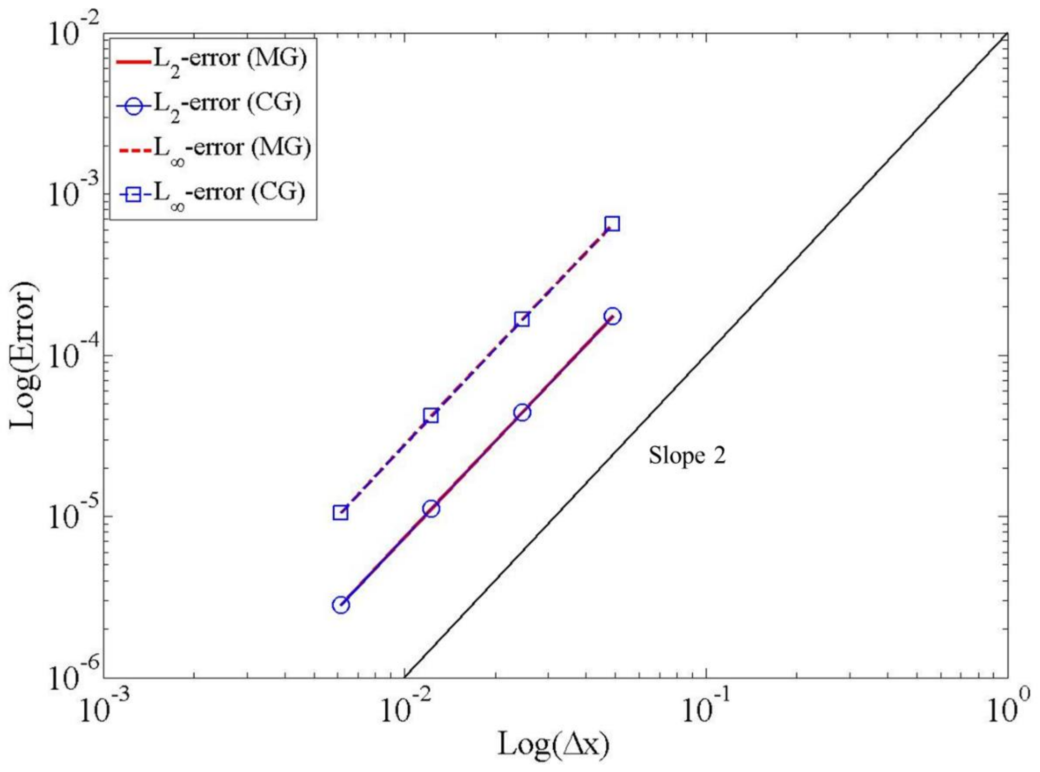


Fig. 3.8. Convergence history of the numerical error using the logarithmic scale. Numerical errors of the MG and CG methods are identical and converge to the analytical solution with the desired rate, which is second order.

Chapter 4

Adaptive Mesh Refinement based Signed Distance Function Computation

A special indicator function must be pre-computed prior to the actual flow simulation based on the immersed/embedded boundary methods [76, 67, 29, 68, 28, 97, 52, 98, 14, 17, 21, 33, 72, 36, 37, 22, 34, 35, 43, 58, 62, 63, 78, 106, 107] to represent a body with complex shape. The most frequently used indicator is the signed distance function (SDF) [75, 38, 21, 25, 84, 36]. As a kind of the level-set function, the SDF can be defined as

$$\phi(\boldsymbol{x},t) = \begin{cases} \phi(\boldsymbol{x},t) = 0 & \text{if } \boldsymbol{x} \text{ is on the interface,} \\ \phi(\boldsymbol{x},t) < 0 & \text{if } \boldsymbol{x} \text{ is inside the body,} \\ \phi(\boldsymbol{x},t) > 0 & \text{if } \boldsymbol{x} \text{ is outside the body, i.e. the fluid region,} \end{cases} \tag{4.1}$$

where the magnitude $|\phi|$ is the shortest distance to the interface. The SDF is defined by the product of the sign (+/-) and distance, thus we need to determine the sign and shortest distance to the interface for any mesh point. To determine the sign, the in/out test of a given point should be performed, and this is more critical than the distance computation.

4.1. Basic concept

In many engineering simulations, an adaptive mesh refinement (AMR) has been utilized to improve accuracy by refining cells in the region of interest [54, 56, 7]. In the current study, however, the AMR strategy is employed to compute a signed distance function (SDF) in the most efficient way.

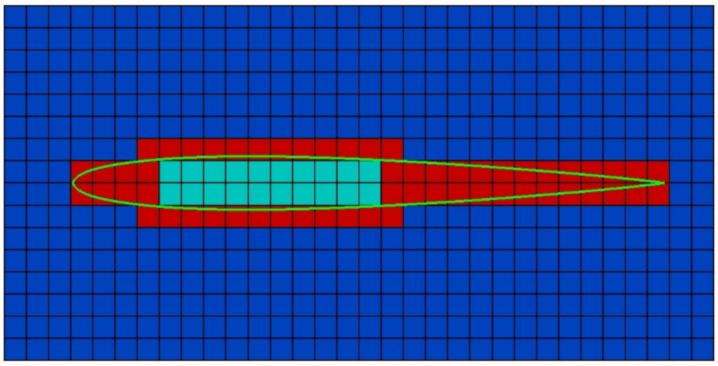


Fig. 4.1. Distinction of the computational grids for fluid simulation: blue and blue-green regions respectively indicate pure outside and inside cells, and red region indicates interface cells.

In the actual fluid simulation around the body on Cartesian meshes, the SDF for cells containing the body boundary is accurately required (red region of Fig. 4.1), whereas the SDF for the rest of the cells can be treated as arbitrarily large values if its sign is correct (blue and blue-green region of Fig. 4.1). The current SDF computation is inspired by this fact. The conceptual structure of the current algorithm is shown in Fig. 4.2.

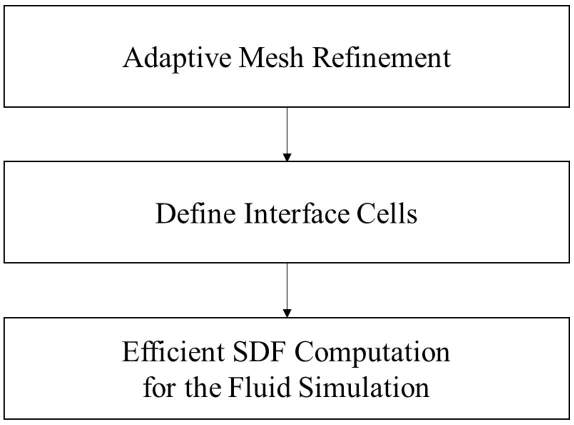


Fig. 4.2. Conceptual structure of the SDF computation using AMR. The interface cells can be effectively defined by the AMR procedure, and the accurate SDF computation is only applied to interface cells.

The AMR procedure is performed only for interface cells, which contain the body boundary, recursively using quadtree (2D) or octree (3D) grid structures. Through this refinement procedure, pure cells (without a body boundary) are not refined further, and only the interface cells are refined until they reach the finest grid level. Here, the finest grid level is pre-defined, and it is the one used for the actual fluid simulation. After completing the AMR procedure, all cells are hierarchically divided into pure cells and interface cells. In the case of interface cells, the sign and distance are accurately computed at every cell vertex. In contrast, for the pure cells, a one-step procedure for determining the sign is needed at a cell interior point (the cell center is considered here), and the distance is considered as an arbitrarily large value. In the current study, however, we interpolate the distance value of pure cell regions in order to obtain better graphical results.

In this way, we can compute the SDF, which is essential for the fluid simulation, in the most efficient way. In summary, the entire procedure of SDF computation driven by the AMR can be described as follows:

1. Interface cells at each mesh level are refined until the size of the cells is the same with the grid used in actual flow simulation.

2. After the AMR procedure, the leaf cells are classified into either interface cells or pure cells.

3. For the interface cells, both the sign and the distance are accurately computed at every vertex.

4. For the pure cells, the sign determination procedure is only performed to the cell center, and the distance is treated as an arbitrarily large value (Here, the distance is interpolated to obtain better graphical results).

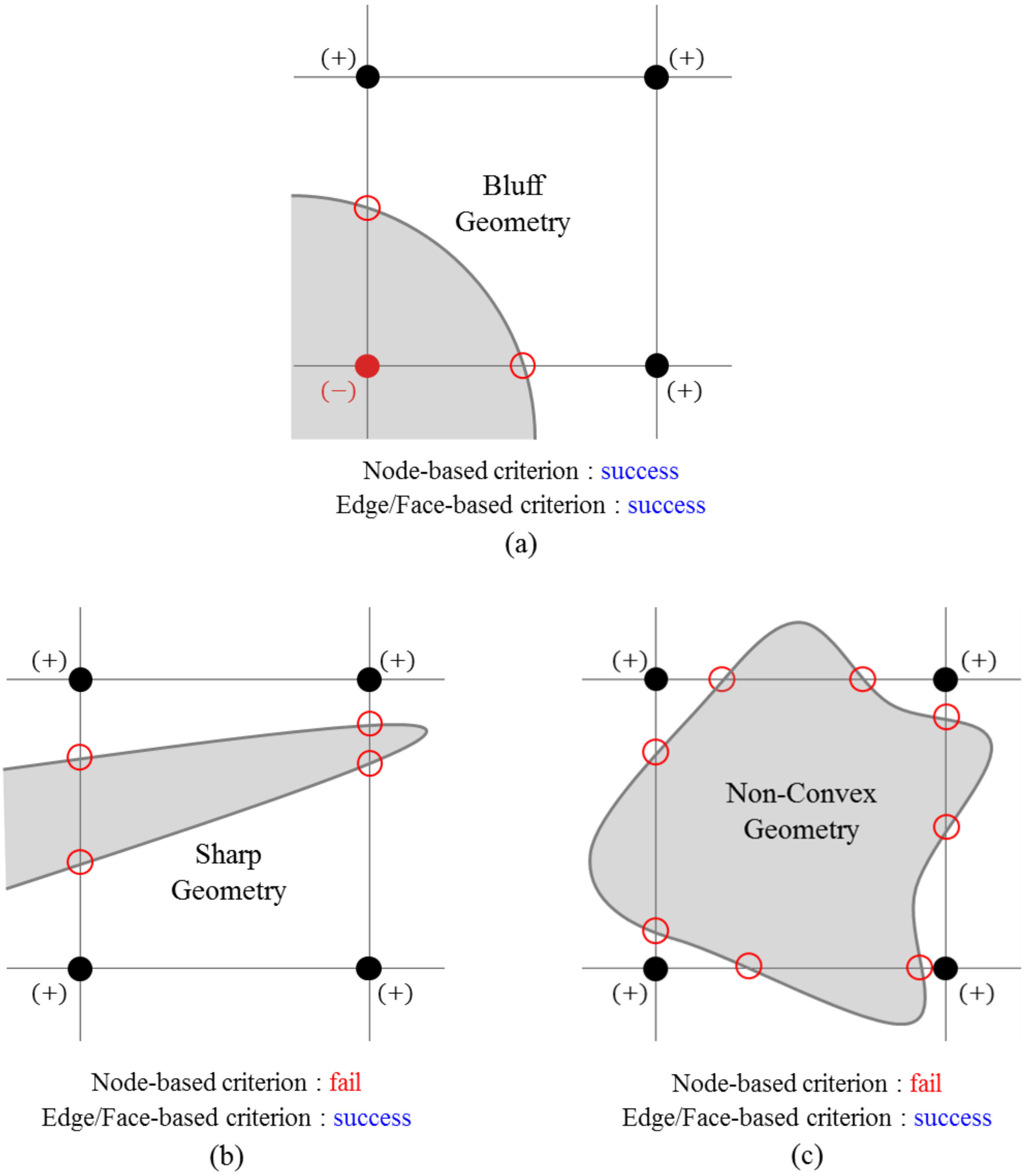

(+)
(+)
Bluff Geometry
(−)
(+)
Node-based criterion : success
Edge/Face-based criterion : success
(a)
(+)
(+)
Sharp Geometry
(+)
(+)
Node-based criterion : fail
Edge/Face-based criterion : success
(b)
(+)
(+)
Non-Convex Geometry
(+)
(+)
Node-based criterion : fail
Edge/Face-based criterion : success
(c)

Fig. 4.3 Two types of mesh refinement criteria can be considered, the first is a node-based criterion and the second is an edge/face-based criterion. (a) If the body shape is bluff and convex, both of refinement criteria can successfully detect the existence of the body. (b,c) if the body shape is sharp or non-convex, then only the edge/face-based criterion can detect the existence and the sub-cell features of the body. Hence, the edge/face-based criterion should be employed for refinement decision making.

4.2. Refinement criterion

An intelligent refinement criterion that can accurately detect the interface should be employed to execute the current AMR. There are two natural candidates that can be applied to the AMR criterion. One is a node-based criterion, and the other is an edge/face-based criterion.

The node-based method checks the sign (in/out) at each node. If negative (interior) and positive (exterior) signs coexist at the nodes of the given cell, then the cell is flagged as the interface cell. In contrast, in the edge/face-based method, the cell is flagged to be the interface cell if the intersection between the cell edge/face and body boundary is detected. Once a cell is flagged as the interface cell, it is then refined to quadtree or octree (3D) grids.

As depicted in Fig. 4.3, both refinement criteria can be successfully applied to a convex bluff geometry. However, the node-based criterion can easily fail for sharp or non-convex geometries. In contrast, the edge/face-based criterion can effectively detect sharp corners and non-convex geometries. For this reason, the edge/face-based method is utilized as the current AMR criterion. One can imagine that the edge/face-based criterion can still fail for the geometry, which is perfectly contained inside the cell. This problem can be simply resolved by setting the level-0 AMR mesh, which has at least one intersection point.

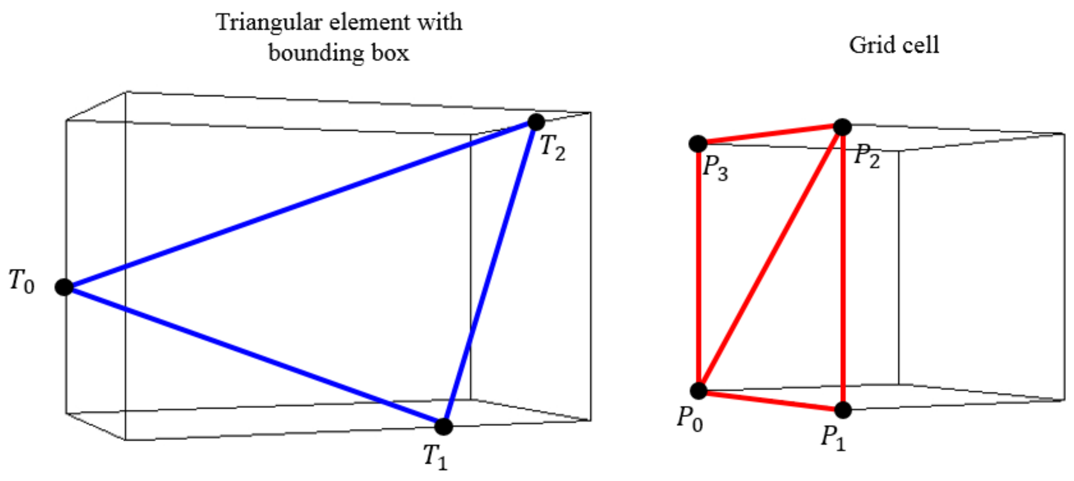


Fig. 4.4. Illustration of triangular element and grid cell face being tested for intersection. The blue triangle inscribed in the bounding box represents a part of tessellated immersed boundary, and the red rectangle indicates the grid cell face being tested for the intersection.

To implement the edge/face-based criterion, various geometric intersection algorithms, which are frequently employed to the interface reconstruction [3, 5], are required. In 2D, boundaries of geometries and grid cells are represented by line-segments. To check whether two line-segments intersect each other, the algorithm presented by [94] is employed. In 3D, boundaries of geometries and grid cells are represented by triangles and square planes, respectively. To check the triangle-plane intersection, the algorithm for the line-triangle intersection [69] is utilized iteratively. Here, the square plane is again divided into two sub-triangles, such as lower and upper triangles (see Fig. 4.4). The intersection between the triangle ($\Delta T_0 T_1 T_2$) and the square plane ($\square P_0 P_1 P_2 P_3$) can be checked as follows:

Step 1. Check whether $\Delta P_0 P_1 P_2$ is intersected by $\overline{T_0 T_1}$ or $\overline{T_1 T_2}$ or $\overline{T_2 T_0}$.

Step 2. Check whether $\Delta P_0 P_2 P_3$ is intersected by $\overline{T_0 T_1}$ or $\overline{T_1 T_2}$ or $\overline{T_2 T_0}$.

Step 3. Check whether $\Delta T_0 T_1 T_2$ is intersected by $\overline{P_0 P_1}$ or $\overline{P_1 P_2}$ or $\overline{P_2 P_3}$ or $\overline{P_3 P_0}$.

For each step, if an intersection is detected, the process is finished. If there is no intersection, another cell face of the given grid is selected. Then, the process is repeated. This process should be applied to all triangle elements. To minimize the operation, the bounding box approach is employed for shortlisting candidate triangles. Note that if there are intersections between the bounding box of the triangular element and the grid cell, this triangle is considered as the candidate.

4.3. Determination of the sign and distance

The best way to determine the sign is to count the number of intersections between an infinite ray starting from a given point and a target geometry. Here, if the ray intersects the body boundary an odd number of times, then the point is inside the body. Alternatively, if the ray intersects the body interface an even number of times (including zero times), then the point is outside the body. This approach is known as the ray-casting algorithm [85, 43, 36] (see Fig. 4.5).

To implement the ray-casting algorithm, geometric algorithms for the ray-line intersection and the ray-triangle intersection are needed in 2D and 3D, respectively. For the 2D case, the ray-line intersection algorithm is implemented by using [94]. A very fast ray-triangle intersection algorithm introduced by [69] is utilized for the 3D case. To determine the sign of the given point in 3D, all triangle elements need to be checked for the intersection. To accelerate this process, the intersection is checked only for the shortlisted candidate triangles by conducting the following process: if the ray is crossing the plane containing the triangle, this triangle is considered as the candidate.

In this study, the distance, i.e., the shortest distance between the geometry and the given point, is computed by using the following algorithms: the shortest distance between a point and a line-segment [92] in 2D and the shortest distance between a point and a triangle [27] in 3D.

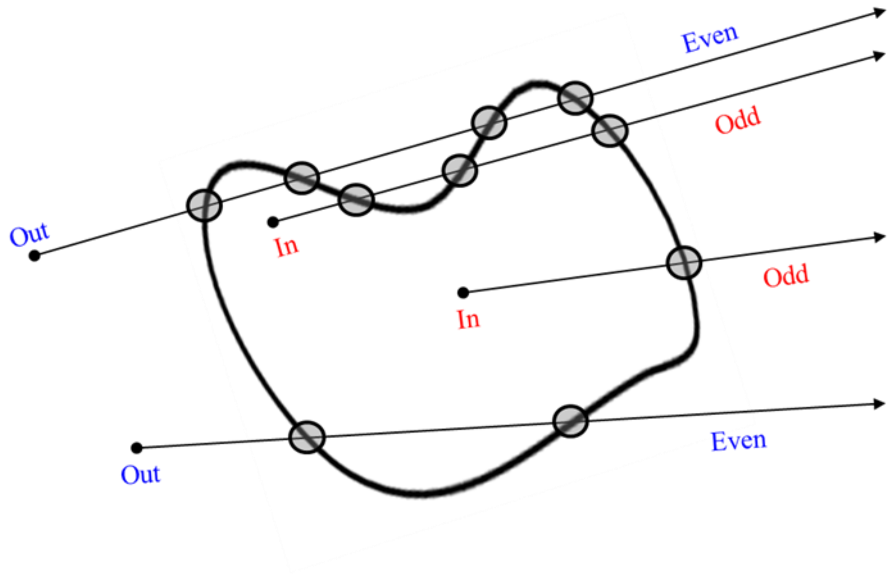


Fig. 4.5. Examples for the application of the ray-casting algorithm

4.4. Verification study

In this sub-chapter, efficiency and accuracy of the current AMR-based algorithm for computing the SDF are demonstrated. Simple smooth geometries (circle and sphere) are utilized as reference geometries for this verification study.

The radii of both the circle and sphere are set to 1. The computational domains of the circle and sphere are set to $[-2,2]^2$ and $[-2,2]^3$, respectively. The top and bottom row of Fig. 4.6 show the AMR procedures for the circle, composed of 128 line-segments, and the sphere, consisting of 1,280 triangles, respectively. As shown in this figure, only cells containing the boundary of the geometry are refined until the specified level. Note that the current AMR procedure is independent of the refinement level of neighbor cells, so the resulting grid is non-graded [7, 71].

It is known that the interface is the one-dimensional lower entity of the computational domain. Based on this property, the complexity of the current SDF computation focusing on interface cells can be expected as $O(n^{d-1})$, where n is the number of cells along the spatial direction and d is the number of spatial dimensions. This means that our AMR-based algorithm can accelerate the computation n times faster than the naive algorithm, whose complexity is $O(n^d)$, using the uniform mesh. This analysis is confirmed by the results of the actual numerical test presented in Fig. 4.7.

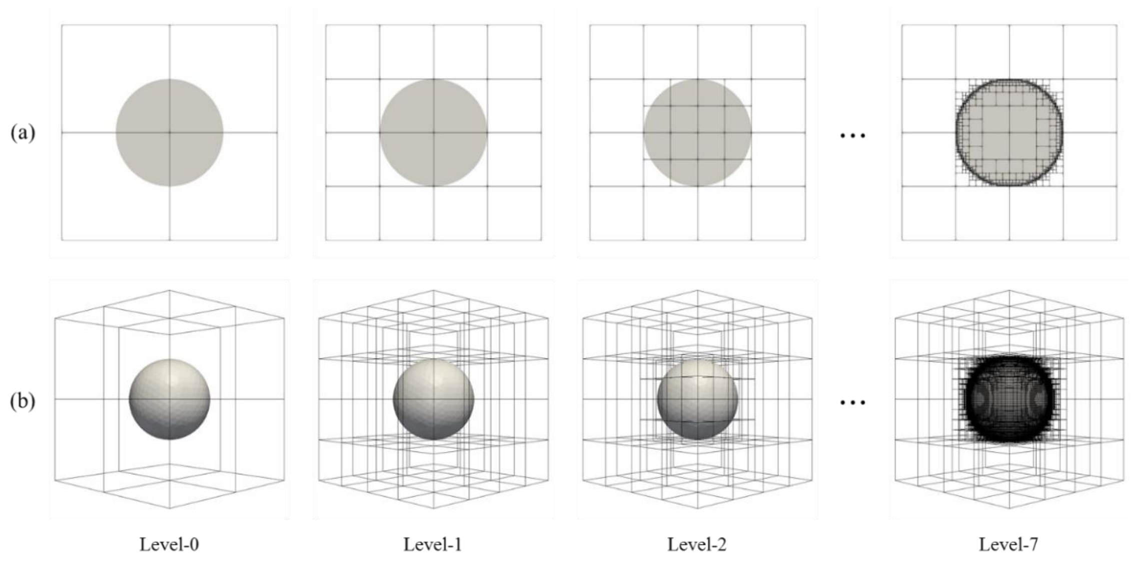
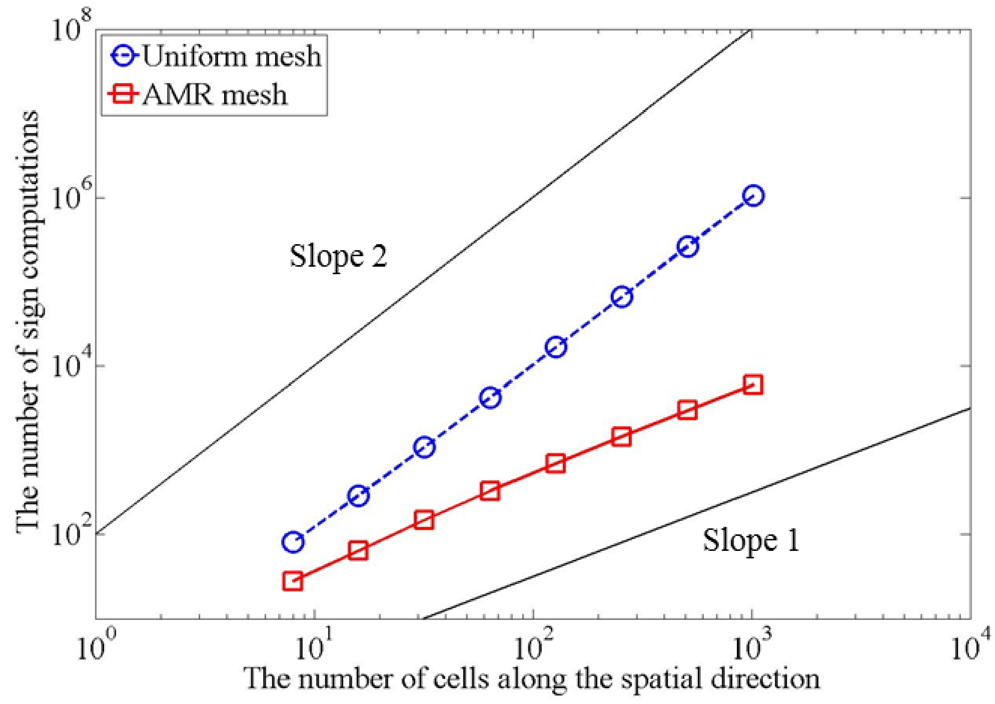
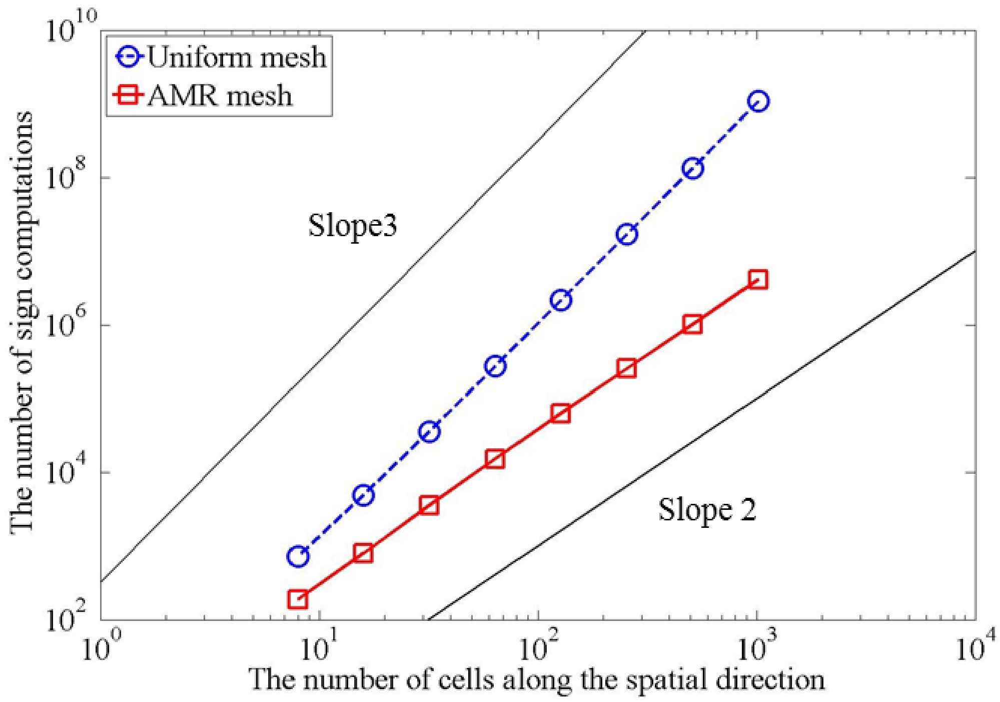


Fig. 4.6. The AMR procedure for the (a) circle with 128 line-segments and (b) sphere with 1,280 triangular surface meshes from the initial level-0 up to the finest level-7 grids. Note that the hierarchical structure of the resulting mesh reveals that the absolutely necessary cells involving the interface are selectively refined.



(a)



(b)

Fig. 4.7. Comparison of the total number of sign computation processes on a uniform mesh and AMR mesh: (a) 2D circle with 128 line-segments, (b) 3D sphere with 1,280 triangular surface meshes.

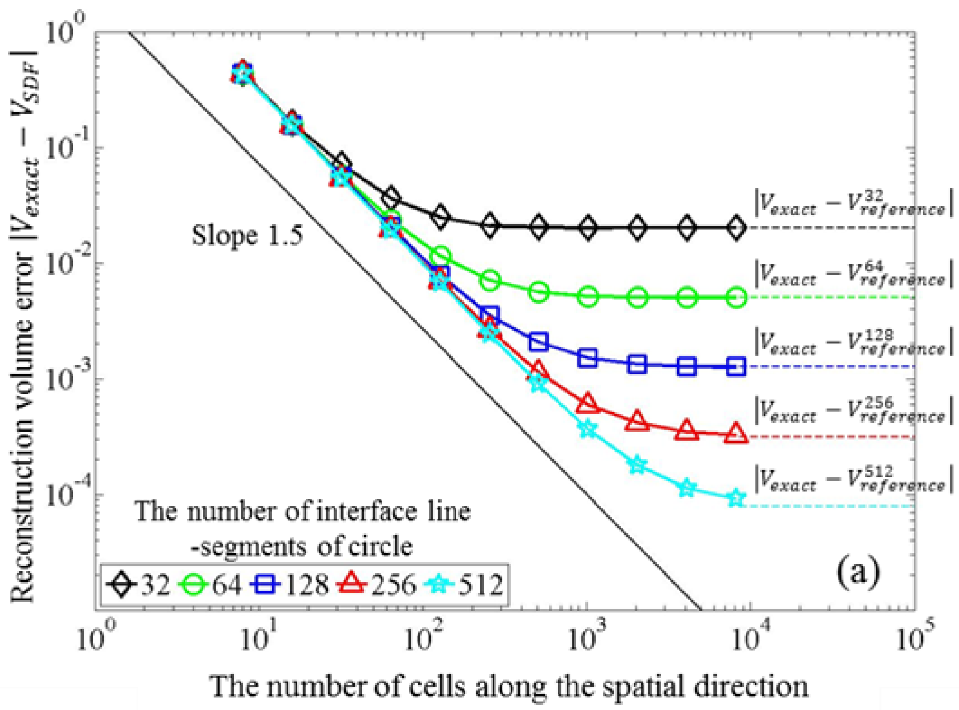
The volume of the reconstructed body (V_{SDF}) is computed to demonstrate the accuracy of the current approach. V_{SDF} is estimated by using the Heaviside function [33]. Since $\nabla H = \delta_{\Gamma}\boldsymbol{n}$, V_{SDF} can be simply computed by using the divergence theorem:

$$V_{SDF} = \int 1 \, d\Omega_b = \int \nabla \cdot \boldsymbol{s} \, d\Omega_b = \int \boldsymbol{s} \cdot \boldsymbol{n} \, d\Gamma_b = \int \boldsymbol{s} \cdot \nabla H \, d\Omega, \tag{4.2}$$

where $\boldsymbol{s}$ is the scaled position vector to be defined as $(x, y)/2$ in 2D and $(x, y, z)/3$ in 3D, and Ω_b and Γ_b refer to the region inside the body and the body interface, respectively.

Here, two kinds of errors are considered to analyze the volume convergence: reference volume error $|V_{exact} - V_{reference}|$ and reconstruction volume error $|V_{exact} - V_{SDF}|$. Fig. 4.8 shows the results of the volume convergence test of the circle and sphere. Based on the definition of reconstruction volume error, it converges to the corresponding reference volume error, but not further. At a low resolution, the reconstruction volume error is quickly bounded by the reference volume error since the circle and sphere can be considered as simple polygonal geometries. However, as the resolution of reference geometries increases, convergence occurs at a particular rate.

As shown in Fig. 4.8, the convergence rates are 1.5 in 2D and 2 in 3D, and these values are exactly consistent with the order-of-accuracy for the approximation of the Heaviside function utilized as a boundary integral [33]. This means that the computed SDF using the current method is valid.



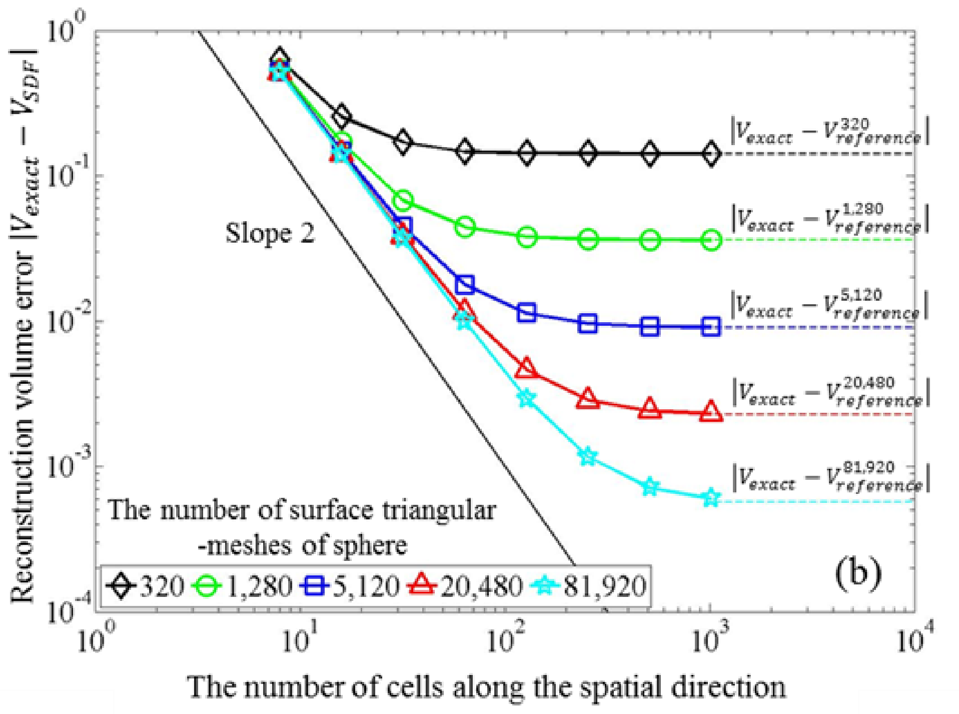


Fig. 4.8. Volume convergence test of (a) circle and (b) sphere by refining the computational grid. The resolution of the circle and sphere increases by increasing the number of interface elements. The reconstruction volume is computed by using the Heaviside function H defined by the computed SDF. Note that the current Heaviside function produces a 1.5 order-of-accuracy in 2D and a second order-of-accuracy in 3D. The reconstruction volume error converges to the reference volume error with the desired rates.

4.5. 2D examples

4.5.1. Multi-element airfoil

The multi-element airfoil (see Fig. 4.9) is considered as a first complex geometry. The multi-element airfoil is a high lift device and improves aerodynamic characteristics for several applications such as wings and turbines. This geometry consists of three separated elements, which have smooth surfaces as well as sharp corners. Since the typical features of engineering geometry, e.g. non-convexity, non-smoothness, corner singularity, multiple disjoint regions, and also smooth regions, are contained in this multi-element airfoil, it is a very appropriate example to demonstrate the robustness and effectiveness of the current approach. In this case, the boundary of the multi-element airfoil is composed of a set of 818 line-segments. The AMR procedure from level-0 to level-7 for the multi-element airfoil is depicted in Fig. 4.10. The level-0 is set to 4×1 uniform Cartesian meshes.

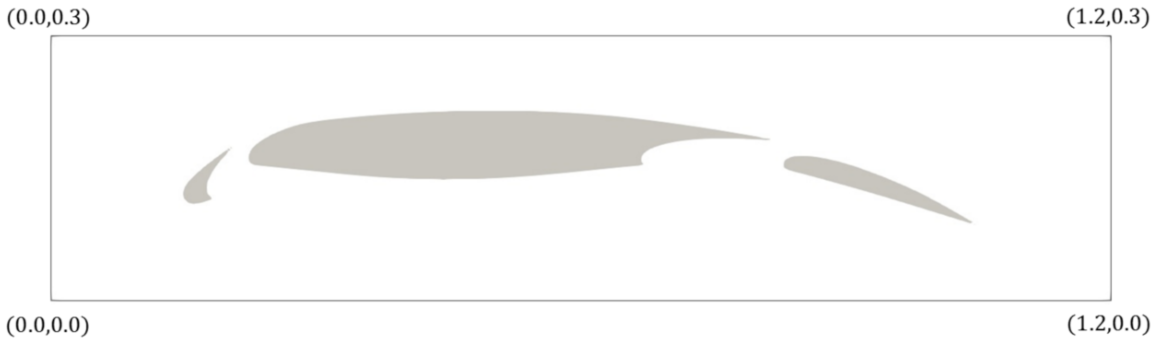


Fig. 4.9. Configuration of the multi-element airfoil [7]. This geometry includes various features of engineering design, for example non-convexity, a non-smooth sharp corner, a smooth bluff surface, and multiple combinations of these.

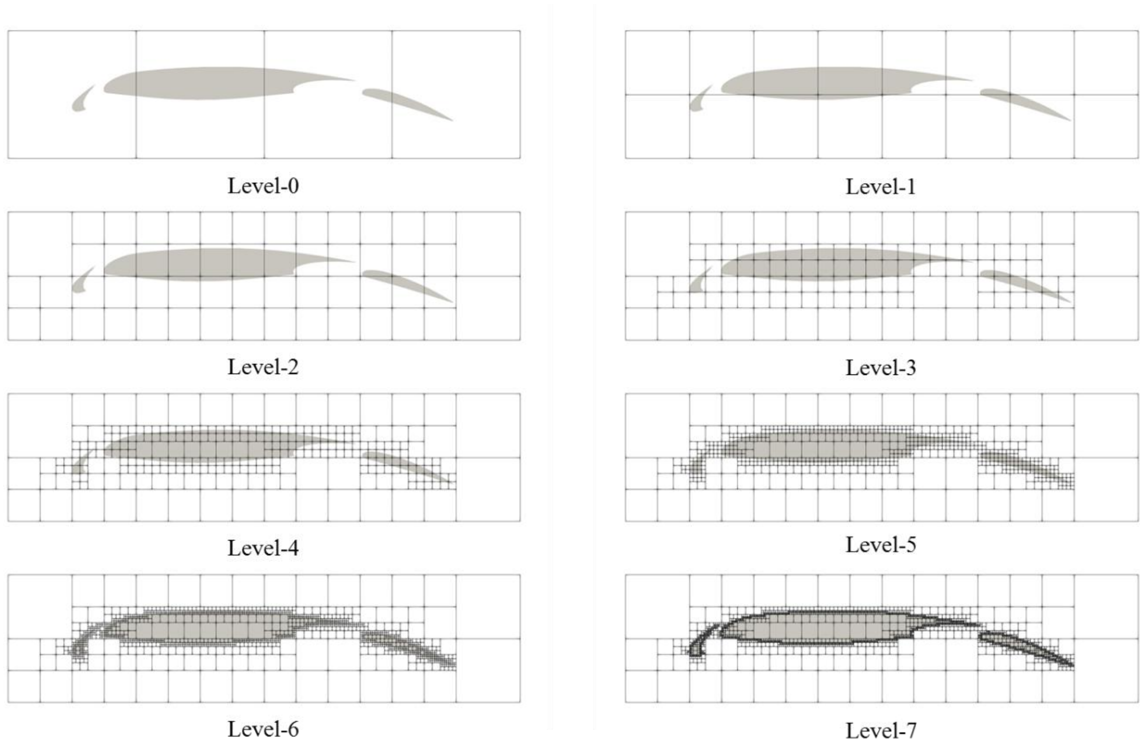


Fig. 4.10. The AMR procedure for the multi-element airfoil. Note that only the cells involving the interface of the geometry are refined regardless of the refinement level of its neighbors. Hence, the resulting AMR mesh is non-graded and optimal in the sense of requiring the least SDF computation.

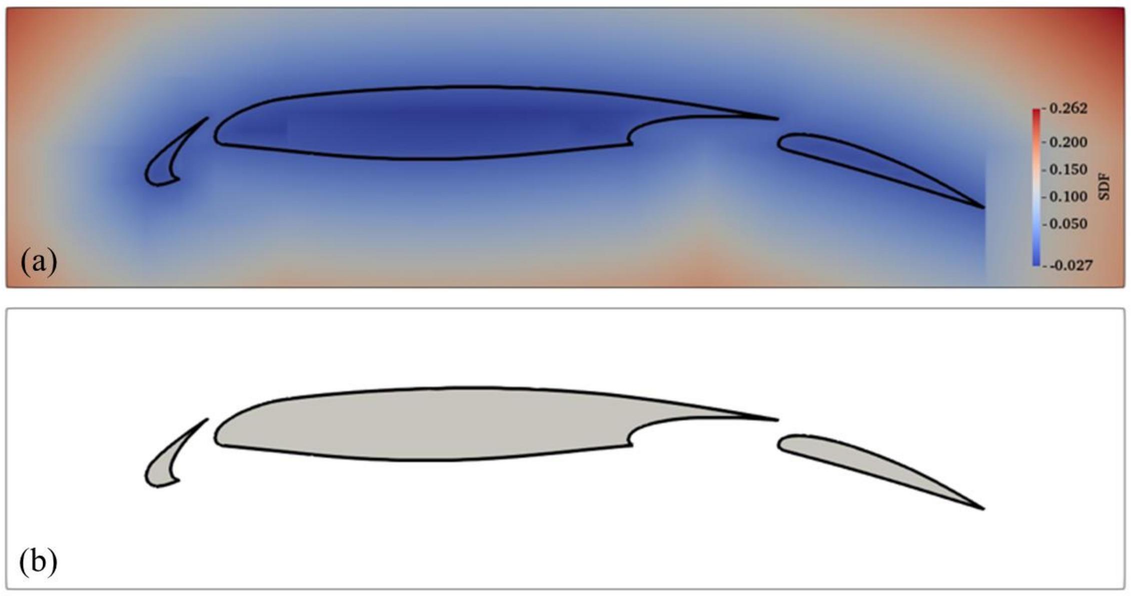


Fig. 4.11. The iso-line of zero SDF with (a) computed global SDF and (b) original geometry. The thick solid line, indicated by the zero SDF, defines the reconstructed geometry of the original multi-element airfoil. The cell-wise SDF distribution is determined by bilinear interpolation, whose accuracy is maximized on the finest level mesh containing the body interface. The shaded region of (b) represents the original geometry.

Fig. 4.10 shows that the current edge-based AMR method is well performed at sharp edges and non-convex interfaces that are included in each separated element. The main benefit of the AMR in this study is to detect the cells over the interface automatically, and the refinement is being carried out independently from the level of neighbor cells. Hence, the final AMR mesh is non-graded, and the number of interface cells is minimized, resulting in optimal performance in the SDF computation.

Fig. 4.11 illustrates the iso-line of zero SDF overlapped with the global SDF distribution and the original geometry of the multi-element airfoil. The cell interior distance value is approximated by bilinear interpolation, whose accuracy is maximum on the finest level mesh containing the interface cells. The SDF of the pure cells of the intermediate mesh level does not need to be accurate as long as its sign is correct. The boundary of the original geometry of the multi-element airfoil is exactly fitted by the iso-line of zero SDF.

4.5.2. S-shape

The S-shape (see Fig. 4.12) that is introduced in [7] is employed as a second 2D example. The S-shape can be produced by applying a divergence-free nonlinear velocity field to the initial circle region. The details are described in [7]. Since the S-shape is composed of a thin filament region, which has sub-cell thickness as well as sharp tips, it can be considered to be a suitable test case for demonstrating the sub-cell feature resolving capability of the current AMR procedure.

The computational domain is $[0,1] \times [0,1]$, and the boundary of the geometry consists of a set of 10^4 line-segments. The AMR procedure from level-0 to level-7 for the S-shape is presented in Fig. 4.13. As depicted in Fig. 4.13, the current AMR method is suitable for both sharp edges and thin filament structures with sub-cell scale features.

The global SDF distribution and original geometry overlapped with the iso-line of zero SDF, which represents the reconstructed body boundary of the original geometry, are shown in Fig. 4.14. Likewise, in the case of the multi-element airfoil, the iso-line of zero SDF clearly recovers the boundary of the original S-shape.

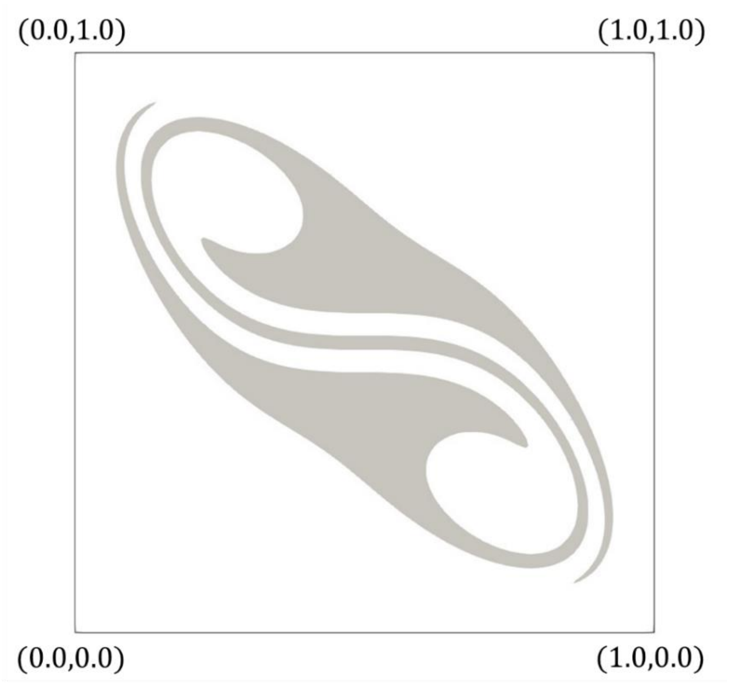


Fig. 4.12. Configuration of the S-shape [7]. This geometry is distinctive for its thin filament structure, which is initially sub-cell thickness that could be difficult to detect by the naive node-based refinement criterion.

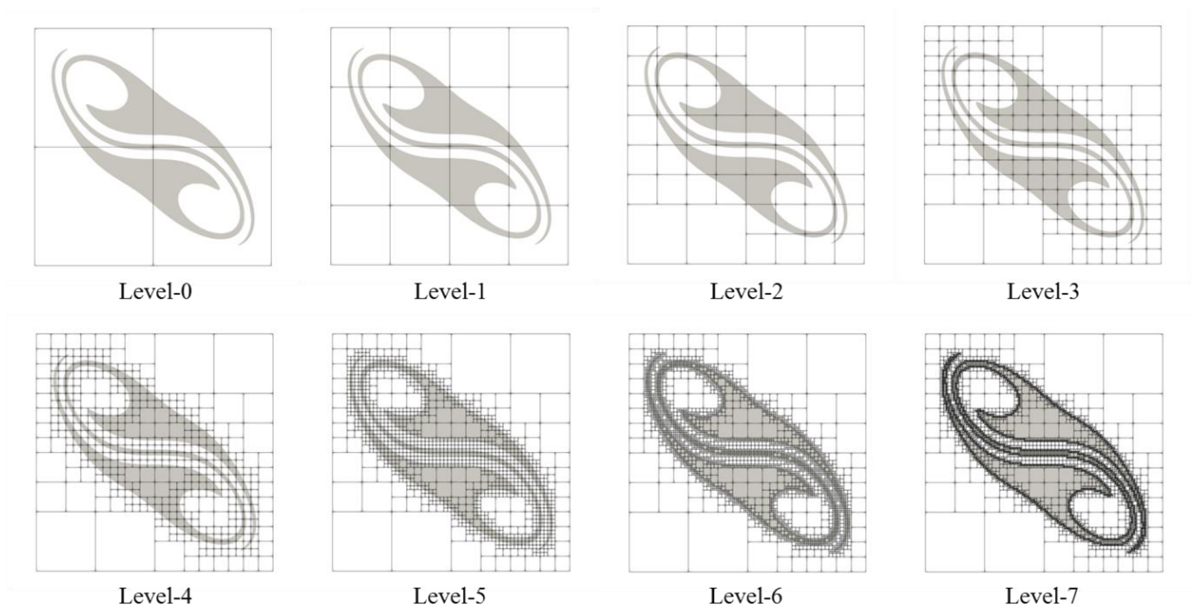


Fig. 4.13. The AMR procedure for the S-shape. The sub-cell thickness filament structure is well resolved by the current refinement criterion.

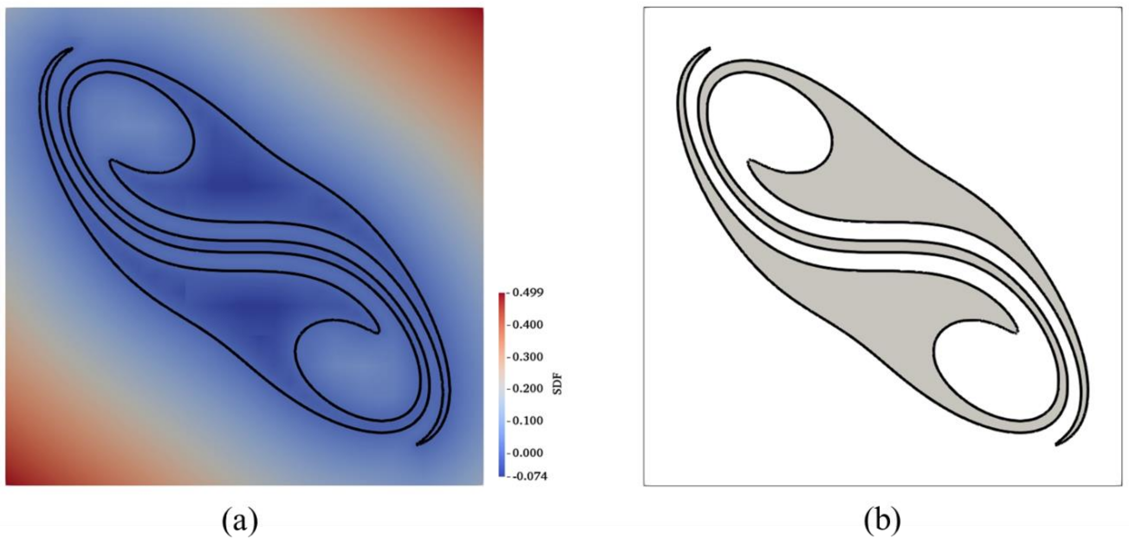


Fig. 4.14. The iso-line of zero SDF with (a) computed global SDF and (b) original S-shape. The thick solid line, marked by the zero SDF, indicates the geometry reconstructed from the original S-shape.

4.6. 3D examples

4.6.1. Propeller

A propeller is a type of fan that transmits power by converting rotational motion into thrust. Therefore, in order to generate the thrust, current propeller consists of 7-sharpe blade whose cross section is foil shape (see Fig. 4.15). Since this kind of propeller is regarded as one of the most complex shape in engineering field, the propeller is a very proper case in order to validate robustness of current methodology. The current propeller geometry is downloaded in online design community *GrabCAD (https://grabcad.com /library/propeller-airplane-1).*

The refinement procedure for the 3D propeller is presented in Fig. 4.16. The refinement is conducted from level-0 to 7. The current AMR algorithm based on detection of intersection between cell`s boundaries and geometry`s interfaces can be straightforwardly expanded to 3D case. As depicted in Fig. 4.16, the AMR is well performed to the 3D propeller.

The iso-surface of the zero SDF is shown in Fig. 4.17. The results are presented with four different background grid levels: grid level-0(64^3), 1(128^3), 2(256^3), 3(512^3). Furthermore, the iso-volume ($\phi \leq 0$) of grid level-3 is overlapped by exact surface meshes of propeller. Because blades of the propeller are too sharp, blades are not well represented at grid level-0. As grid level increases, the shape of propeller is well resolved. The overlapped image shows that the negative region of the SDF at grid level-3 accurately recovers the original geometry.

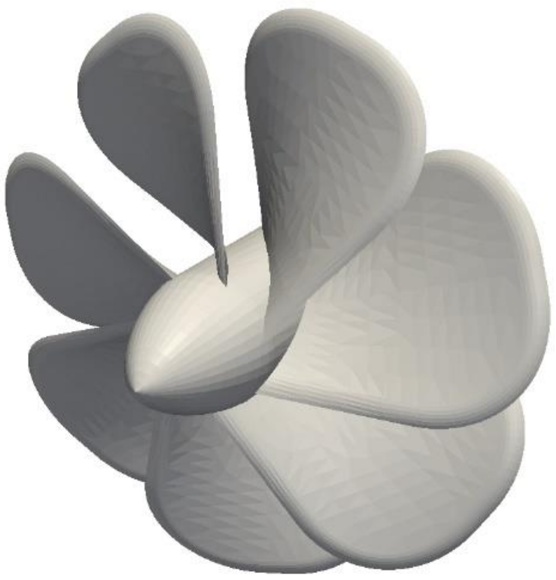


Fig. 4.15. The rendered surface of 3D propeller that consists of 7-blade and cone shaped hub. This 3D geometry is composed of 14,480 triangular suface meshes, and it is downloaded in online design community GrabCAD (https://grabcad.com/library/propeller-airplane-1).

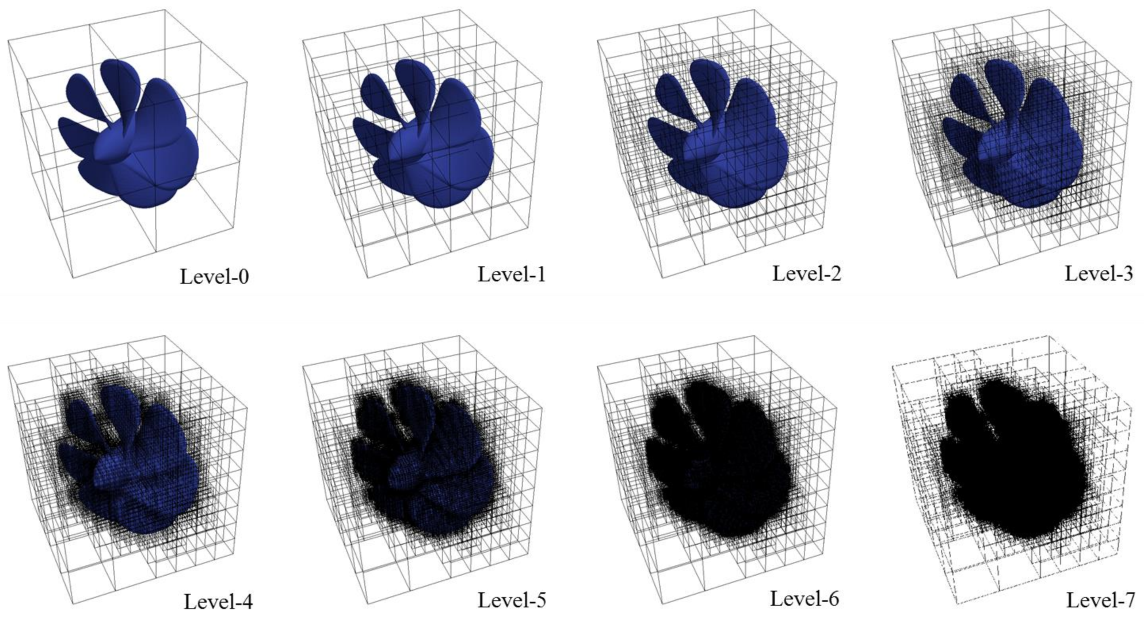


Fig. 4.16. The AMR procedure for the 3D propeller

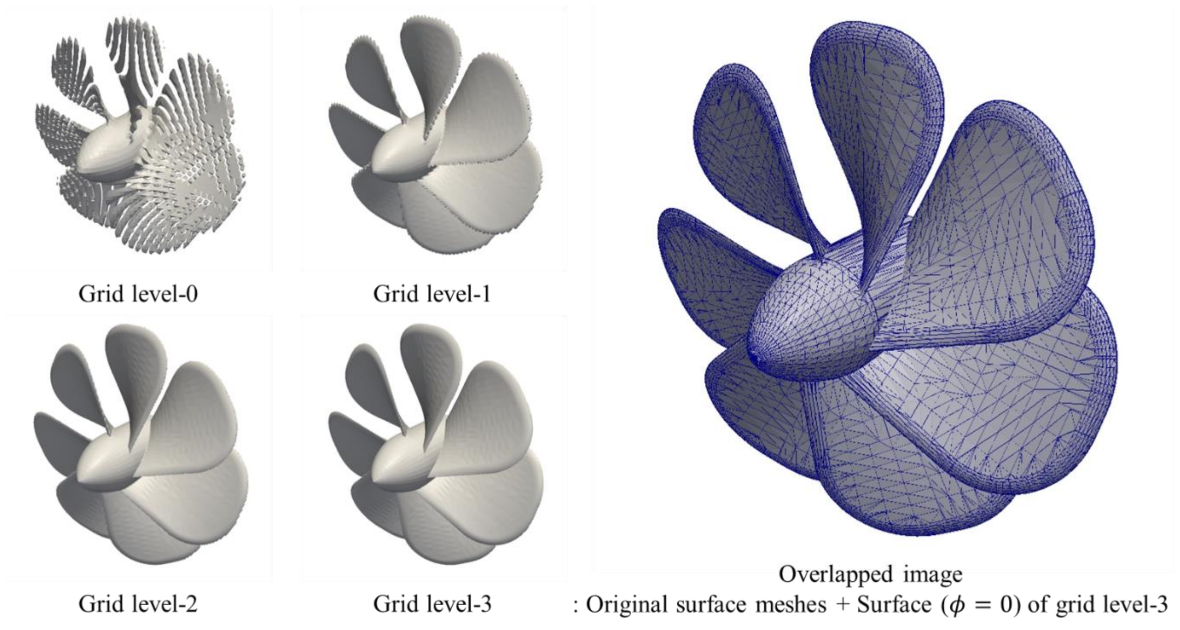

Grid level-0
Grid level-1
Grid level-2
Grid level-3
Overlapped image
: Original surface meshes + Surface ($\phi = 0$) of grid level-3

Fig. 4.17. The iso-surface of zero SDF for the propeller at different background grid levels: grid level-0, level-1, level-2, level-3. The iso-surface of zero SDF at grid level-3 is overlapped by original triangular surface meshes.

4.6.2. Underwater robot

Finally, the most challenging test case is presented in 3D. A Korean underwater walking robot called "Crabster" (see Fig. 4.18) [108] is employed as the 3D example for demonstrating the effectiveness of the current approach. The 3D stereolithography (STL) file that consists of triangular surface meshes of Crabster is produced from the computer-aided-design (CAD) file of Crabster provided by the marine robotics department of Korea Research Institute of Ships and Ocean Engineering (KRISO).

Crabster is composed of a main body and six legs, which are used for exploring the deep-sea environment by walking on the sea-bed. Due to the complexity of its shape, it is very hard to simulate flow around Crabster without a robust and efficient method of computing the SDF on the Cartesian grid system.

The entire AMR procedure for the SDF computation of Crabster is depicted in Fig. 4.19. As shown in this figure, the AMR is properly applied around the boundary of Crabster. The iso-surface of the zero SDF is shown in Fig. 4.20. The results are presented with four different computational grids: grid level-0 ($64^2 \times 32$), 1 ($128^2 \times 64$), 2 ($256^2 \times 128$), and 3 ($512^2 \times 256$). The original shape of Crabster is recovered as the grid level increases. In addition, the iso-surface ($\phi = 0$) of grid level-3 exactly fits the original surface triangular meshes and is visually indistinguishable. The SDF of the mid-cross section of Crabster at grid level-3 is presented in Fig. 4.21, where the crab-like shape is easily observable. Based on these results, we confirm that once the 3D geometry (surface meshes) of any complex shapes is provided, the SDF can be instantly and accurately computed through the current adaptive SDF computation algorithm.

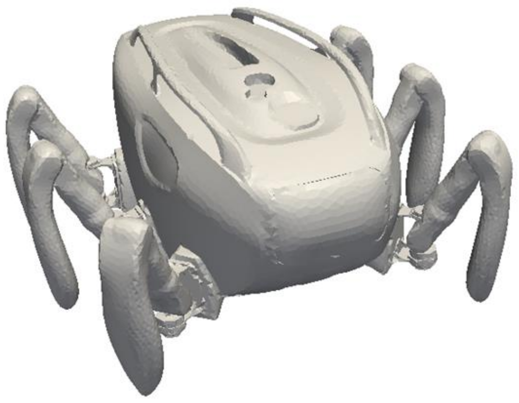


Fig. 4.18. The rendered surface of the underwater walking robot "Crabster". Crabster is a bio-inspired underwater robot which consists of a main body and six legs. This 3D geometry is composed of 49,400 triangular surface meshes, and its original geometric information is provided by the Korea Research Institute of Ships and Ocean Engineering (KRISO) [108].

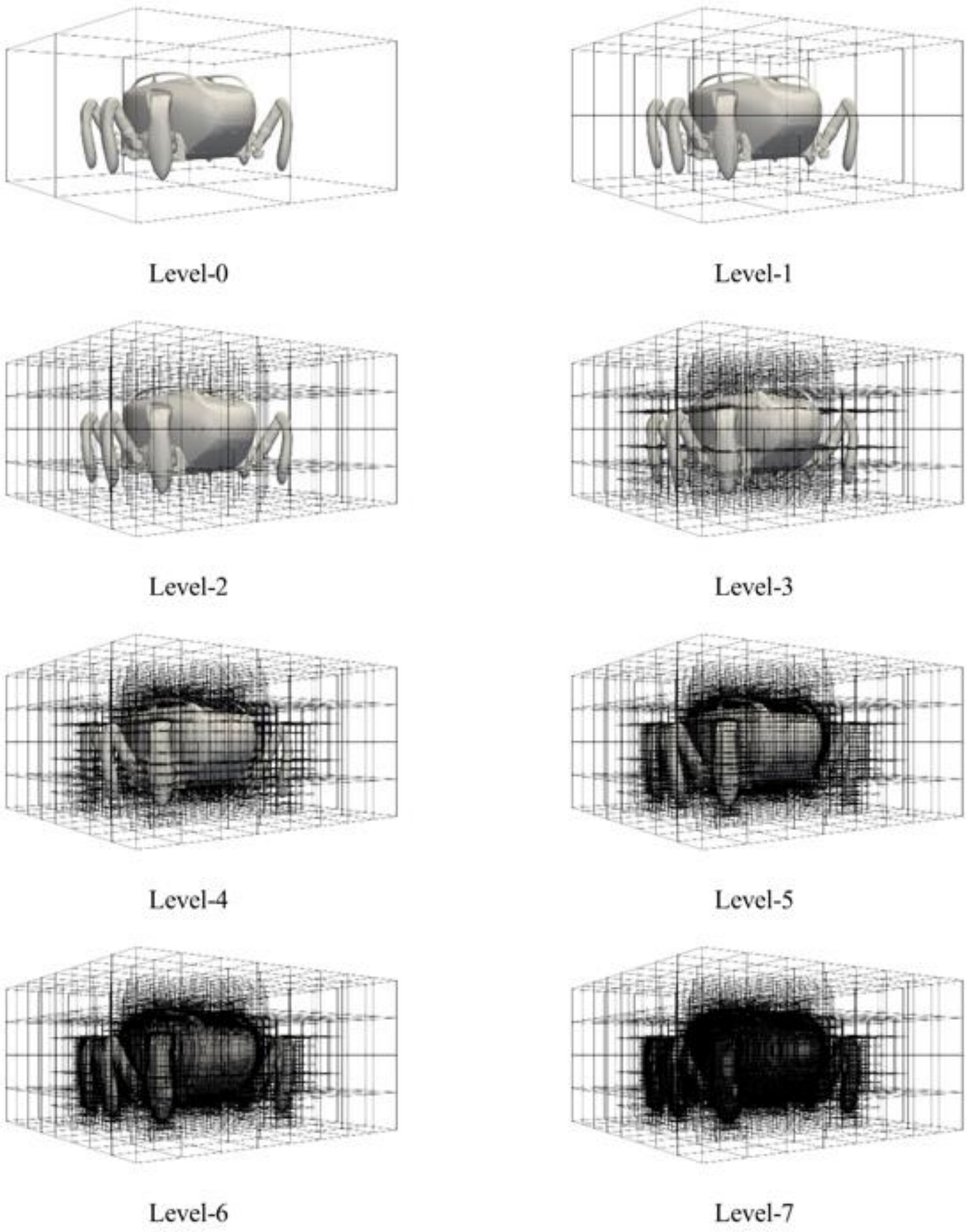


Fig. 4.19. The AMR procedure for Crabster from level-0 to 7. This result shows that the robustness and effectiveness of current AMR algorithm towards a very complex shape geometry.

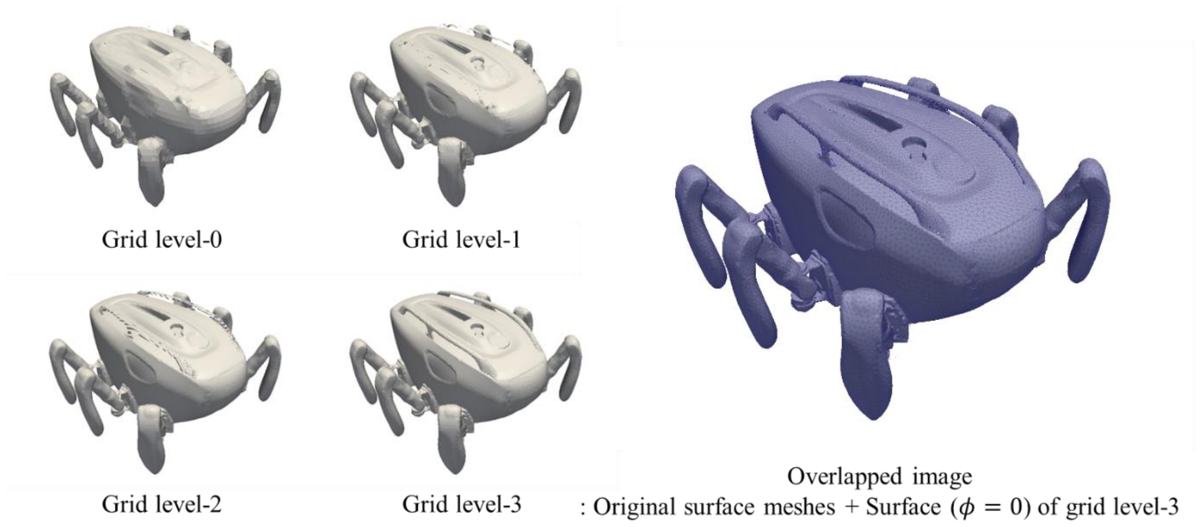

Grid level-0
Grid level-1
Grid level-2
Grid level-3
Overlapped image
: Original surface meshes + Surface ($\phi = 0$) of grid level-3

Fig. 4.20. The iso-surface of zero SDF for Crabster at different computational grid levels: grid level-0, level-1, level-2, level-3. The iso-surface of zero SDF at grid level-3 is overlapped by the original triangular surface meshes, and the two surfaces are visually indistinguishable.

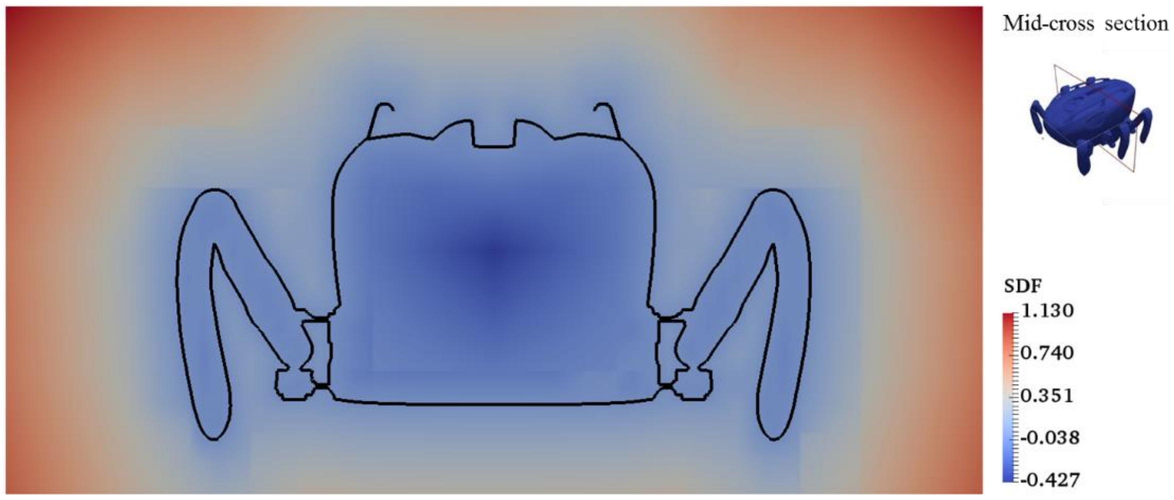


Fig. 4.21. The SDF distribution of the mid-cross section of Crabster, where the crab-like shape is observed. The thick solid line, marked by the zero SDF, indicates the geometry reconstructed from the original geometry in the mid-cross section.

4.7. SDF transformation for a moving body representation

A procedure for an accurate SDF transformation is described that avoids re-computation of the SDF. The key idea is that the new SDF of the moving body can be accurately and efficiently updated by transforming it from the initial SDF using a high-order interpolation. An example of the current approach for a moving NACA0012 airfoil is presented in Fig. 4.22. The procedure of the current SDF transformation is divided into two parts: one is the coordinate transformation, and the other is the interpolation. The details of each part are described as follows.

Two different coordinate systems, namely the global system and local (body-fixed) system, are utilized. The global coordinate system, indicated by (x, y) in Fig. 4.22, is for the fluid simulation, and the local system, represented by (x_b, y_b) in Fig. 4.22, is for the initial SDF computation. The center of the body-fixed coordinate system, indicated by (x_c, y_c) in Fig. 4.22, coincides with the point where the body motion is defined. The position defined in the global coordinate system can be transformed to that of the body-fixed coordinate system as follows:

$$\begin{Bmatrix} x_b \\ y_b \end{Bmatrix} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \begin{Bmatrix} x - x_c \\ y - y_c \end{Bmatrix}, \tag{4.3}$$

where θ is the rotation angle of the body, whose positive direction is counter-clockwise.

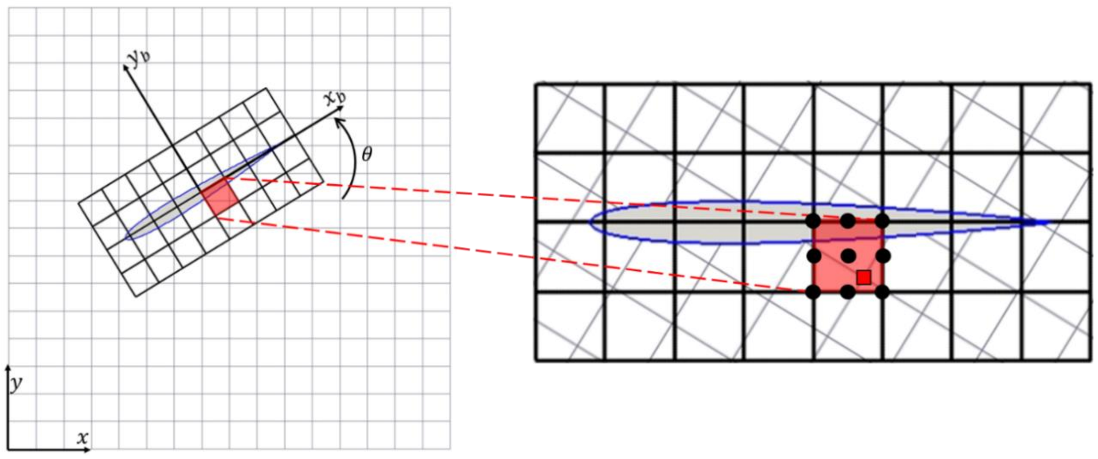


Fig. 4.22. An example for the SDF transformation of an NACA0012 airfoil. The left is an overlapped illustration of the global fluid domain and the body-fixed SDF domain, and the right is a zoomed-in view for the body-fixed domain. To apply biquadratic interpolation, the p-refined grid system (●) is employed to the body-fixed domain. The position of the grid point (■) for the fluid simulation, originally defined at the global coordinate, is transformed to the values for the local coordinate, and the corresponding SDF is determined by interpolating the SDFs defined at grid points (●).

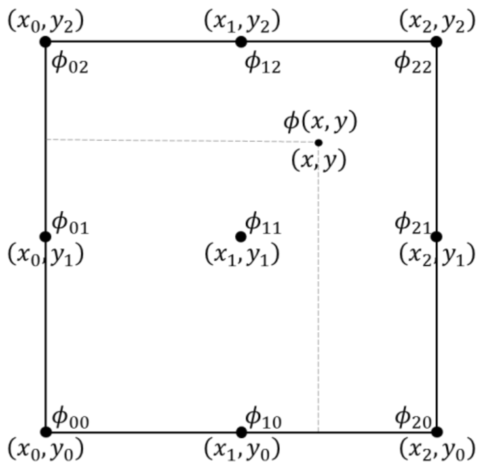


Fig. 4.23. The layout of a p-refined grid cell for biquadratic interpolation

Generally, the global grids do not coincide with the local girds (see Fig. 4.22). In this case, the SDF at the global grid point can be transformed from the SDF information of the local cell involving that grid point by using the interpolation technique. Here, biquadratic interpolation is used. The local cell involving the given global grid point can be easily traced by using the transformed position information, Eq. (4.3), of the given point from global to the local coordinate system. If the transformed position is not included in the body-fixed domain, the corresponding SDF can be considered as an arbitrary positive value. Through this process, the new SDF of the moving body in the fluid simulation can be efficiently updated by using the initial SDF.

By applying the p-refinement technique, the SDF can be accurately interpolated by an arbitrary high-order scheme. In this study, biquadratic interpolation is used, which has third-order accuracy. Because the fluid solver, which is described in Chapter 2, is based on the second-order accurate algorithm, an interpolation scheme that is one order higher can be considered sufficient for accurately representing the body information. The layout of a p-refined grid for the biquadratic interpolation is shown in Fig. 4.23, and this configuration corresponds to the square shaded region represented by the symbol (●) in Fig. 4.22. To apply the biquadratic interpolation, the additional operations for SDFs $(\phi_{10}, \phi_{01}, \phi_{11}, \phi_{21}, \phi_{12})$ are required when determining the initial SDF computation where the SDF computation is performed only to vertices. The biquadratic interpolation can be conducted at an arbitrary interior point of the cell as follows:

$$\phi(x, y) = \sum_{i=0}^{2} \sum_{j=0}^{2} \left\{ \left(\prod_{l=0, l \neq i}^{2} \frac{x - x_l}{x_i - x_l} \right) \left(\prod_{m=0, m \neq j}^{2} \frac{y - y_m}{y_j - y_m} \right) \phi_{ij} \right\}. \tag{4.4}$$

Chapter 5

Parallelization

5.1. Hybrid MPI/OpenMP parallelization

Parallelization is a type of computation in which many processors execute a job simultaneously, and it is desired for flow simulations due to following two reasons. First is to reduce the wall-clock computation time. Naturally, when flow simulations are carried out, multiprocessor-based computations are faster than single process-based operation. Second is to overcome memory limitations of machines. In order to solve a large-scale problem, such as turbulent flows with high Reynold number and flow past complex shaped bodies, much more memory is required excessing the memory capacity of single machine. By dividing the computational domain to each processor, the computation for a large-scale problem can be possible without any memory limitations.

In CPU-based computation, Message Passing Interface (MPI) and Open Multi-Processing (OpenMP) are standardized programming approaches for parallel computing. MPI is based on distributed memory system for each processor, and interactions between processors are conducted by communications. On the other hand, OpenMP utilizes shared memory systems, thus the additional overhead for parallelization such as communications can be avoided. Based on the author`s experience, the programming of MPI is more complicated in terms of writing a code due to communications, and OpenMP should be more carefully applied because of shared memory system which can result in false data sharing.

Each method, i.e. MPI and OpenMP, has advantages compared to the other, and vice versa. Therefore, in order to complement each method and maximize the efficiency, the hybrid MPI/OpenMP parallelization [82] is gaining a popularity for high performance computing. In this approach, the computational domain is divided into several MPI processors, and threads for OpenMP are allocated to each MPI processor. Fig. 5.1 shows the schematic figure of the hybrid MPI/Open-MP parallelization.

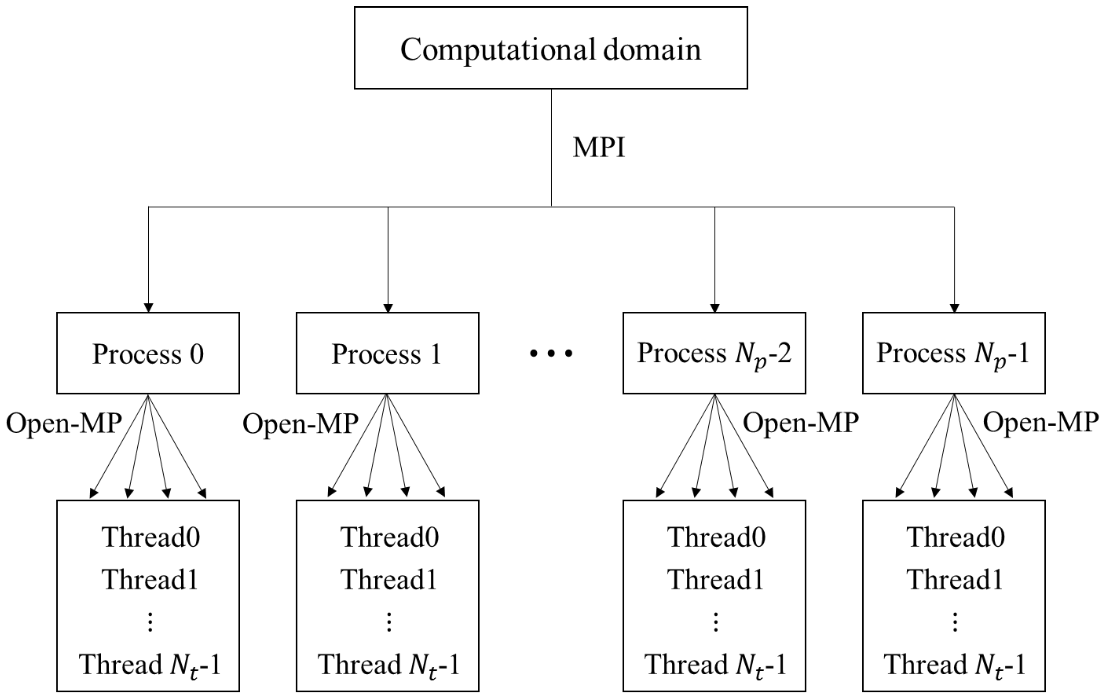


Fig. 5.1. The schematic figure of the hybrid MPI/OpenMP parallelization where N_p and N_t refer to the number of processors (MPI) and threads (OpenMP), respectively.

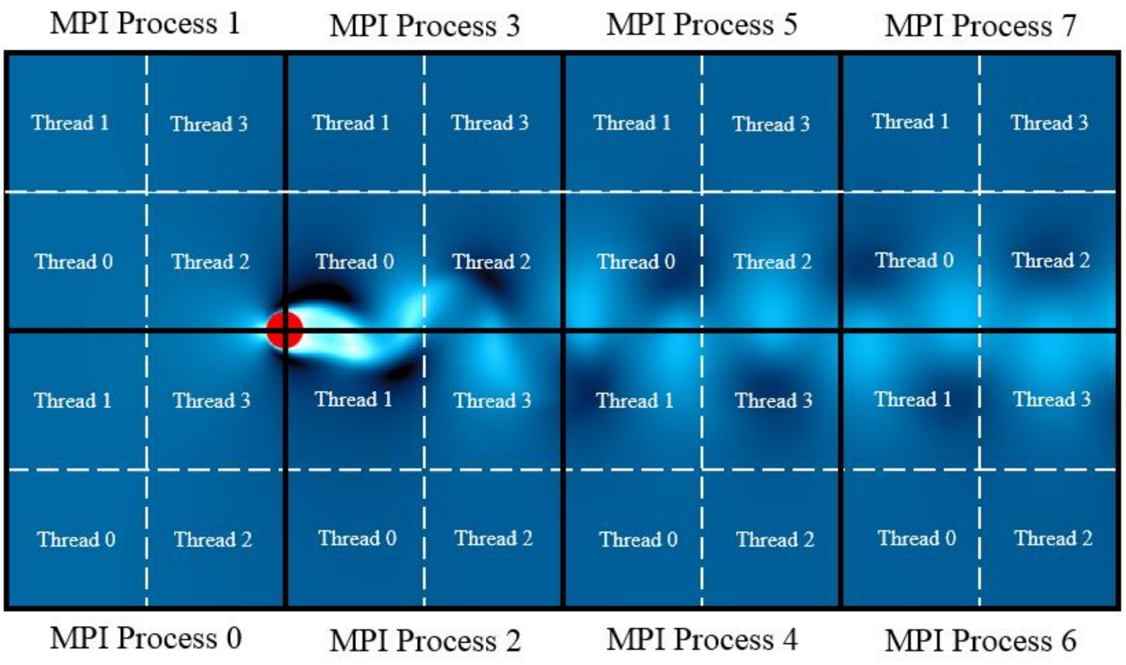


Fig. 5.2. The application of the hybrid MPI/OpenMP parallelization for flow past a circular cylinder problem at $Re = 100$. The computational domain is divided into 8 processors for MPI, and 4 threads for OpenMP are assigned to each processor.

In the current research, the computational domain is divided into uniform Cartesian blocks for MPI parallelization, i.e. squares in 2D and cubes in 3D. For example, Fig. 5.2 illustrates the application of the hybrid MPI/OpenMP parallelization for flow past a circular cylinder problem where 8 MPI processors are applied, and 4 threads for OpenMP are allocated to each processor.

As mentioned before, communications, which consist of send and receive process, between processors are required to MPI parallelization. In the current study, non-blocking communications based on *MPI_Isend* and *MPI_Irecv* functions are employed. To reduce the overhead resulted from communications, directionally sweeping communications are applied. For example, if communications are carried out from left to right, the data receiving from a left processor and the data sending to a right processor are needed to a given processor.

For the flow solver based on the fully Eulerian approach using Cartesian meshes, generally, only communications with face neighbor processors are needed because the spatial discretization at a given point is performed utilizing face neighbor cells. For the current solution algorithm based on the semi-Lagrangian method, however, communications with all neighbor processors wrapping a given processor are required due to backtracking procedure which is for finding departure points of previous time step. For this reason, communications with vertex neighbors are additionally required in 2D case (see Fig. 5.3), and communications with vertex and edge neighbors are needed for 3D case (see Fig. 5.4). As a result, total 8 and 26 communication routines are required to 2D and 3D domains, respectively. For the velocity Helmholtz equation and pressure Poisson equation, only communications with face neighbors are applied.

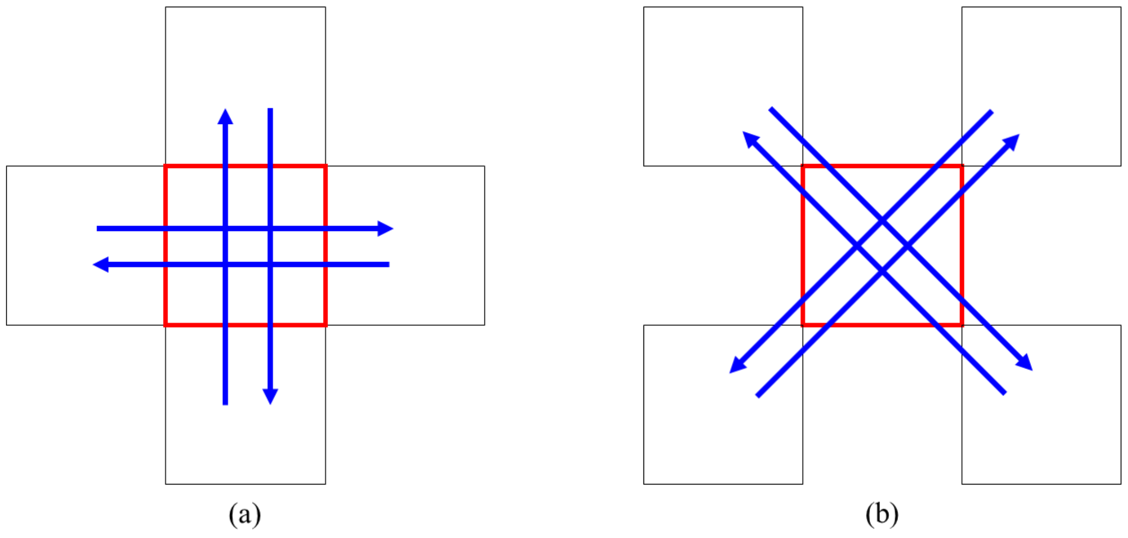


Fig. 5.3. Scenarios of MPI communications for the 2D domain. Total 8 sweeping communications are required: (a) 4 communications for face neighbors; (b) 4 communications for vertex neighbors.

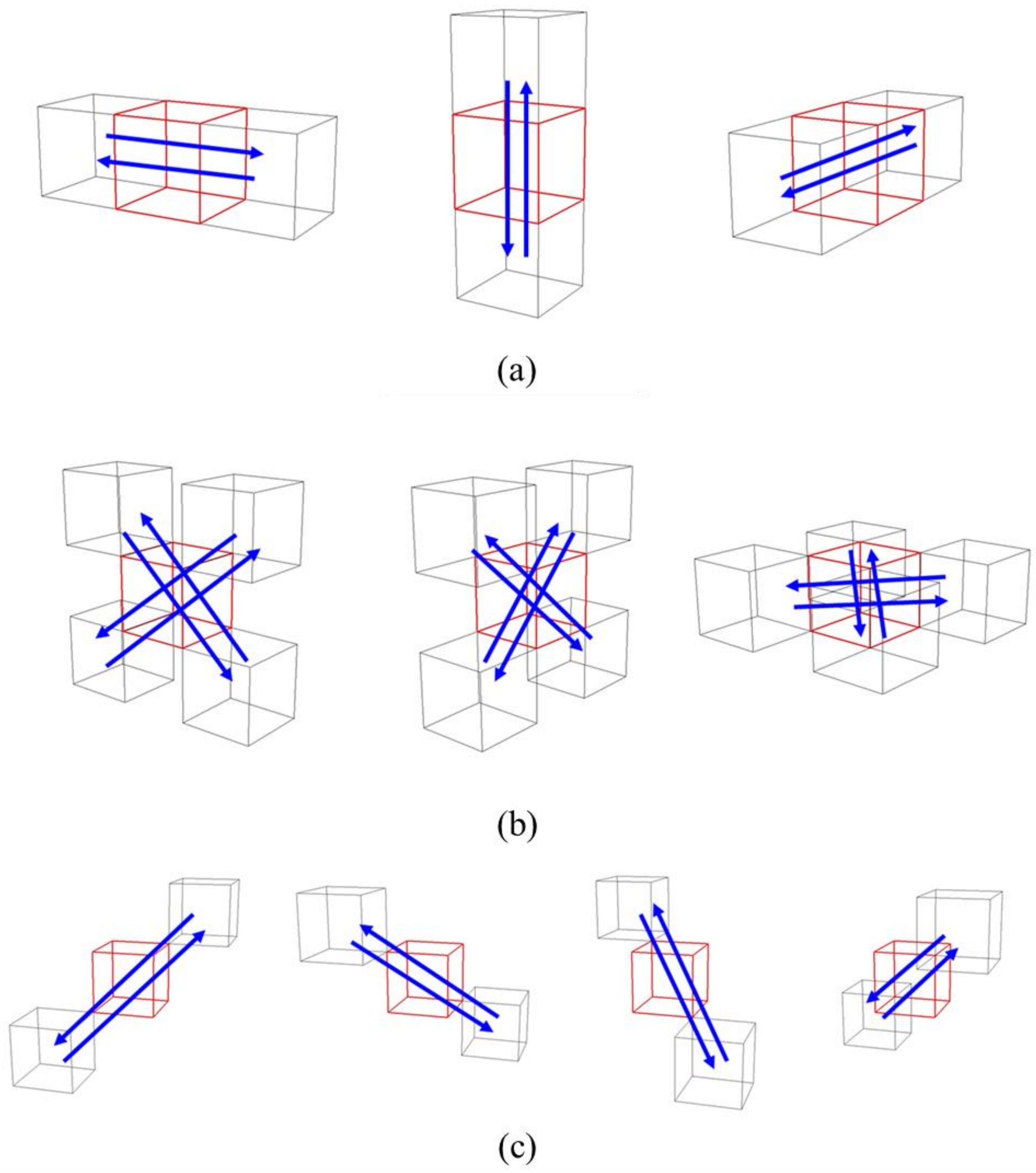


Fig. 5.4. Scenarios of MPI communications for the 3D domain. Total 26 sweeping communications are required: (a) 6 communications for face neighbors; (b) 12 communications for edge neighbors; (c) 8 communications for vertex neighbors.

5.2. Parallel multigrid method

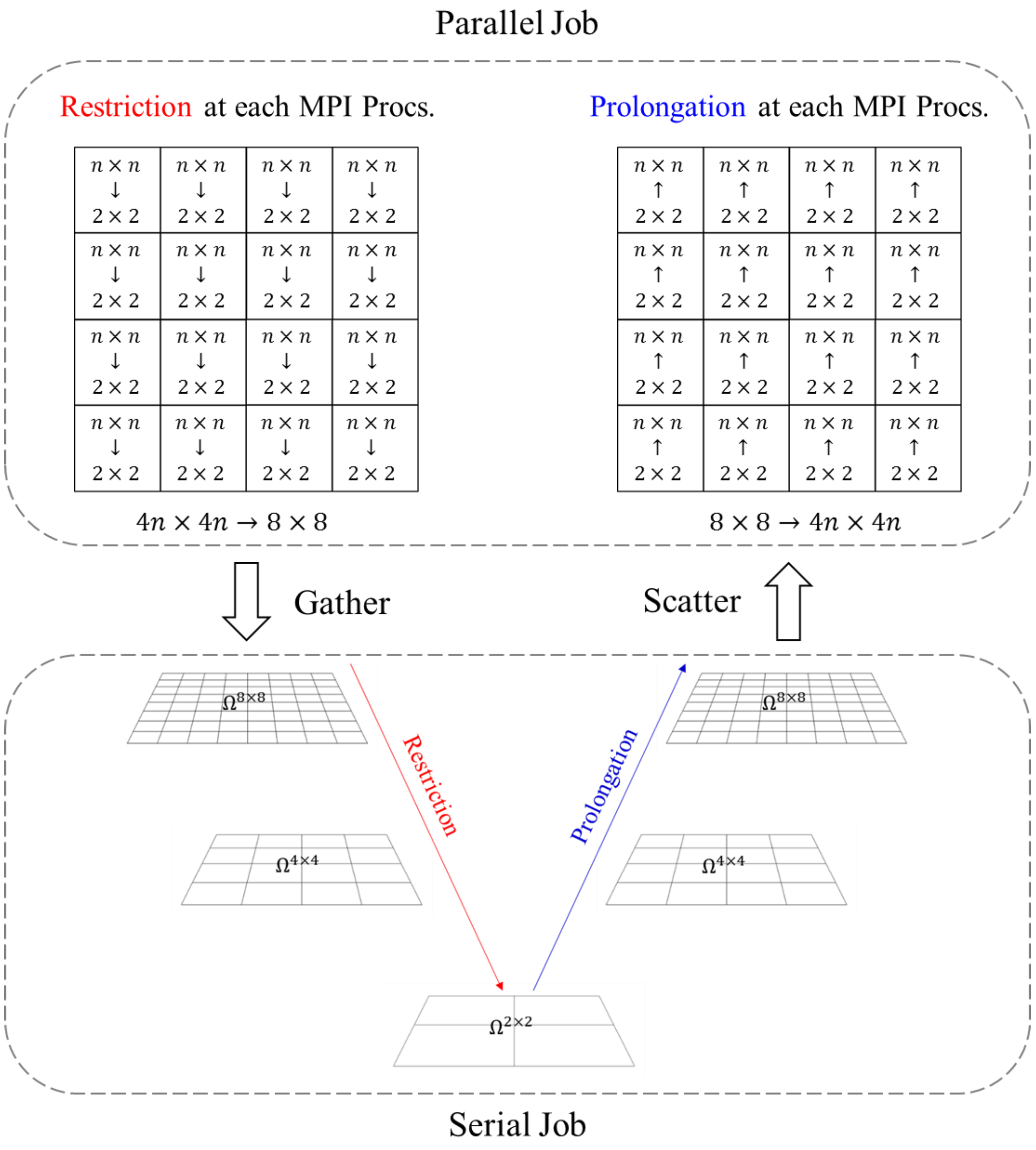
In the actual computation, the most time-consuming part of the projection method is to solve the pressure Poisson equation. For this reason, the parallel scalability of the flow simulation highly depends on the performance of the MG method in the parallel computation. In this sub-chapter, the parallelization strategy for the current MG algorithm using the Heaviside function restriction is presented.

In the current MG algorithm, the convergence speed can be enhanced as the final grid of the restriction procedure is as coarse as possible. For this reason, the coarsest grid for the restriction is set to 2×2 in the 2D square domain and $2 \times 2 \times 2$ in the 3D cube domain regardless of the finest mesh resolution in the current study. Let us consider the case where the MPI parallelization is applied. If the computational domain is divided into $n_p \times n_p$ MPI processes in 2D, the coarsest grids of the entire computational domain would be $2n_p \times 2n_p$ since the restriction procedure is carried out until there are 2×2 grids at each MPI process. For this reason, the convergence speed can decrease as the number of MPI processes increases in the parallel computation.

To overcome this disadvantage in the parallel computation, the multilevel algorithm [74] should be applied. An example of the multilevel algorithm is shown in Fig. 5.5. In this example, the computational domain, composed of a $4n \times 4n$ grid, is divided into 4×4 MPI processes. Based on Fig. 5.5, the multilevel algorithm for a single V-cycle can be summarized as follows:

1. The restriction procedure ($n \times n \rightarrow 2 \times 2$) is carried out at each MPI process.
2. The information of the coarsest grid at each MPI process is gathered into a single process.
3. The serial V-cycle procedure ($8 \times 8 \rightarrow 2 \times 2 \rightarrow 8 \times 8$) is conducted on a single process.
4. The results of the serial V-cycle are scattered to corresponding MPI processes.
5. The prolongation procedure ($2 \times 2 \rightarrow n \times n$) is performed at each MPI process.

Through this procedure, the MG method can be applied to the parallel computation without any disadvantages due to parallelization. This algorithm can be similarly applied to the 3D problem. The multilevel algorithm can be readily applied to the current multigrid algorithm. Comparing to the application of the multilevel algorithm towards the standard MG method on the regular domain, the only difference is to *gather the Heaviside function* in the aggregation process.



Parallel Job
Restriction at each MPI Procs.
Prolongation at each MPI Procs.
$n \times n \downarrow 2 \times 2$
$n \times n \uparrow 2 \times 2$
$4n \times 4n \rightarrow 8 \times 8$
$8 \times 8 \rightarrow 4n \times 4n$
Gather
Scatter
$\Omega^{8\times8}$
$\Omega^{4\times4}$
$\Omega^{2\times2}$
Restriction
Prolongation
Serial Job

Fig. 5.5. The multi-level V-cycle method for parallel multigrid method in 2D domain

5.3. KISTI`s *Nurion* supercomputer

In order to conduct parallel computation, *Nurion* supercomputer operated by Korea Institute of Science and Technology Information (KISTI) is utilized in the current study. KISTI`s *Nurion* supercomputer consists of two different nodes. One is the computation node, and the other is the CPU-only node. The computation node system is composed of 8,305 Intel Xeon Phi processors (codename : Knight Landing), and the CPU-only node system consists of 132 Intel Xeon processors (codename : Skylake). The maximum performance of *Nurion* supercomputer is 25.7 petaflops, which ranks 11th in the world by June 2018 (http://www.top500.org). The specification of the KISTI`s *Nurion* supercomputer is summarized in Table 5.1.

Table 5.1
Summary of specification of KISTI`s *Nurion* supercomputer

	Computation node	CPU-only node
Model	Cray CS500	
System structure	Xeon Phi cluster	CPU cluster
CPU type	Intel Xeon Phi 7250 1.4GHz	Intel Xeon 6148 2.4GHz
Total nodes	8,305	132
CPU per node	1	2
Cores per CPU	68	20
Total cores	564,740	5,280
Memory per node	96GB	192GB
Maximum performance	25.3PetaFlops	0.4PetaFlops

5.4. Scalability

Scalability is how the performance of a parallel computation changes as the number of processors increases and can be categorized by two types. One is strong scalability, and the other is weak scalability. Strong scalability means the parallel performance when total problem size stays as the number of processors increases, showing how much computation time for a given problem can be reduced by parallel computation. Weak scalability indicates the parallelization efficiency when the problem size increases at the same rate as the number of processors by keeping the same amount of work per processor, thus appropriateness of parallel computation for a large-scale problem can be evaluated by weak scalability.

Scalability can be assessed by two parameters, namely speed-up and efficiency. Speed-up for the parallel computation can be defined by

$$\text{Speed-up} = \frac{\text{Parallel rate}}{\text{Sequential rate}},$$

where the sequential rate refers to computation rate of the sequential program on a single core, and the parallel rate means the computation rate of the parallel program on K cores. Here, the computation rate is the number of cells divided by computation time. Therefore, ideal speed-up is the number of cores (K) applied to parallel computation. Next, efficiency can be given by

$$\text{Efficiency} = \frac{\text{Speed-up}}{\text{The number of cores } (K)},$$

thus ideal efficiency is 1 because ideal speed-up is the number of cores.

In order to evaluate strong and weak scalability of the current incompressible flow solver, a well-known 3D benchmark problem, namely flow past a sphere, is solved. The entire computational domain is set to $[-8D, 8D]^3$ where D means the diameter of the sphere, and the center of the sphere is positioned at the origin. The Reynolds number is set to 100, so the flow is laminar. The details of the boundary condition are described in Chapter 7.1.

For each scaling, computational set-up is summarized in Table 5.2. In the case of strong scaling, the number of grids is fixed to $512 \times 512 \times 512\ (\cong 134.2 \times 10^6)$, and the number of cores for computation increases from 32 to 2,048. In the case of weak scaling, the number of grids increases from $256 \times 256 \times 256\ (\cong 16.8 \times 10^6)$ to $2048 \times 2048 \times 2048\ (\cong 8.56 \times 10^9)$, and the number of cores increases 32 to 16,384, keeping the same number of grids per the core.

As depicted in Figs. 5.6 and 5.7, efficiencies of strong and weak scalability are 0.67 and 0.75, respectively. For the case of weak scalability, especially, efficiency is more than 0.7 even if the number of grids is multi-billions order. Based on these results, we confirm that the current parallelized incompressible flow solver is proper to parallel computation.

Table 5.2
Computational set-up of flow past a sphere problem for evaluating scalability

(a) Strong scalability

	Case0	Case1	Case2
Grids		$512 \times 512 \times 512\ (\cong 134.2 \times 10^6)$	
Cores	32	256	2,048

(b) Weak scalability

	Case0	Case1	Case2	Case3
Grids	$256 \times 256 \times 256$ $(\cong 16.8 \times 10^6)$	$512 \times 512 \times 512$ $(\cong 134.2 \times 10^6)$	$1024 \times 1024 \times 1024$ $(\cong 1.07 \times 10^9)$	$2048 \times 2048 \times 2048$ $(\cong 8.56 \times 10^9)$
Cores	32	256	2,048	16,384

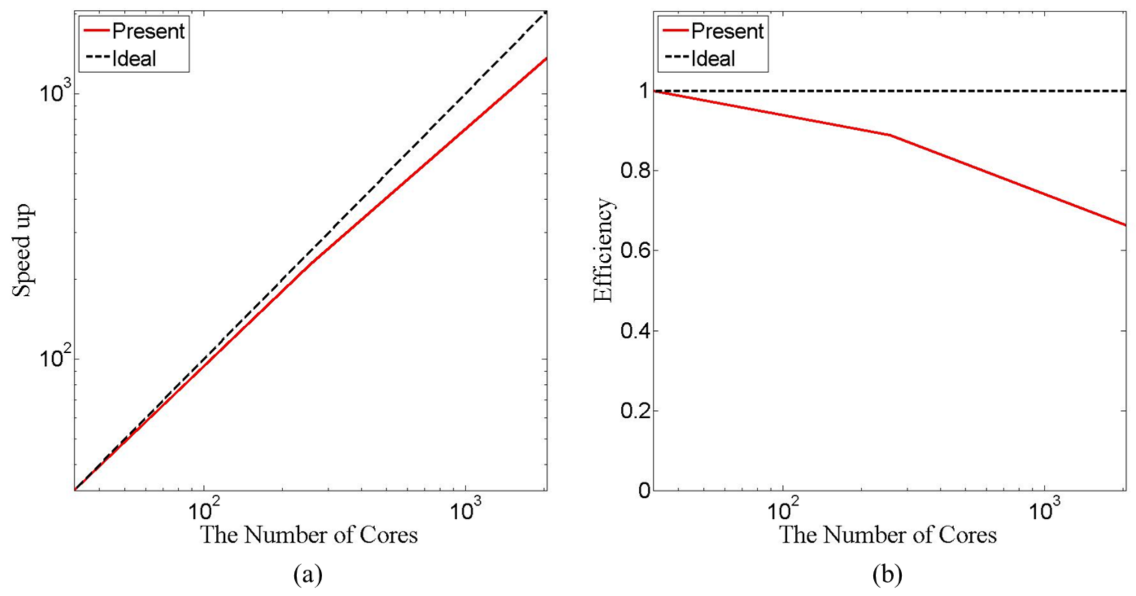


Fig. 5.6. Strong scalability of the parallel computation for flow past a sphere problem using log-log scale: (a) speed-up; (b) efficiency. Efficiency of strong scaling is 0.67.

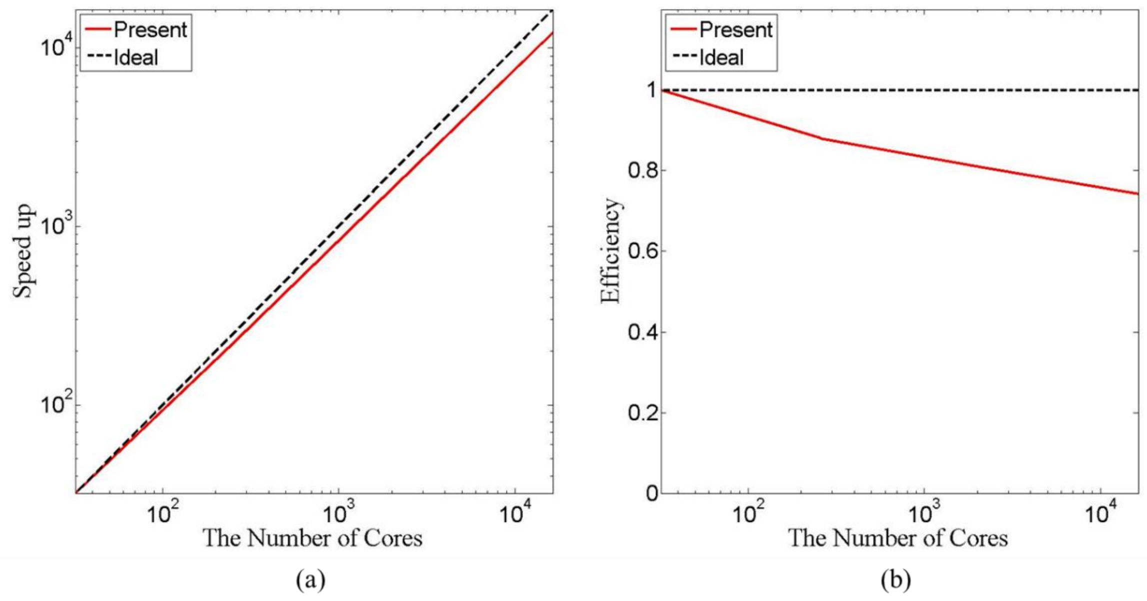


Fig. 5.7. Weak scalability of the parallel computation for flow past a sphere problem using log-log scale: (a) speed-up; (b) efficiency. Efficiency of weak scaling is 0.75.

Chapter 6

2D Simulation Results and Discussion

6.1. Verification study

The order-of-accuracy study for the 2D Taylor-Green vortex decay problem [95] is carried out to verify the current solution algorithm for the fluid. The Taylor-Green vortex is an unsteady flow of a decaying vortex in an incompressible flow regime. Since analytical solutions exist, the error can be measured to the pointwise value at any time moment. Hence, this problem is considered as a representative example to verify an unsteady flow solver. For this problem, the governing equations are nondimensionalized by a reference length L and velocity V_0, thus all physical variables presented in this sub-chapter is considered as nondimensionalized quantities. The information of these reference variables is incorporated into the Reynolds number (Re) defined as $Re = V_0 L/\nu$ where ν is the kinematic viscosity. The nondimensionalized analytical solutions of Taylor-Green vortex problem are given by

$$u = sin(x)\,cos(y)\,e^{-2t/Re}, \tag{6.1}$$

$$v = -\,cos(x)\,sin(y)\,e^{-2t/Re}, \tag{6.2}$$

$$p = \frac{1}{4}(cos(2x) + cos(2y))e^{-4t/Re}. \tag{6.3}$$

Here, Re is set to 100. The computational domain of $[0,2\pi]^2$ is used, and a periodic boundary condition is imposed at all boundaries. The initial conditions are set to $u = sin(x)\,cos(y)$ and $v = -\,cos(x)\,sin(y)$ based on the analytical solutions.

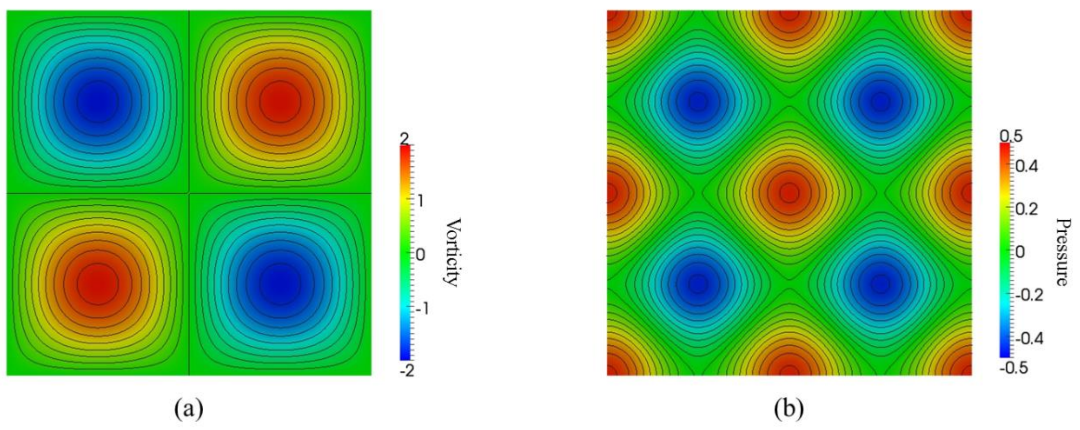


Fig. 6.1. Instantaneous results of 2D Taylor-Green Vortex problem at t=2.0: (a) vorticity contour, (b) pressure contour. Contour levels for vorticity and pressure are [-2:0.2:2] and [-0.5:0.05:0.5], respectively.

Three different accuracies are considered in this verification study: spatial accuracy; temporal accuracy; combined accuracy. Since the error computed from an unsteady problem contains the both spatial and temporal components, the spatial accuracy should be evaluated by decreasing the Δx with the constant Δt. In contrast, the temporal accuracy should be analyzed by decreasing the Δt with the fixed Δx. The combined accuracy is measured by fixing the value of $\Delta t/\Delta x$ such that Δt should decrease as the number of computational grids increases. Because current solution algorithm is unconditionally stable, this kind of thorough analysis for various accuracies can be possible.

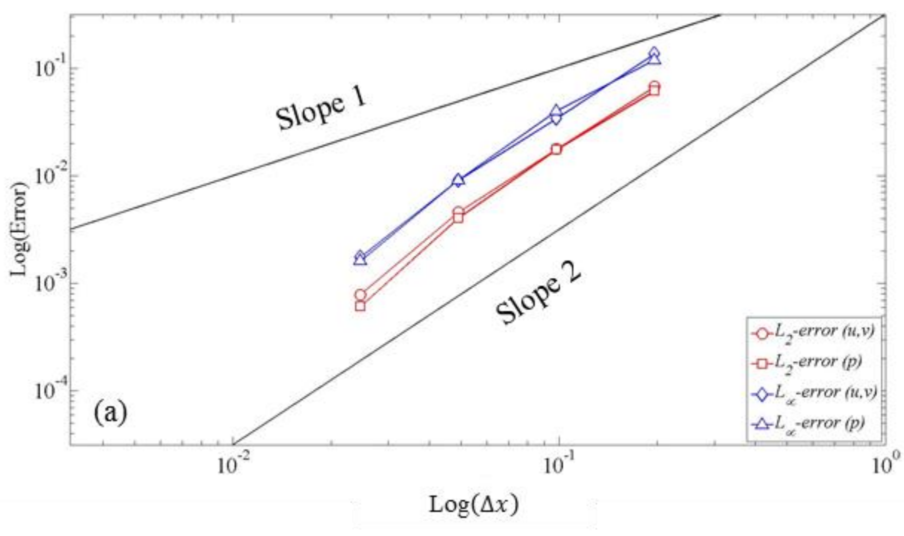
The L_2- and L_∞-errors [65] are employed to evaluate the order-of-accuracy. The L_2-error can be defined as

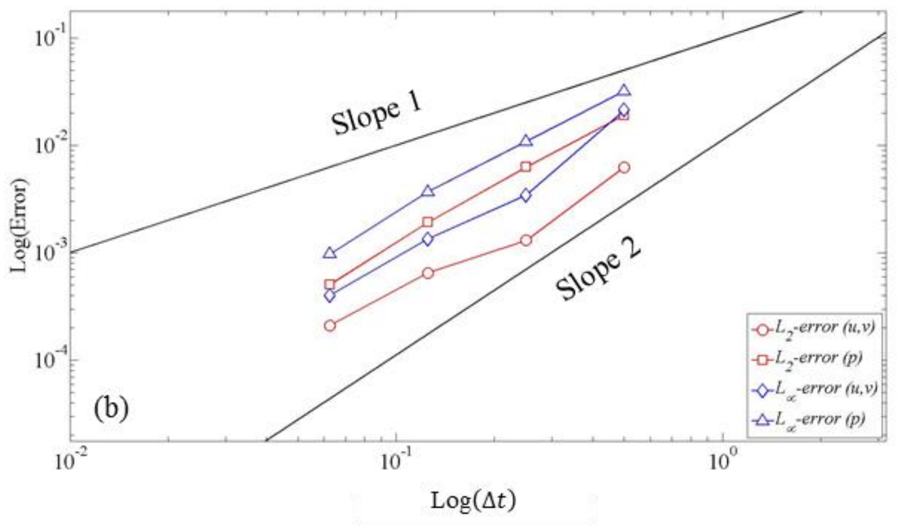
$$L_2\text{-}error = \left(\int_\Omega \left(\psi - \psi_{analytical}\right)^2 d\Omega \right)^{1/2} = \left(\sum_{i,j}^{N_{cells}} \int_{\Omega_{i,j}} \left(\psi - \psi_{analytical}\right)^2 d\Omega \right)^{1/2}, \tag{6.4}$$

where ψ is the primitive variable of the governing equations, i.e. velocity or pressure. The definition of L_∞-error norm follows Eq. (6.4) excepting that the maximum error in an absolute sense is counted. Errors are measured at time moment $t = 2.0$. At this moment, the instantaneous vorticity and pressure contours on 256×256 grid system are presented in Fig. 6.1.

Table 6.1 shows the order-of-accuracy for Taylor-Green vortex problem with respect to the three types of accuracy: (a) pure spatial accuracy; (b) pure temporal accuracy; (c) spatial/temporal combined accuracy. The spatial accuracy is measured by conducting simulations on four different Cartesian grid systems ($32^2, 64^2, 128^2, 256^2$) with constant $\Delta t = 0.1$. The temporal accuracy is analyzed by simulating Taylor-Green vortex problem on very fine mesh (1024×1024) with four different time steps, which are Δt = 0.5, 0.25, 0.125, and 0.0625. In order to assess the combined accuracy, simulations are conducted on four different uniform Cartesian grid systems ($32^2, 64^2, 128^2, 256^2$) with constant $\Delta t/\Delta x = 5.09$. Corresponding time steps are 1, 0.5, 0.25, and 0.125.

As shown in Table 6.1, the spatial and combined errors are well converged to analytical solution with the second-order rate. The pure temporal order-of-accuracy, however, is slightly inferior to the second-order rate. As described in [104], decreasing Δt with fixed mesh-size could incur a slightly sub-optimal accuracy. Nevertheless, for the spatial/temporal combined order-of-accuracy, which is the typical strategy of unsteady flow simulation, second order-of-accuracy is clearly observed for all flow variables. Fig. 6.2 depicts the results of accuracy analysis using a log-log graph. The velocity is represented by only one symbol since the convergence rate of u and v are measured to be exactly same.





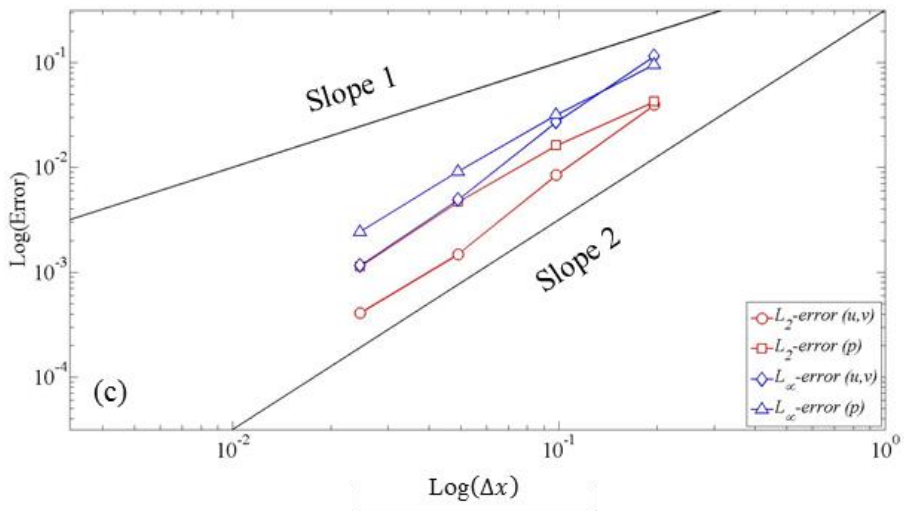


Fig. 6.2. Verification study for the 2D Taylor-Green vortex problem: (a) pure spatial error with fixed Δt; (b) pure temporal error with fixed Δx; (c) spatial/temporal combined error with fixed $\Delta t/\Delta x$.

Table 6.1
The results of verification study for the Taylor-Green vortex problem

(a) Pure spatial accuracy : The number of grids increases with fixed Δt

Grid	$\frac{\Delta t}{\Delta x}$	Velocity (u, v)				Pressure			
		L_2	Rate	L_∞	Rate	L_2	Rate	L_∞	Rate
32^2	0.51	6.75E-02		1.36E-01		6.18E-02		1.19E-01	
64^2	1.02	1.77E-02	1.93	3.42E-02	1.99	1.76E-02	1.81	4.00E-02	1.57
128^2	2.04	4.60E-03	1.94	9.12E-03	1.91	4.06E-03	2.12	9.13E-03	2.13
256^2	4.08	7.87E-04	2.55	1.75E-03	2.38	6.08E-04	2.74	1.62E-03	2.49

(b) Pure temporal accuracy : The size of time step decreases with fixed Δx

Δt	$\frac{\Delta t}{\Delta x}$	Velocity (u, v)				Pressure			
		L_2	Rate	L_∞	Rate	L_2	Rate	L_∞	Rate
0.5	81.5	6.26E-03		2.13E-02		1.91E-02		3.19E-02	
0.25	40.7	1.31E-03	2.26	3.44E-03	2.63	6.30E-03	1.60	1.09E-02	1.55
0.125	20.4	6.49E-04	1.01	1.35E-03	1.35	1.92E-03	1.71	3.70E-03	1.56
0.0625	10.2	2.11E-04	1.62	4.02E-04	1.75	5.07E-04	1.92	9.74E-04	1.93

(c) Spatial/temporal Combined accuracy : The number of grids increases with fixed $\Delta t/\Delta x$

Grid	$\frac{\Delta t}{\Delta x}$	Velocity (u, v)				Pressure			
		L_2	Rate	L_∞	Rate	L_2	Rate	L_∞	Rate
32^2	5.09	3.94E-02		1.15E-01		4.26E-02		9.58E-02	
64^2	5.09	8.47E-03	2.22	2.72E-02	2.08	1.62E-02	1.39	3.18E-02	1.59
128^2	5.09	1.49E-03	2.51	4.94E-03	2.46	4.69E-03	1.79	9.18E-03	1.79
256^2	5.09	4.09E-04	1.86	1.16E-03	2.09	1.14E-03	2.04	2.44E-03	1.91

6.2. Flow past a circular cylinder

The computational domain of the flow past a circular cylinder problem is set as shown in Fig. 6.3. The total domain size is $[-8D, 24D] \times [-8D, 8D]$, where D is the diameter of the circular cylinder, and the center of the cylinder is located at the origin. We impose the Dirichlet boundary condition of $\boldsymbol{U} = (u_\infty, 0)$ at the left, the slip boundary condition at the top and bottom, the convective boundary condition (a convection velocity is the average value of the right boundary) at the right, and the no-slip wall boundary condition at the cylinder surface. For the pressure, the Dirichlet boundary condition is imposed on the outlet as $p_\infty = 0$, and other boundaries are set to the Neumann condition. Since an equally spaced 1024×512 Cartesian mesh system is used in the entire computational domain, 32×32 cells cover the $D \times D$ domain. For all simulations presented in this sub-chapter, the Reynolds number (Re) is based on the cylinder diameter D and the free-stream velocity u_∞, and the CFL number based on the free-stream velocity u_∞ is set to 4.

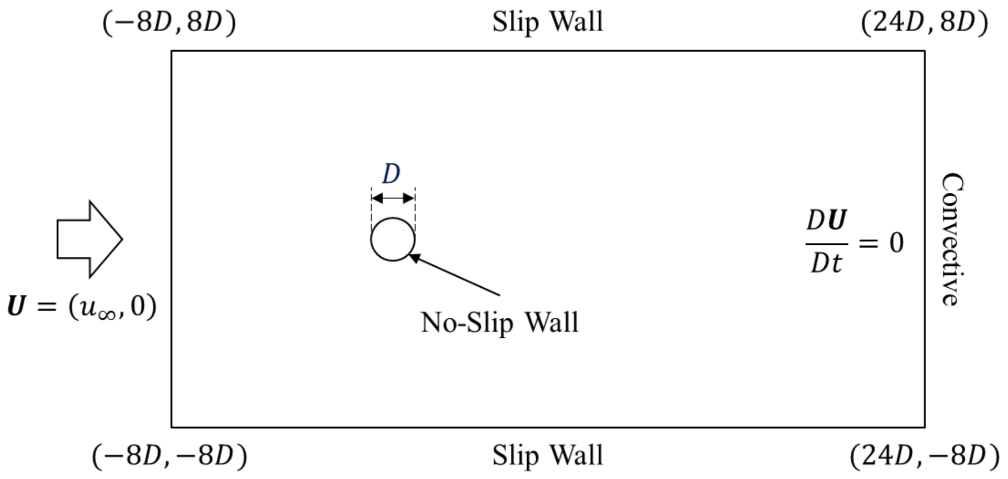


Fig. 6.3. Layout of the computational domain for flow past a circular cylinder problem.

6.2.1. Steady flow around a circular cylinder: The effect of the Heaviside function for the pressure boundary condition

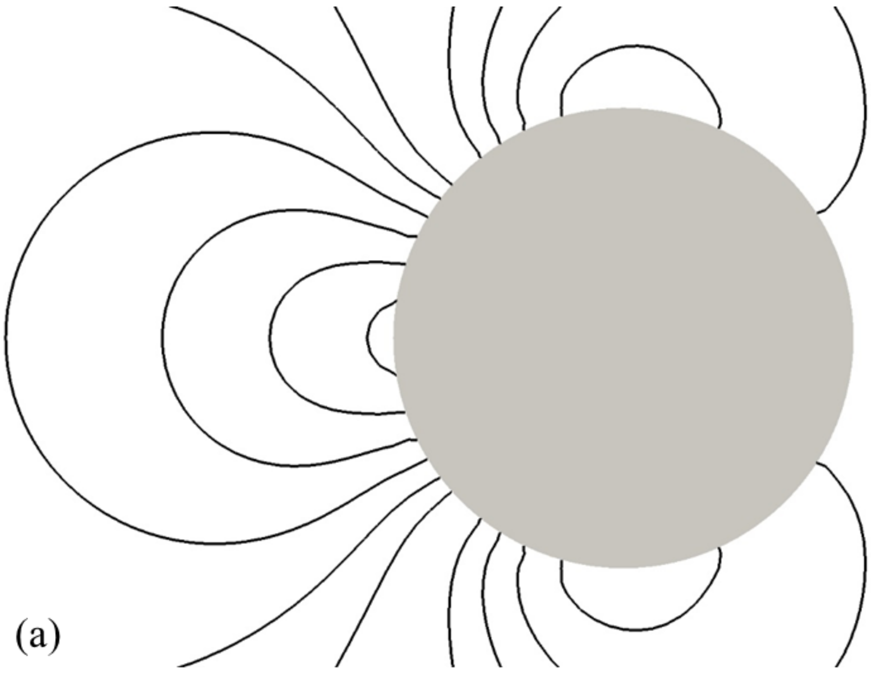
The effect of the Heaviside function for the pressure Neumann boundary condition is investigated by solving the pressure Poisson equation in two different ways. One is the current solution algorithm with the Heaviside function, and the other is a conventional algorithm without the Heaviside function, where the pressure Neumann boundary condition is not guaranteed. To simply distinguish these two approaches, the former algorithm is defined as "$A1$", and the latter one is considered as "$A2$". For this particular purpose, Reynolds numbers of 20 and 40 are considered because of the simulation results in the representative steady solution. As shown in Table 6.2, $A1$ predicts the drag coefficient well compared to previous numerical results, while $A2$ produces a relative overestimation.

The effect of the Heaviside function becomes more evident by the comparison of the local pressure coefficient distribution near the cylinder surface. Fig. 6.4 illustrates the pressure coefficient contours around the circular cylinder at $Re = 40$. In the contours of $A1$ shown in Fig. 6.4-(a), the normality of the pressure contours is achieved well everywhere on the cylinder surface. In contrast, the $A2$ scheme generates abnormal contours near the leading stagnation point as well as on the top and bottom lateral sides. Fig. 6.5 shows the distribution of wall pressure coefficients along the cylinder surface. For both algorithms, the overall trend follows the reference result [90] well. In the case of $\mathrm{A2}$, however, discrepancies are observed around the region of $30° \leq \theta \leq 90°$.

Through the comparative study presented in this sub-chapter, it is numerically confirmed that the pressure Neumann boundary condition can be accurately imposed at the cylinder surface by the Heaviside function. In addition, the importance of the pressure boundary condition in the flow simulation is demonstrated.

Table 6.2
Drag coefficients of flow past a circular cylinder at $Re = 20$ and 40

	$Re = 20$	$Re = 40$
Calhoun [19]	2.19	1.62
Russel and Wang [81]	2.13	1.60
Present $A1$	2.17	1.60
Present $A2$	2.30	1.66



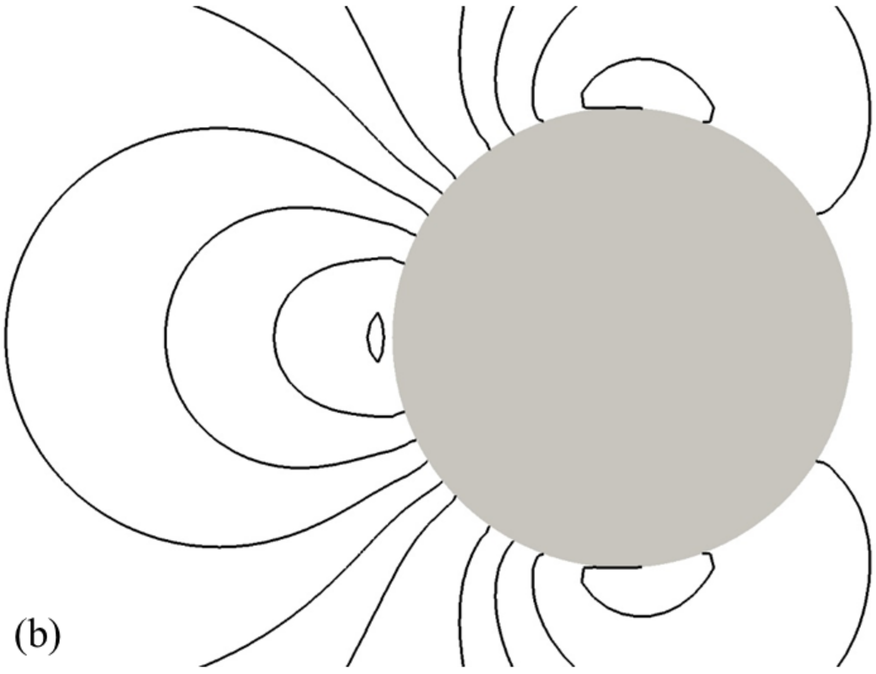


Fig. 6.4. Contours of the pressure coefficient: (a) current algorithm with the Heaviside function considered as $A1$ in this paper; (b) current algorithm without the Heaviside function defined as $A2$ in this paepr. Contour levels for the pressure coefficient are $[-0.8: 0.2: 1.2]$.

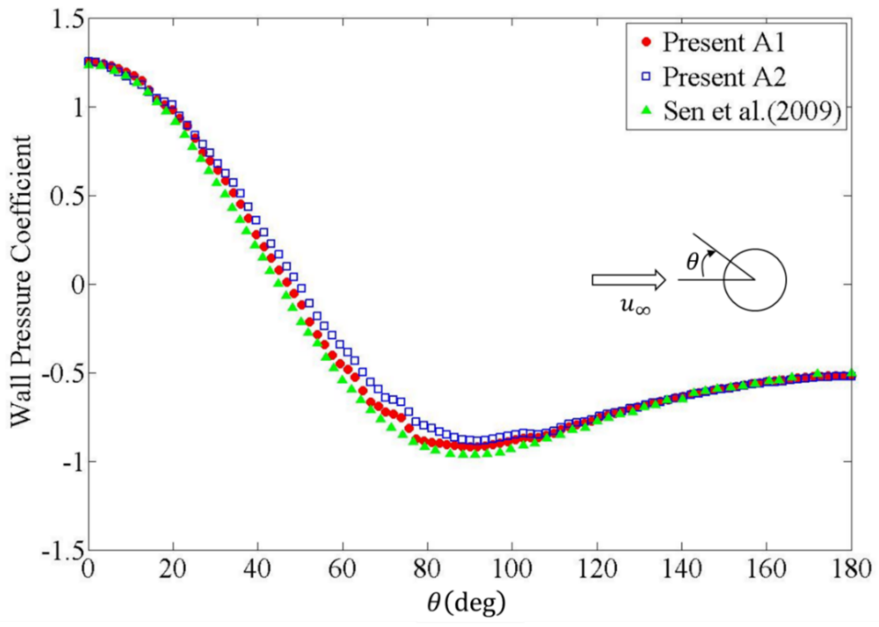


Fig. 6.5. Wall pressure coefficients along the cylinder surface at $Re = 40$. Sen et al. [90] is the numerical results based on the body-fitted mesh system.

6.2.2. Unsteady flow around a circular cylinder

In this sub-chapter, the simulations results at $100 \leq Re \leq 200$ are presented. Fig. 6.6 depicts the vorticity contours at $Re = 100$ and 200. The size of the zoomed-in view shown in Fig. 6.6 is $[-D, 18D] \times [-3D \times 3D]$**.** The time histories of the drag and lift coefficients at $Re = 100$ and 200 are shown in Fig. 6.7. Here, the drag coefficient (C_D) and lift coefficient (C_L) are computed as

$$C_D = \frac{F_D}{\frac{1}{2}\rho u_\infty^2 D} \text{ and } C_L = \frac{F_L}{\frac{1}{2}\rho u_\infty^2 D}, \tag{6.5}$$

where F_D is the drag force and F_L is the lift force. Recall that since $\nabla H = \boldsymbol{n}\delta_\Gamma$, the force vector $\boldsymbol{F} = (F_D, F_L)$ acting on the circular cylinder can be simply computed as

$$\boldsymbol{F} = \int_\Omega (-p\boldsymbol{I} + 2\mu\boldsymbol{D}) \cdot \nabla H \, d\Omega, \tag{6.6}$$

Here, $\boldsymbol{I}$ is the identity tensor and $\boldsymbol{D}$ is the strain rate tensor of the fluid velocity. In the Cartesian mesh-based method, this kind of force calculation is difficult because the body boundary does not coincide with the grid line. However, once the Heaviside function is defined, the hydrodynamic forces can be easily computed with second-order accuracy using the current approach.

Table 6.3
Drag and lift coefficients of current and previous results

	$Re = 100$		$Re = 200$	
	C_D	C_L	C_D	C_L
Choi et al. [21]	1.34 ± 0.011	± 0.315	1.36 ± 0.048	± 0.64
Ng et al. [72]	1.37 ± 0.016	± 0.36	1.37 ± 0.050	± 0.72
Liu et al. [60]	1.35 ± 0.012	± 0.339	1.31 ± 0.049	± 0.69
Braza et al. [10]	1.36 ± 0.015	± 0.25	1.40 ± 0.050	± 0.75
Present	1.34 ± 0.012	± 0.37	1.40 ± 0.050	± 0.73

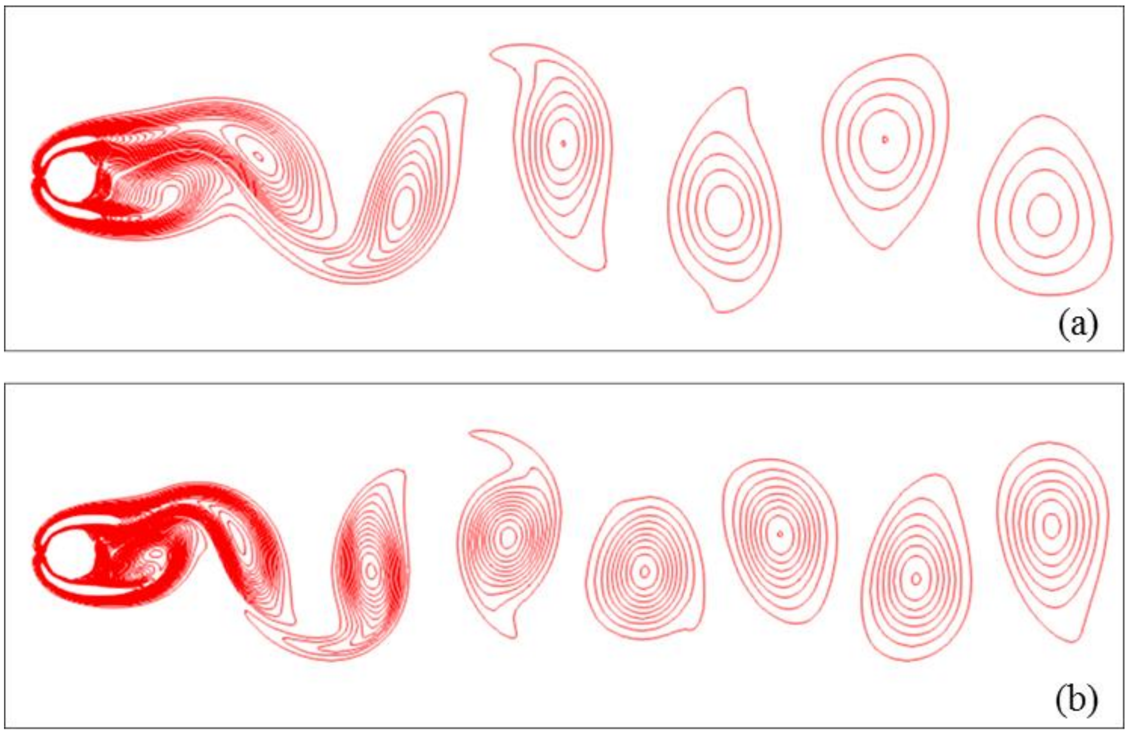


Fig. 6.6. Instantaneous vorticity contours: (a) $Re = 100$; (b) $Re = 200$. The size of the captured domain is $[-D, 18D] \times [-3D \times 3D]$. Contours are presented from $-4u_\infty/D$ to $4u_\infty/D$ with 41 intervals.

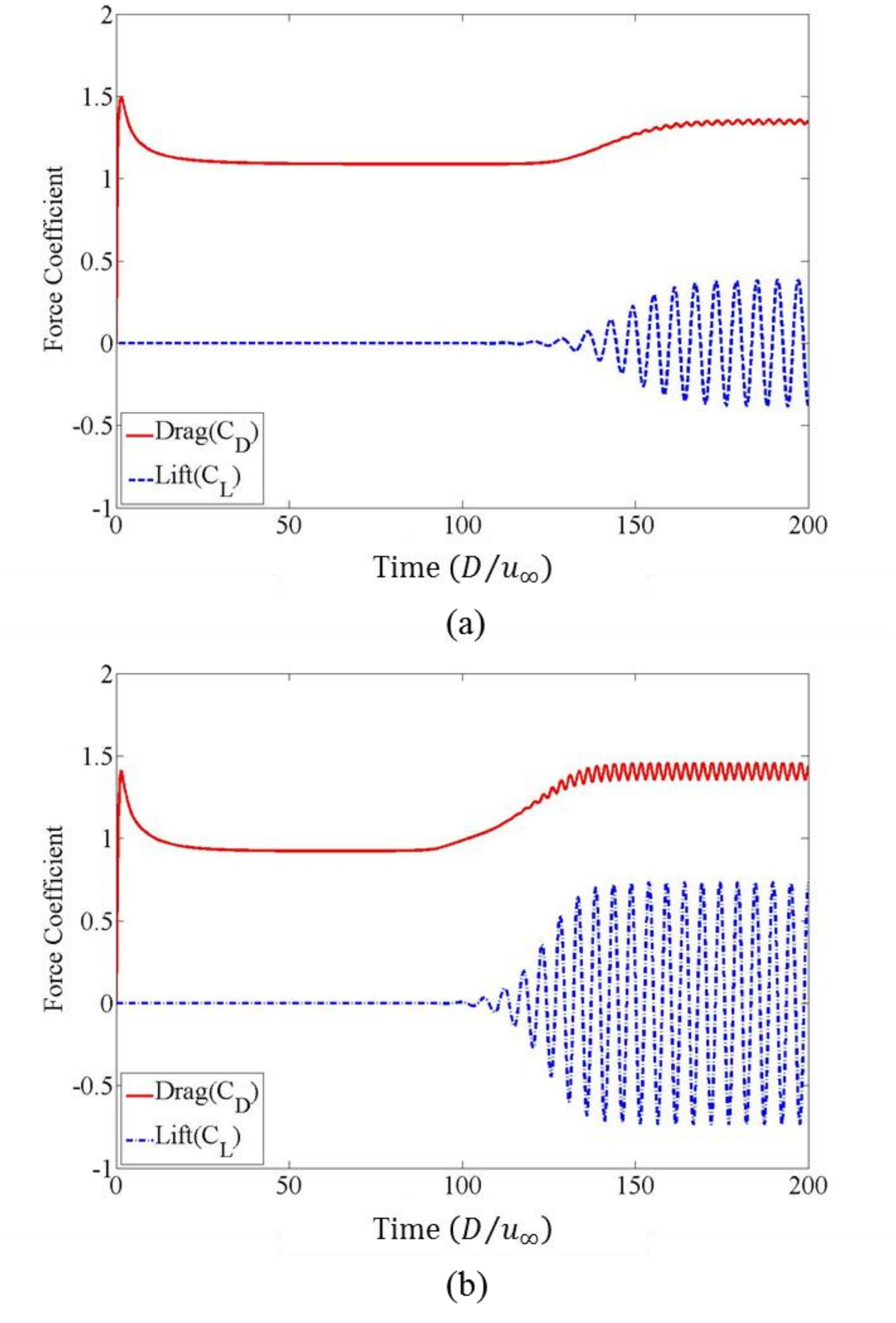


Fig. 6.7. Time history of drag and lift coefficient: (a) $Re = 100$; (b) $Re = 200$. The solid line refers to the drag coefficient, and the dotted line indicates lift coefficient.

To demonstrate the validity of the simulation results, a quantitative comparison of unsteady C_D and C_L with previous results is presented in Table 6.3. Since C_D and C_L oscillate due to vortex shedding, both are described by using the mean value and amplitude. Table 6.3 shows that the current numerical results are in good agreement with other studies.

Next, the Strouhal number St is compared at low Reynolds numbers (< 200). The Strouhal number is an important parameter in many engineering fields, especially for designing slender circular structures, such as deep-sea risers, because Strouhal number is directly related to the excitation frequency of the vortex-induced-vibration (VIV). The Strouhal number is defined as

$$St = \frac{fD}{u_\infty}, \qquad (6.7)$$

where f is the shedding frequency, which is same as the frequency of the lift oscillation or half of the drag oscillation. Results are compared with experimental and validated numerical data. Fig. 6.8 shows that the present results are well matched to the results of other studies.

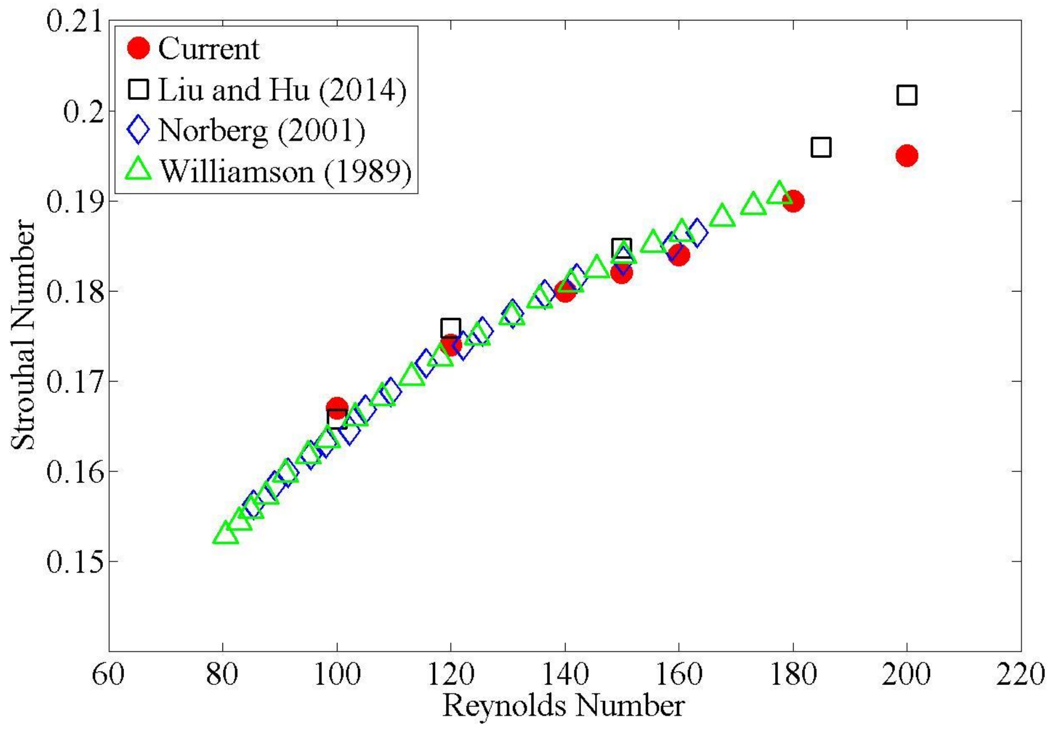


Fig. 6.8. Comparison results of the Strouhal number versus the Reynolds number. Liu and Hu [63] is result of numerical simulation. Norberg [73] and Williamson [99] are experimental results.

6.3. Flow past a NACA0012 airfoil

In this sub-chapter, simulation results of NACA0012 airfoil with various angles of attack are presented. Since there is no analytic form of the SDF for NACA0012, AMR-based determination methods of the SDF are applied. Fig. 6.9 shows the layout of the computational domain together with AMR results for the airfoil at the zero angle of attack. The size of the domain is $[-2c, 6c] \times [-2c, 2c]$ where c is the chord length of NACA0012, and the nose of the airfoil is located at $(0,0)$. A 1024×512 uniform Cartesian mesh system is used, thus the rectangular region covering the foil consists of 128×12 uniform Cartesian cells. As depicted in Fig. 6.9, boundary conditions are set equal to the previous cylinder problem.

All simulations are conducted at the fixed value of $Re = 1000$ based on the chord length c and the free stream velocity u_∞, and the CFL number based on the free-stream velocity u_∞ is set to 4. The angle of attack α is varied about the specific point that is half of chord the length. Simulations are performed for a range of α from $0°$ to $20°$. For each case, the SDF is independently computed with the AMR as depicted in Fig. 6.10.

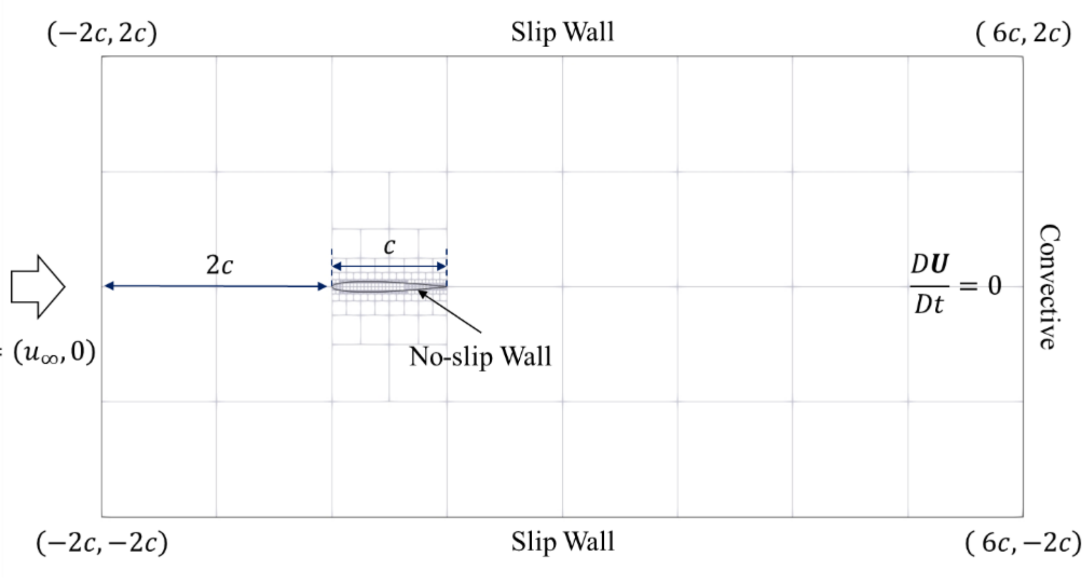


Fig. 6.9. Computational domain for flow past NACA0012. The angle of attack for this case is zero. The adaptive mesh structure refined from the coarsest background mesh indicates the successful automatic detection of the foil shape and the optimal refinement utilized for the SDF computation. Note that there is no refinement needed for the cells adjacent to the leading and trailing edge of the foil.

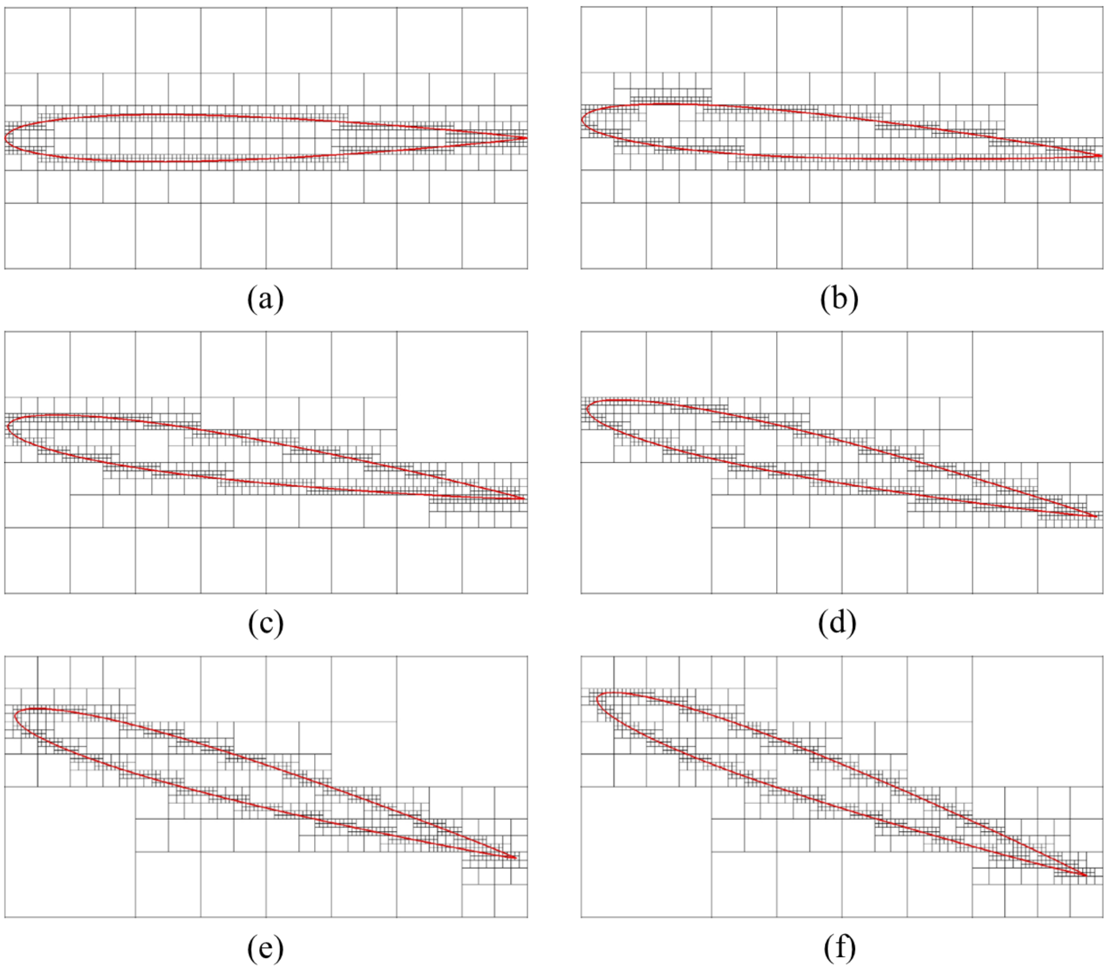


Fig. 6.10. Close-up view of adaptive mesh refinement near NACA0012 for computing SDF: (a) $\alpha = 0°$; (b) $\alpha = 4°$; (c) $\alpha = 8°$; (d) $\alpha = 12°$; (e) $\alpha = 16°$; (f) $\alpha = 20°$. The size of captured domain is $[0, c] \times [-0.25c, 0.25c]$.

Fig. 6.11 illustrates the instantaneous vorticity fields and contours for six different angles of attack: $\alpha = 0°,\ 4°,\ 8°,\ 12°,\ 16°$, and $20°$. Since the narrow wake region is symmetric, stable, and steady at $\alpha = 0°$, vortex shedding is not observed. However, for $\alpha \geq 8°$, the flow in the wake region becomes asymmetric and unstable, and unsteady vortex shedding is produced. As α increases, the separation region becomes larger, and stronger and wide vortex shedding is observed. This behavior is in good agreement with previously reported results [42, 57].

The airfoil has the typical features of an engineering object used for fluid machinery, namely a smooth streamlined low curvature part and also sharp corners. It is important to be able to accurately represent the geometric complexity of the body to predict the flow phenomena over such an engineering object. Therefore, based on the presented simulation results, the current AMR-based computation method for SDF is a promising approach that can be applied to various engineering objects on a Cartesian mesh.

Fig. 6.12 shows the drag and lift coefficients versus the angle of attack. For the case of an airfoil, the reference length is considered as the chord length when computing force coefficients. Current simulation results are compared with the results from the body-fitted mesh based simulations [42, 57, 61]. Although a minor discrepancy for drag and lift coefficient is observed at $\alpha = 16°$, the overall tendency is similar to previous results.

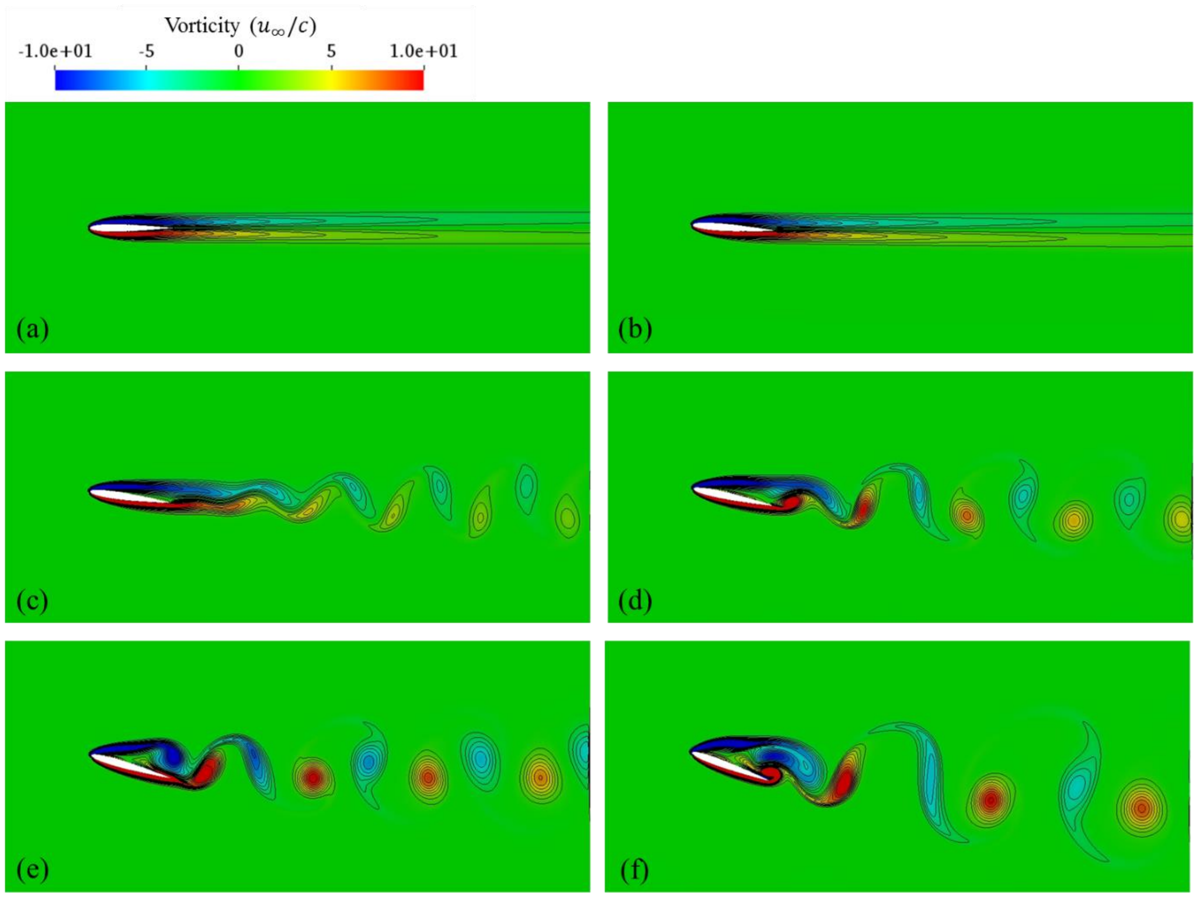
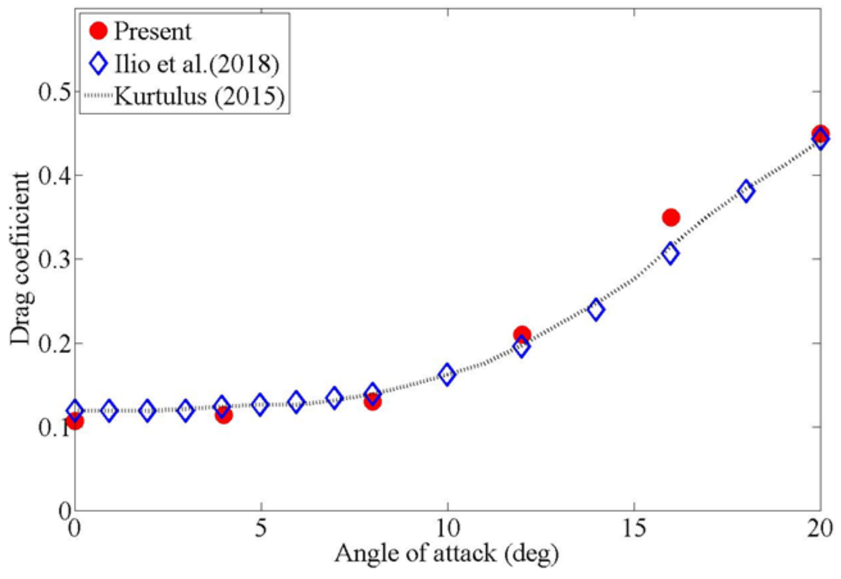
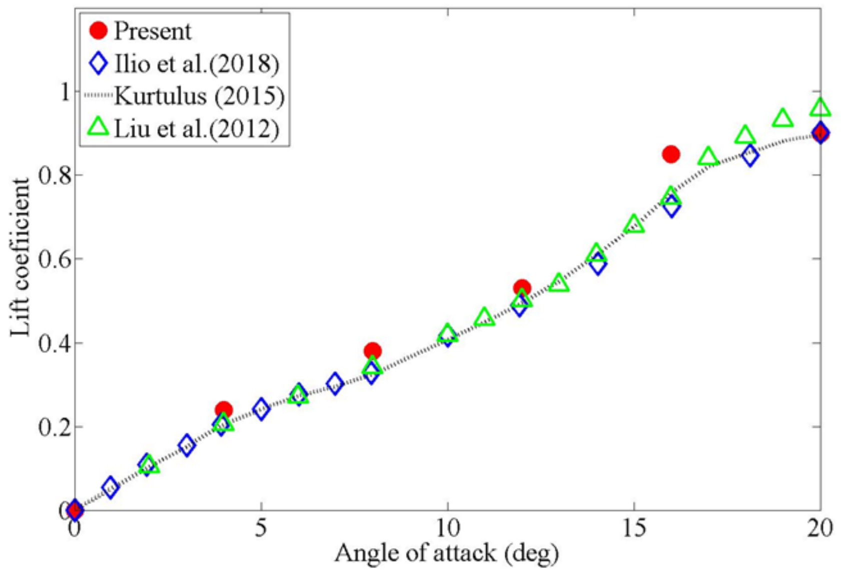


Fig. 6.11. Instantaneous vorticity fields and contours(a) $\alpha = 0°$; (b) $\alpha = 4°$; (c) $\alpha = 8°$; (d) $\alpha = 12°$; (e) $\alpha = 16°$; (f) $\alpha = 20°$. Contours are presented from -$10u_\infty/c$ to $10u_\infty/c$ with 21 intervals. The captured domain is $[-c, 6c] \times [-1.5c, 1.5c]$.



(a)

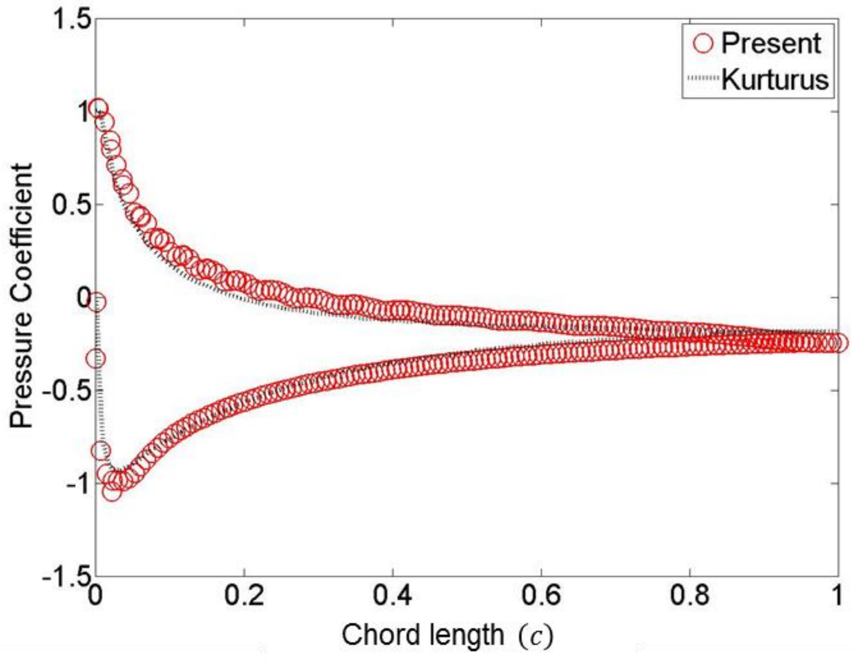


(b)

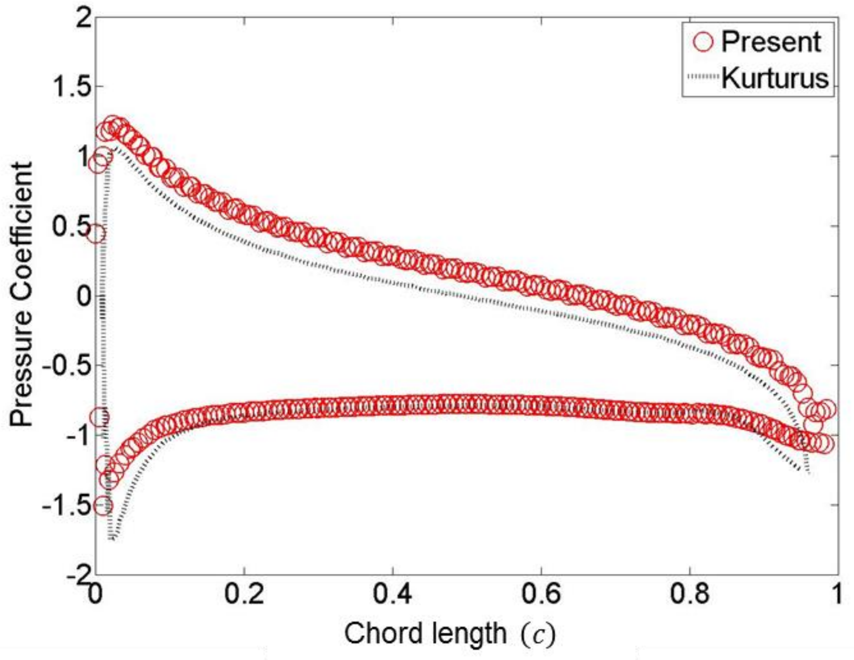
Fig. 6.12. (a) Mean drag and (b) lift coefficients versus the angle of attack. Present results (●) are compared with the Ilio et al. (2018) [42] (◇), Kurtulus (2015) [57] (···), and Liu et al. (2012) [61] (△). All reference data is obtained from the body-fitted mesh based simulation.

The mean pressure coefficient along the airfoil surface is compared at specific α: $\alpha = 8°$ and $19°$. The higher pressure is acting on the bottom surface of the airfoil as α increases, so the lower and upper lines depicted in Fig. 6.13 indicate a pressure distribution acting on the top and bottom surface of the airfoil, respectively. Unlike the body-fitted mesh simulation, the airfoil surface does not coincide with grid points. This could result in additional difficulties in computing the local pressure distribution on the body surface. However, this local pressure can be simply traced by finding cells whose $|\nabla H| \neq 0$ in the current approach where $|\nabla H|$ is non-zero at the interface cells. This can be considered as an additional benefit of employing the Heaviside function for the body representation.

As shown in Fig. 6.13, the current results are well matched to validated data at $\alpha = 8°$, although slight discrepancies are observed at $\alpha = 19°$. The overall pressure distribution on the bottom surface (upper line) is slightly overestimated, but the magnitude compared to the absolute pressure value is small and consistent throughout the surface. For the top surface of the airfoil (lower line), minor discrepancies observed at the nose and tailing edge could be attributed to the violent flow separation. Both discrepancies can be improved by the better mesh resolution, wider flow domain, or more sophisticated boundary condition imposition, which are not on the main focus of this paper.



(a)



(b)

Fig. 6.13. Distribution of mean pressure coefficients along NACA0012 surface including both the top and bottom region: (a) $\alpha = 8°$; (b) $\alpha = 19°$. Reference data indicated by a dotted-line $(\cdots)$ are computed by Kurtulus [57] using a body-fitted mesh system.

6.4. Flow over an oscillating cylinder

The oscillating cylinder problem at the low Reynolds number (Re) and Keulegan-Carpenter number (KC) introduced by Dütsch et al. [24] is a very famous test case for validating moving body simulations because comparable and reliable experimental data exist. In this problem, the motion of the cylinder is described by a sinusoidal harmonic motion as follows:

$$x(t) = -A\, sin(2\pi f t), \tag{6.8}$$

where $x(t)$ is the x-directional position of the cylinder`s center with respect to time, and A and f are the amplitude and the frequency of oscillation motion, respectively. The corresponding x-directional velocity of the cylinder is given by

$$V(t) = -2\pi f A\, cos(2\pi f t) = -V_0\, cos(2\pi f t), \tag{6.9}$$

where V_0 is the maximum x-directional velocity of the cylinder.

The Reynolds number (Re) and Keulegan-Carpenter number (KC), which are key parameters to determine the flow characteristics induced by the oscillatory motion of the cylinder, are defined as

$$Re = \frac{V_0 D}{\nu} \text{ and } KC = \frac{V_0}{Df}, \tag{6.10}$$

where ν is the kinematic viscosity of the fluid, and D is the diameter of the cylinder. Here, the Re and KC numbers are set to 100 and 5, respectively.

The entire computational domain of the oscillating cylinder problem is shown in Fig. 6.14. The total domain size is $[-8D, 8D] \times [-8D, 8D]$, and the cylinder is located at the center of the domain. We impose the no-slip wall boundary condition, i.e. $\boldsymbol{U} = \boldsymbol{U}_b$ and $\partial p/\partial \boldsymbol{n} = 0$, at the cylinder surface, and the outflow boundary condition, i.e., $\partial \boldsymbol{U}/\partial \boldsymbol{n} = 0$ and $p = p_\infty$, at the rest of the boundary. Based on the author`s experience, at least 32 grids along the diameter are required to represent the cylinder shape in the uniform Cartesian grid system. To satisfy this requirement, the 512×512 uniform Cartesian grids are used. The CFL number, based on the maximum x-directional velocity of the cylinder V_0, is set to 0.67. Note that this CFL condition is determined only by the accuracy requirement, not by the stability because the current solution algorithm is unconditional stable for moving body simulation.

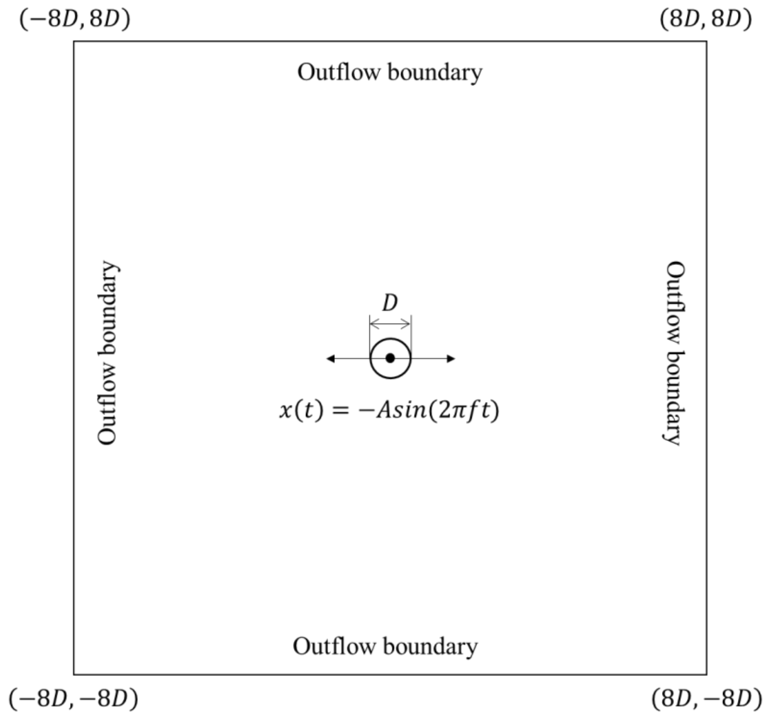


Fig. 6.14. The layout of the entire computational domain for the oscillating cylinder problem

The exact form of the SDF for the cylinder exists, thus the information of the cylinder shape at every time step can be exactly expressed in the fluid solver without additional computational cost. However, to compare simulation results of the exact SDF with transformed SDF, we numerically compute the SDF of the cylinder at the initial position by using the AMR-based SDF computation presented in Chapter 4.1, and the SDF at each time step is determined by transforming the initial SDF (see Chapter 4.7) without any re-computation for the in/out test and the shortest distance. The cylinder`s boundary is represented by 128 line-segments, and p-refinement ($p = 2$) is applied for biquadratic interpolation.

The time history of the drag coefficient (C_d), which is defined as $F_d/(0.5\rho V_0^2 D)$ where F_d is the drag force, is presented in Fig. 6.15. As shown in this figure, the drag coefficients computed from the exact SDF and the transformed SDF are almost the same and are visually indistinguishable, and these results are in good agreement with the experimental results [24]. Based on these results, we confirm that the current simulation using the transformed SDF can be applied to the moving body simulation.

The instantaneous contours of the vorticity and pressure coefficient at four different phases ($2\pi f t = 0°, 96°, 192°, 288°$) of the oscillation are presented in Fig. 6.16. These results are obtained by utilizing the transformed SDF. The patterns of both contours are very similar to the previous numerical results [21, 22, 58, 63, 106, 107].

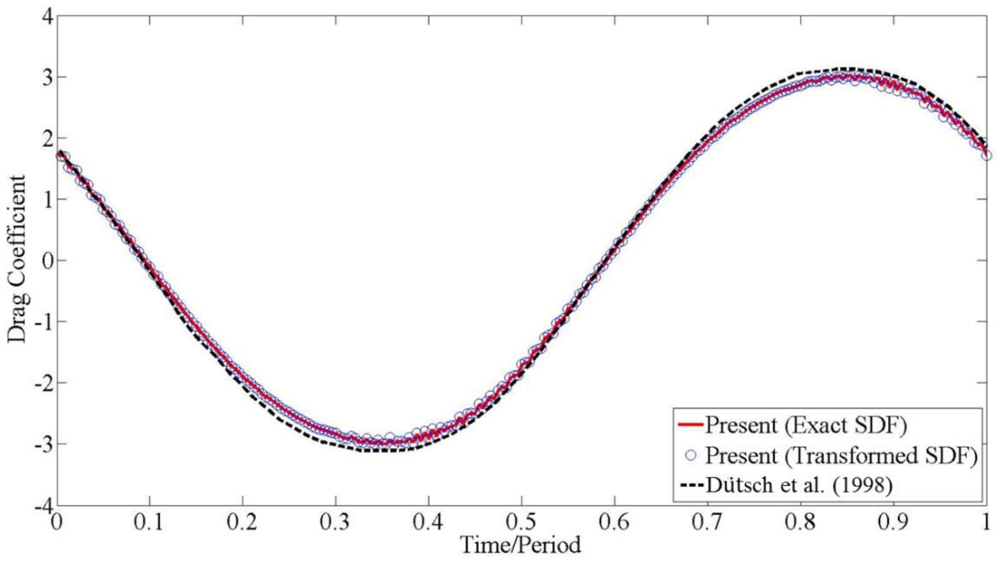


Fig. 6.15. The time history of the drag coefficient within a single oscillation period. The reference data of Dütsch et al. [24] is based on a water tank experiment.

Fig. 6.17 shows the velocity profiles of the x-directional velocity u and the y-directional velocity v along the vertical direction at four different locations ($x = -0.6D, 0, 0.6D, 1.2D$) for three different phases ($2\pi f t = 180°, 210°, 330°$) of the oscillation. The present simulation results are based on the transformed SDF and show good agreement with the experimental results [24]. Although there are slight discrepancies, we believe that these differences are due to the limitation of the simplified 2D simulation because similar patterns are observable in all previous 2D numerical results [21, 22, 58, 63, 106, 107].

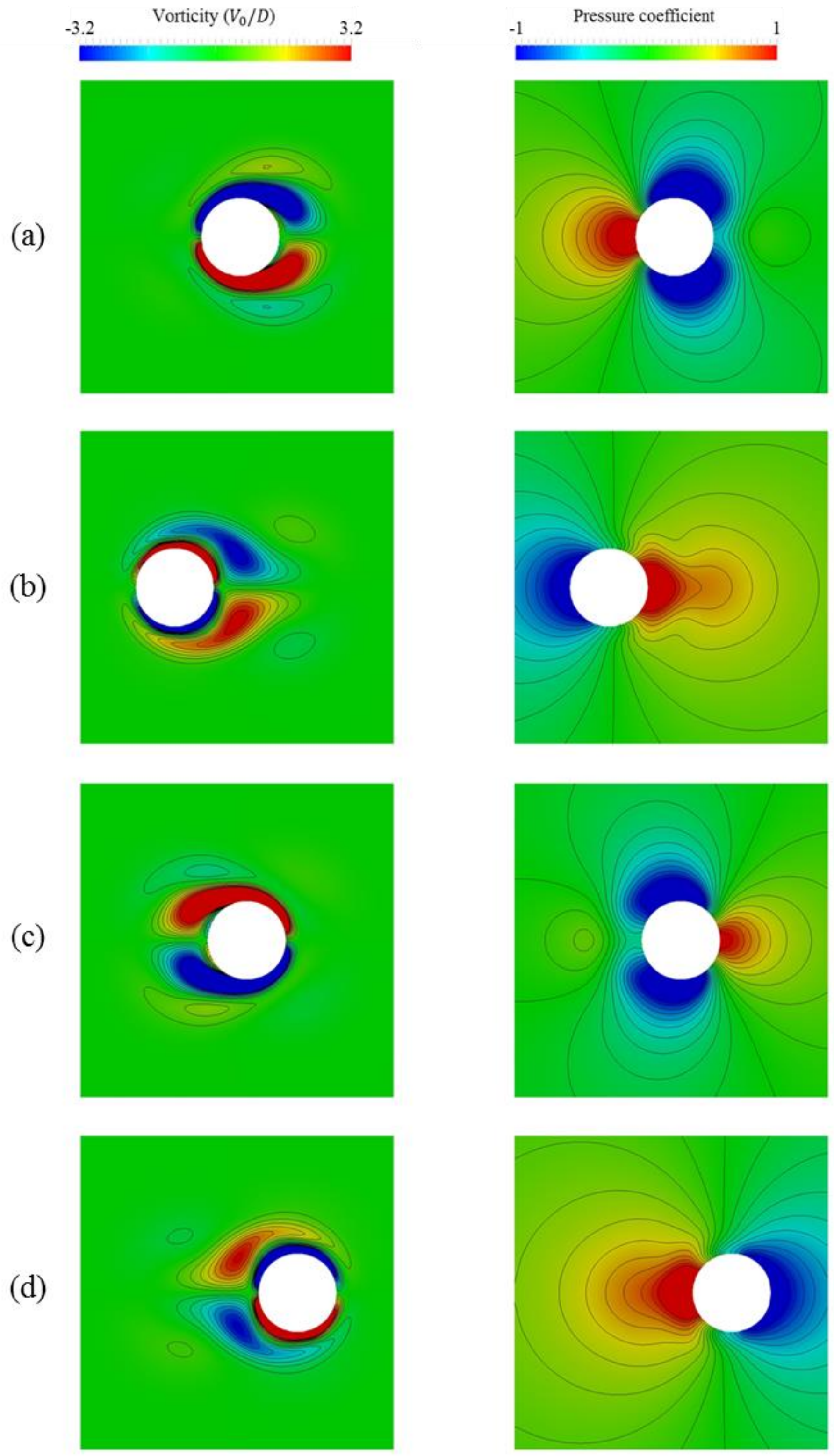


Fig. 6.16. Snapshots of vorticity contours (left column) and pressure coefficient contours (right column) at different phases $(2\pi f t)$ of the oscillation: (a) $0°$; (b) $96°$; (c) $192°$; (d) $288°$. The size of the captured domain is $[-2D, 2D] \times [-2D, 2D]$. The vorticity contours are presented from $-3.2V_0/D$ to $3.2V_0/D$ with 16 intervals. The contours of the pressure coefficient are presented from -1.0 to 1.0 with 20 intervals.

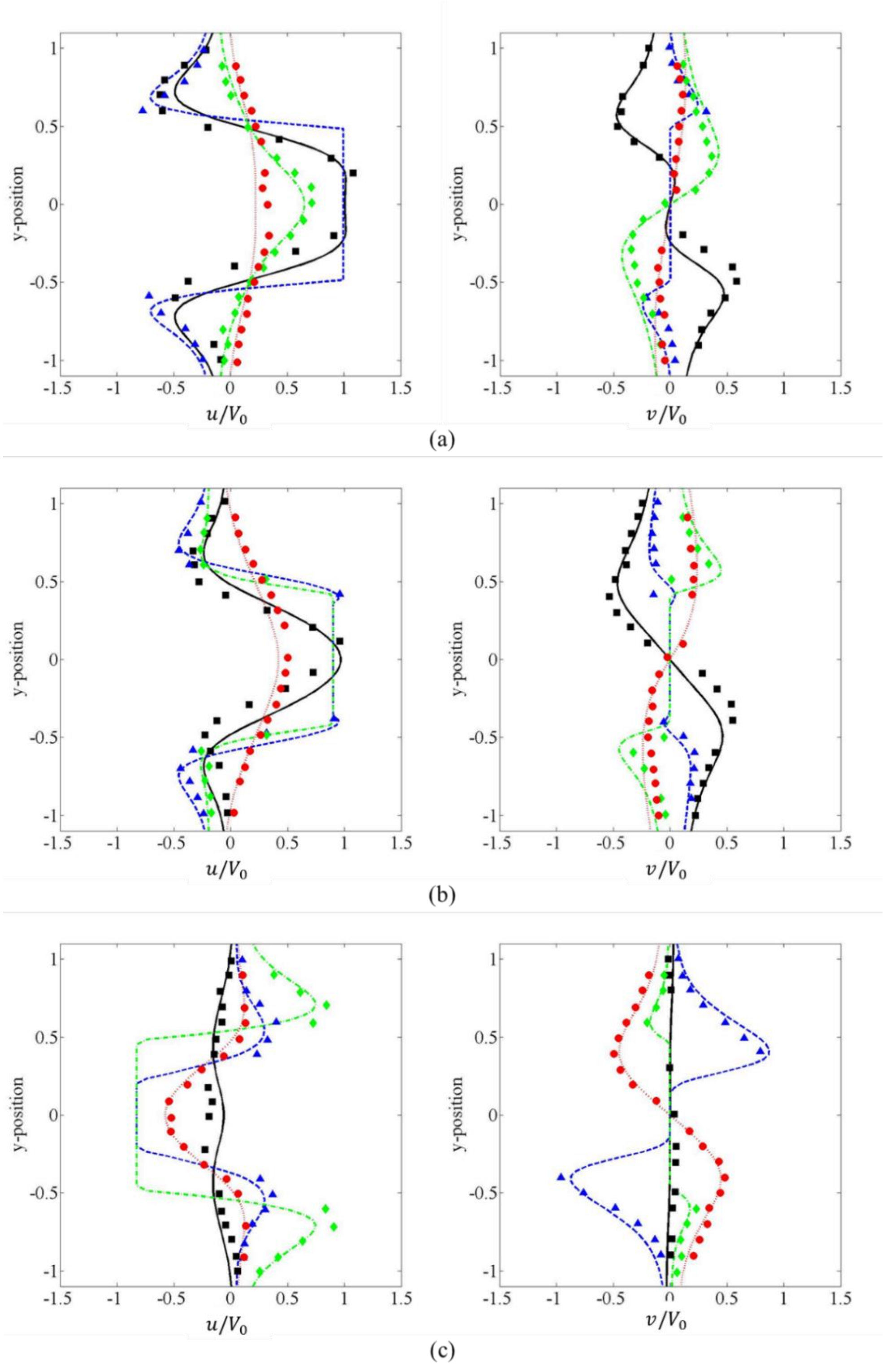


Fig. 6.17. Velocity profiles of the x-direction velocity u (right column) and the y-directional velocity v (left column) along the y-position at four different vertical sections ($x = -0.6D, 0, 0.6D, 1.2D$) for three different phases ($2\pi f t$): (a) 180°; (b) 210°; (c) 330°. The experimental results [24] are indicated by ■ at $x = -0.6D$, ▲ at $x = 0$, ◆ at $x = 0.6D$, and ● at $x = 1.2D$. The current numerical results based on the transformed SDF are presented by — at $x = -0.6D$, $- -$ at $x = 0$, $- \cdot -$ at $x = 0.6D$, and $\cdots$ at $x = 1.2D$.

6.5. Flow around a flapping wing

The example for the flow induced by a flapping wing is introduced by Wang [100], with the aim of investigating the 2D mechanism for an insect hovering employing a simple ellipse as a wing. This problem can be considered to be a very appropriate test case for the moving body simulation because a moving object has not only translational but also rotational motions. For this reason, this example is a very famous test case for moving body simulations, like the previous oscillating cylinder problem.

The configuration of the flapping wing is presented in Fig. 6.18. The wing is a simple 2D ellipse defined by c and b which are lengths along major and minor axes, respectively. The aspect ratio of the wing is defined as $AR = c/b = 4$. The wing translationally oscillates along a stroke plane inclined at an angle β with a prescribed rotational motion. The prescribed translational and rotational motions of the wing are given by

$$A(t) = \frac{A_0}{2}\left\{cos\left(\frac{2\pi t}{T}\right) + 1\right\}, \tag{6.11}$$

$$\theta(t) = -\theta_0\left\{1 - sin\left(\frac{2\pi t}{T}\right)\right\}, \tag{6.12}$$

where $A(t)$ and $\theta(t)$ refer to translational and rotational motions, respectively. A_0 provides the total length of the flapping path, θ_0 is the initial angle of attack, which is positive for the counterclockwise direction, an T is the flapping period. Since the stroke plane is inclined at β, the motions about the global (inertial) coordinate can be defined as

$$x(t) = \frac{A_0}{2}\left\{cos\left(\frac{2\pi t}{T}\right) + 1\right\}cos(\beta), \tag{6.13}$$

$$y(t) = \frac{A_0}{2}\left\{cos\left(\frac{2\pi t}{T}\right) + 1\right\}sin(\beta). \tag{6.14}$$

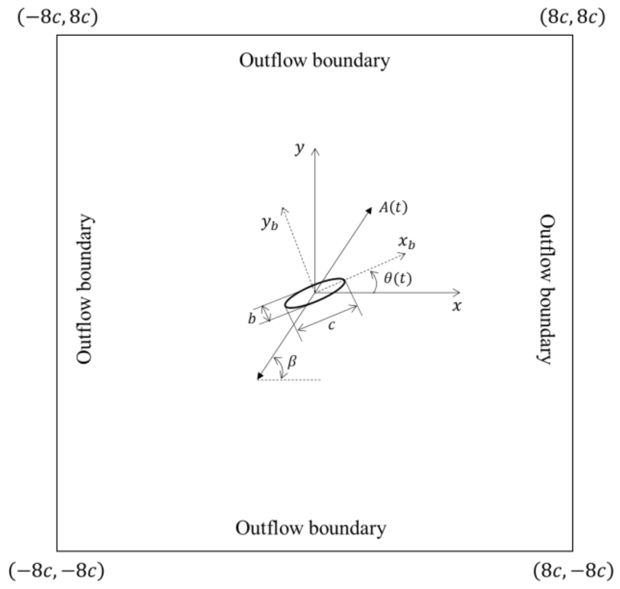


Fig. 6.18. The layout of the entire computational domain and the configuration for the flapping wing. Here, (x, y) means the global (inertial) coordinate and (x_b, x_b) refers to the body-fitted coordinate.

The parameters utilized to define the hovering motion are chosen to be the same as those used in previous studies [100, 22, 62, 63, 107]: $A_0 = 2.5c$, $\theta_0 = \pi/4$, $T = \pi A_0/c$, and $\beta = \pi/3$. The maximum speed V_0 is $\pi A_0/T$. The Reynolds number is defined as $Re = V_0 c/\nu$ and set to 157.

As shown in Fig. 6.18, the size of the computational domain is chosen to be $[-8c, 8c] \times [-8c, 8c]$, and the uniform 512×512 Cartesian grids are employed to cover the entire computational domain. Based on this condition, 32 uniform grids are located along the major axis of the wing (ellipse). The boundary condition is equally set to that of the previous oscillating cylinder problem. The center of the wing is initially located at a point $(A_0 cos\beta, A_0 sin\beta)$ and oscillates towards a point (0,0). The CFL number based on the maximum speed V_0 is set to 1.28.

Here, the simulations for the flow around the flapping wing are conducted by using two different methods in terms of representing a moving body. One is based on the exact SDF, and the other is based on the transformation of the numerically computed SDF (see Chapter 4.7) using the AMR-based algorithm (see Chapter 4.1). In the latter case, the boundary of the ellipse is represented by a set of 256 line-segments.

Fig. 6.19 shows that the results of these two methods overlap exactly and are in good agreement with the previous numerical results [100] obtained by the body-fitted mesh-based simulation. For both the drag and lift coefficients, there are discrepancies at the peak points. We believe that this error comes from the difference between mesh systems because current trends are similarly observable at numerical results based on the immersed/embedded boundary methods [22, 62, 63, 107].

The instantaneous vorticity fields at four different times ($t = 0.25T, 0.44T, 0.74T, 0.99T$) are presented in Fig. 6.20. These results are obtained by the numerical simulation using the current SDF transformation algorithm. As shown in this figure, the vortex dipole, which is produced by a pair of positive and negative vortices near the wing, is generated moving in the downward direction. This phenomenon results in generating the lift of the hovering wing. The current results are very similar to the results of the previous numerical simulations [100, 22, 62, 63, 107]. Based on this result, we re-confirm that the current SDF transformation algorithm for the moving body simulation, which can avoid the re-computation of the SDF involving the in/out test and the computation for the shortest distance, and can also be safely applied to engineering problems containing a moving object.

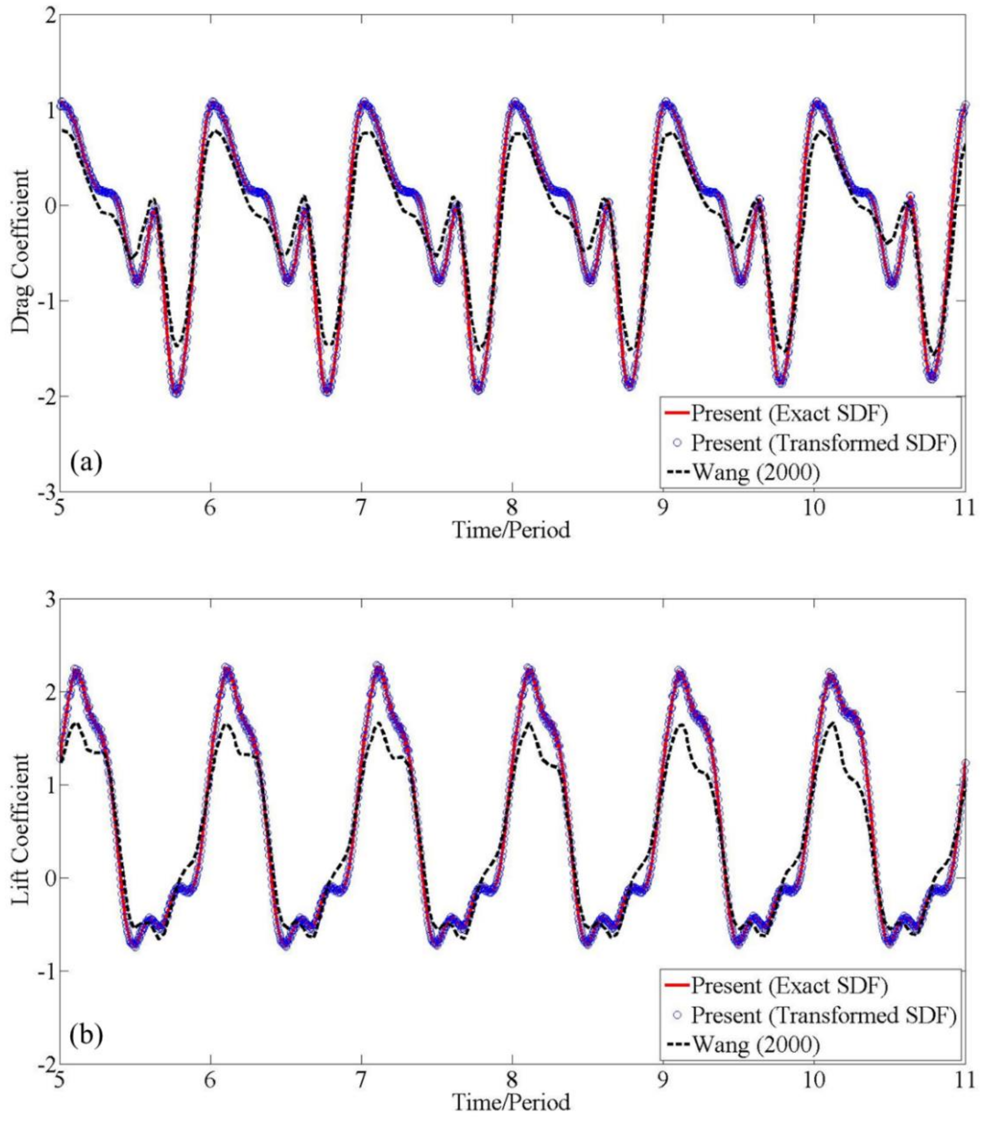


Fig. 6.19. Time history of the (a) drag and (b) lift coefficients for the flow around a flapping wing at $Re = 157$. The results of Wang [100] are obtained by the numerical simulation based on the body-fitted grid system.

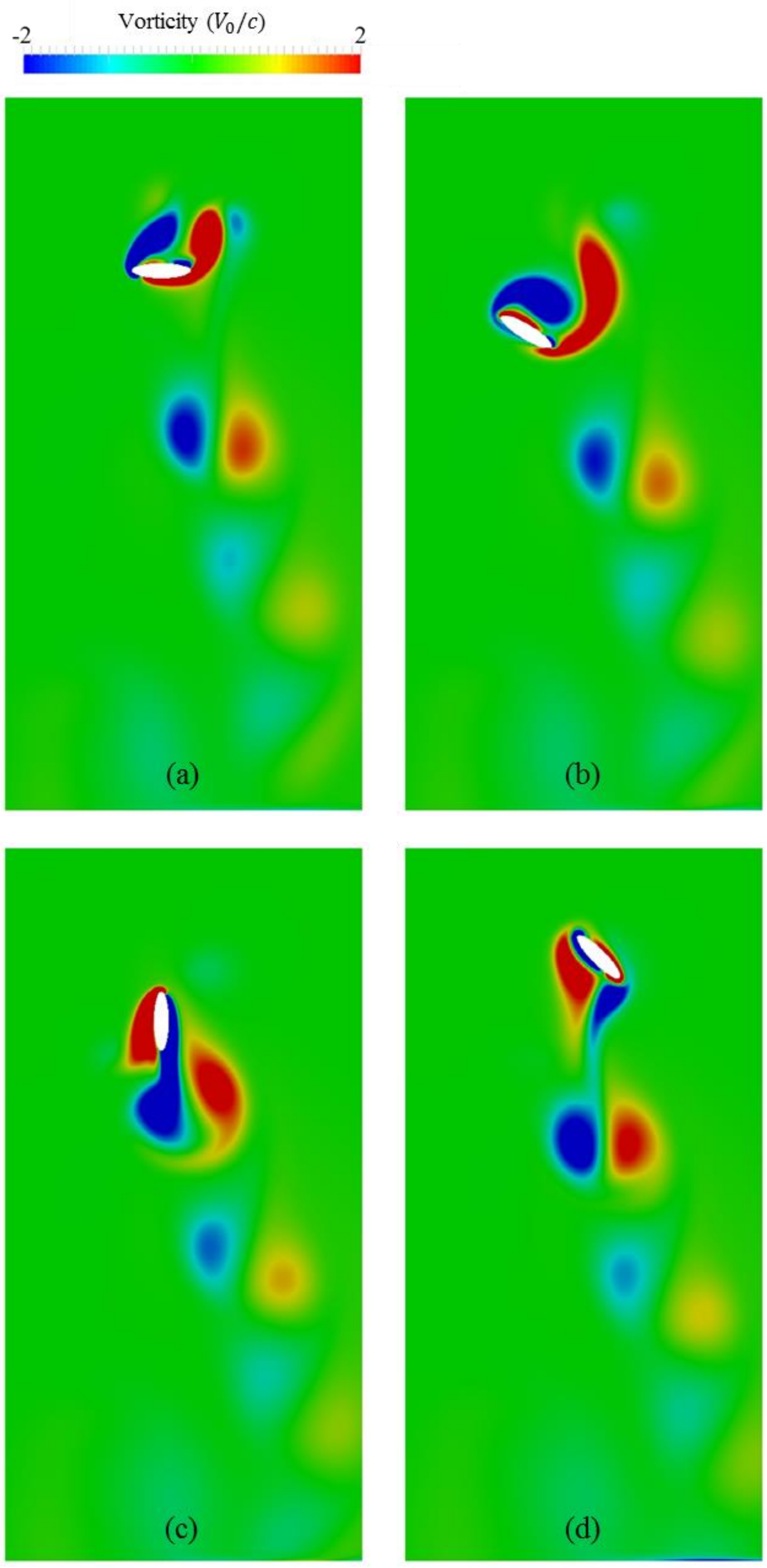


Fig. 6.20. Snapshot of the vorticity fields induced by the flapping wing during one flapping period at four different time moments: (a) $t = 0.25T$; (b) $t = 0.44T$; (c) $t = 0.74T$; (d) $t = 0.99T$. The size of the zoomed-in domain is $[-2c, 4c] \times [-8c, 4c]$.

6.6. Flow past a heaving and pitching NACA0012 airfoil

In this sub-chapter, the simulation results for the flow past a heaving and pitching NACA0012 airfoil are presented. This example is introduced as one of the advanced test cases in the fifth international workshops on high-order CFD methods (HiOCFD5) [79]. This problem is originally aimed at investigating the accuracy and performance of high-order compressible flow solvers. In this example, since the airfoil, which is the most famous object for the engineering simulation, has both translation and rotational motions, we adopt it as the advanced test case of the moving body simulation in the current study. Although the simulation results of [79] are obtained by compressible flow solvers, the comparison with the current incompressible flow solver can be possible due to the Mach number of 0.2, where the compressible effect is very small.

This example consists of three different cases: (1) pure heaving; (2) smooth heaving and pitching; (3) violent heaving and pitching. For each respective case, the heaving (h) and pitching (θ) motions are defined as

$$h_1(t) = \frac{t^2}{4}(3-t) \quad \text{and} \quad \theta_1(t) = 0, \tag{6.15}$$

$$h_2(t) = \frac{t^2}{4}(3-t) \quad \text{and} \quad \theta_2(t) = -\frac{\pi}{3}t^2(t^2-4t+4), \tag{6.16}$$

$$h_3(t) = \frac{t^3}{16}(-8t^3+51t^2-111t+84) \quad \text{and} \quad \theta_3(t) = -\frac{4\pi}{9}t^2(t^2-4t+4), \tag{6.17}$$

where the subscript refers to the case numbers. The motions are defined about a point located at the airfoil $\mathrm{c}/3$ location from the leading edge, where c is the chord length of the airfoil. The positive heaving motion is an upward movement, and the positive pitching motion is counterclockwise rotation.

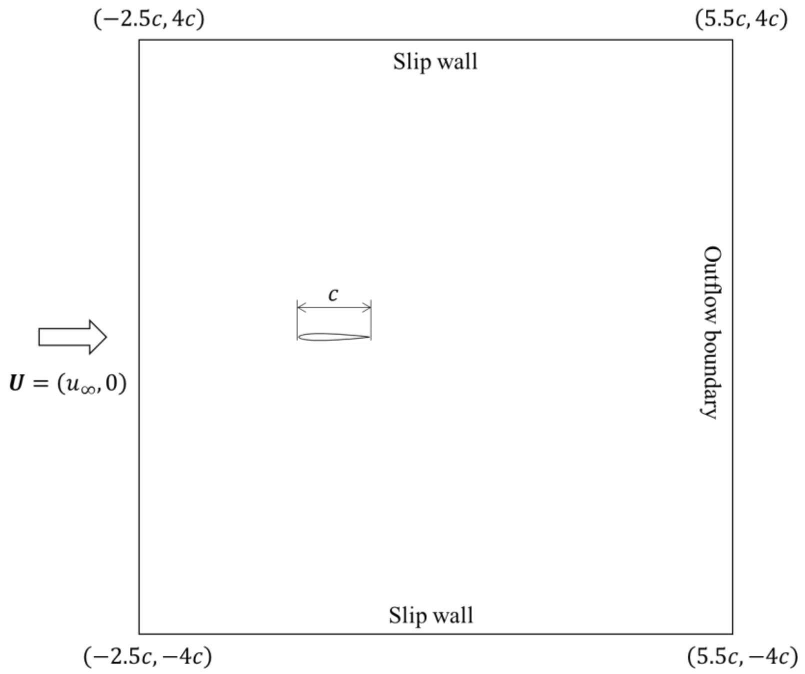


Fig. 6.21. The layout of the computational domain for the flow past the heaving and pitching NACA0012 airfoil

The entire computational domain for the flow past the heaving and pitch NACA0012 airfoil is shown in Fig. 6.21. The computational domain is chosen to be $[-2.5c, 5.5c] \times [-4c, 4c]$, and a point located at the $c/3$ location from the leading edge is positioned at $(0,0)$. We impose the Dirichlet boundary condition of $\boldsymbol{U} = (u_\infty, 0)$ at the left, the slip boundary condition at top and bottom, the outflow condition, and the no-slip wall boundary condition at the airfoil`s boundary. For the pressure, the Dirichlet boundary condition is imposed on the outlet as $p_\infty = 0$, and other boundaries are considered to follow the Neumann condition, i.e., $dp/d\boldsymbol{n} = 0$. A 1024×1024 uniform Cartesian mesh system is used, thus the rectangular region covering the foil consists of 128×12 uniform Cartesian cells. The Reynolds number, based on the free-stream velocity u_∞ and the chord length c, is set to 1000, and the CFL number, based on the free-stream velocity, is 4. Here, the geometry of the NACA0012 airfoil is defined by a set of 69 line-segments, and the SDF is numerically computed by the AMR-based algorithm.

After converging to the steady-state, the prescribed heaving and pitching motions, defined as Eqs. (6.15)-(6.17), are applied to the NACA0012 airfoil during a time interval of $2c/u_\infty$. The converged drag coefficient of UC Berkeley`s simulation [79] is 0.12, and that of the current analysis is 0.11. The comparison is considered to be very satisfactory. For this problem, the drag coefficient is defined as $C_d = F_d/(0.5\rho u_\infty^2 c)$, where F_d is the drag force.

The instantaneous vorticity fields at four different time moments ($t = 0.5c/u_\infty, 1.0c/u_\infty, 1.5c/u_\infty, 2.0c/u_\infty$) are shown in Fig. 6.22 with respect to each case. The behaviors of the vortex for each case are in good agreement with physical plausibility. Fig. 6.23 shows the time history of the drag and lift coefficients, where the line components indicate the UC Berkeley`s results based on the body-fitted mesh system among [79] and the symbols refer to the current results. As shown in this figure, the current results for both the drag and lift coefficient are well matched to the reference data in cases 1 and 2, whereas there is a remarkable discrepancy at the second peak of the drag in case 3. This discrepancy can be induced by the differences between the two mesh systems. Since the heaving and pitching motions of case 3 are very rapid, it is hard to predict this situation using the uniform Cartesian mesh system. This kind of discrepancy at peak points is also observable in the previous flapping wing problem.

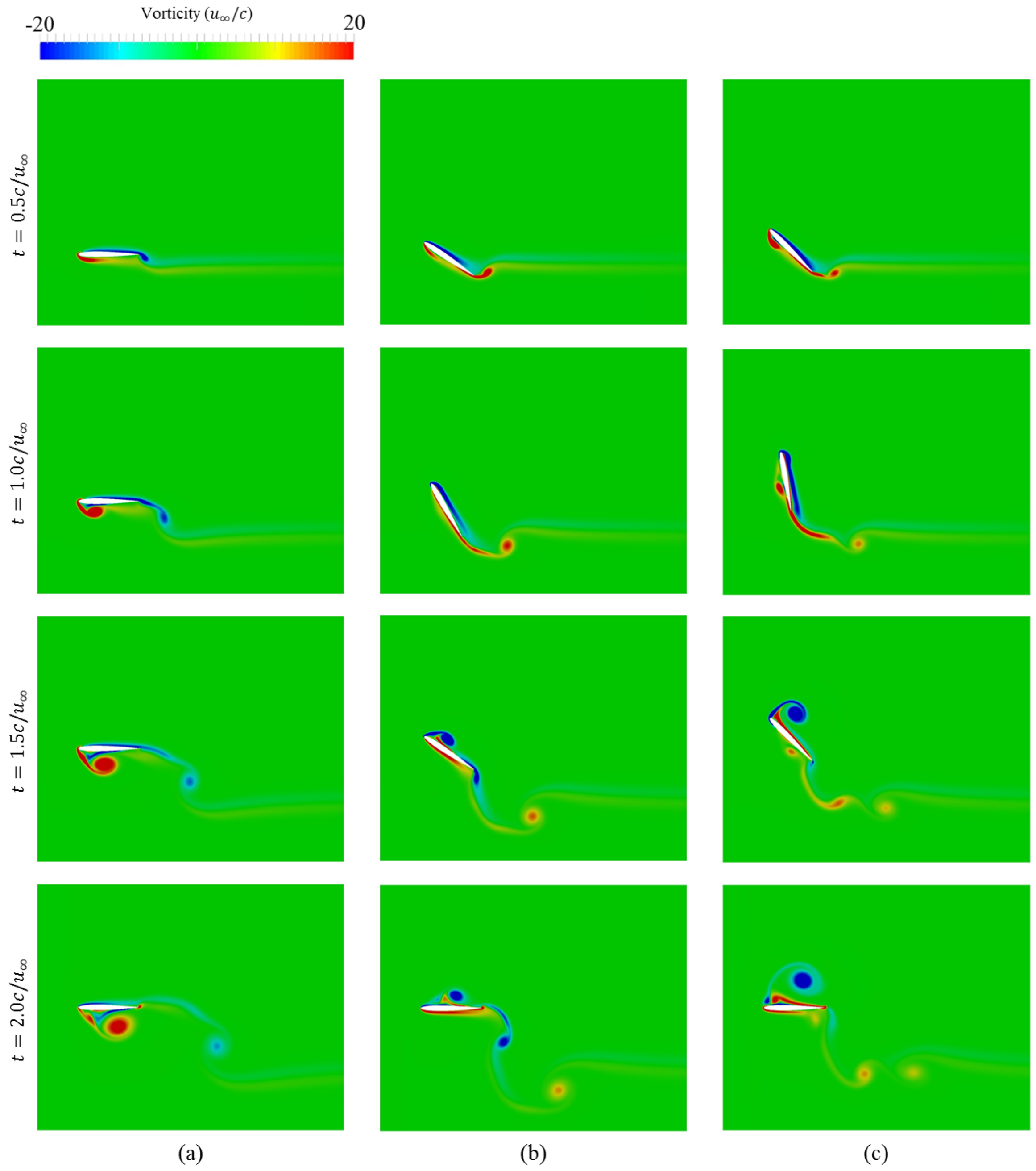


Fig. 6.22. Snapshots of the instantaneous vorticity field at four different time moments: (a) case 1; (b) case 2; (c) case 3.

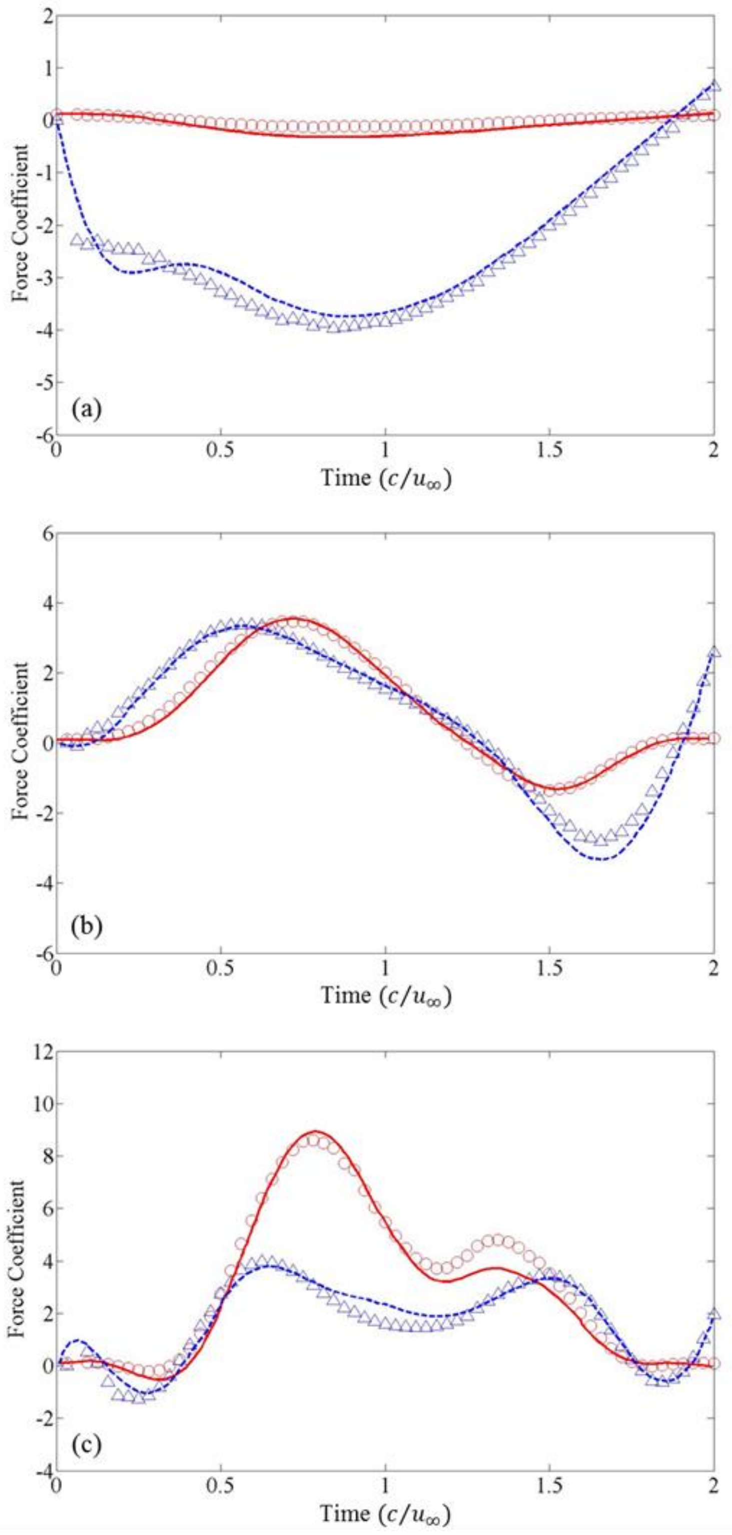


Fig. 6.23. Time history of the drag and lift coefficient with different motions: (a) case 1 for pure heaving; (b) case 2 for smooth heaving and pitching; (c) case 3 for violent heaving and pitching. The numerical results, based on the UC Berkeley simulation using the body-fitted mesh system [79], are indicated by **—** and – – for the drag and lift coefficient, respectively. The current numerical results based on the transformed SDF are presented by ○ and △ for the drag and lift coefficient, respectively

Chapter 7

3D Simulation Results and Discussion

7.1. Flow past a sphere

7.1.1. Laminar flow

A well-known three-dimensional benchmark problem, namely flow past a sphere, is presented in this sub-chapter. As shown in Fig. 7.1, the computational domain is $[-4D, 12D] \times [-4D, 4D]^2$ where D refers to the diameter of the sphere, and the center of the sphere is located at $(0,0,0)$. The Dirichlet boundary condition is imposed at the inlet (left) boundary with the uniform flow condition $\boldsymbol{U} = (u_\infty, 0,0)$, and the outlet (right) boundary condition is set to the convective boundary conditions. The sphere surface is considered as the no-slip wall, and the slip boundary condition is applied to other boundaries (side walls). In the case of pressure, the Dirichlet boundary condition is applied at the outlet as $p_\infty = 0$, and the Neumann boundary condition, i.e. $dp/d\boldsymbol{n} = 0$, is imposed for the rest of the boundaries.

Table 7.1
Drag coefficients of flow past a sphere compared with previous results

	$Re = 100$	$Re = 250$	$Re = 300$
Kallinderis and Ahn [53]	1.084	-	-
Kim et al. [52]	1.087	0.701	0.657
Choi et al. [21]	1.09	0.700	0.658
Johnson & Patel [47]	-	0.700	0.656
Present	1.07	0.710	0.672

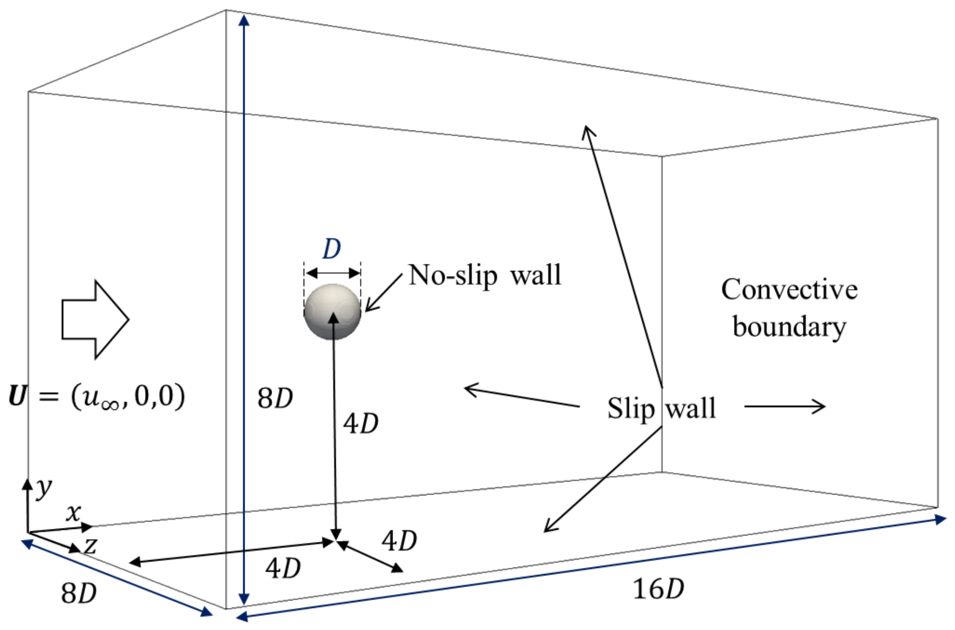


Fig. 7.1. Layout of the computational domain for laminar flow past a sphere problem.

A $512 \times 256 \times 256$ uniform Cartesian mesh system is utilized in the entire computational domain. The subdomain that covers the sphere is discretized by $32 \times 32 \times 32$ uniform size cells. For all simulations presented in this sub-chapter, the Reynolds number (Re) is based on the sphere diameter D and the free-stream velocity u_∞, and the CFL number based on the free-stream velocity u_∞ is set to 4. The simulation is conducted at three different Reynolds numbers: $Re = 100, 250,$ and 300. This problem was computed on KISTI`s *Nurion* supercomputer. The computational domain is divided into $8 \times 4 \times 4$ uniform Cartesian blocks for MPI parallelization; thus, a total of 128 MPI processes are assigned and 16 OpenMP threads are utilized at each MPI process. Therefore, a total of 2,048 cores are used for the parallel computation.

For the sphere, C_D is defined as $F_D/(0.5\rho u_\infty^2 A)$, where A is the projected area of the sphere, i.e., $A = \pi D^2/4$, and should not be confused with the surface area. Here, F_D can be computed in the similar manner to the case of the cylinder problem. As shown in Table 7.1, current results for the drag coefficient are in good agreement with previous results. Fig. 7.2 depicts the variation of the wall pressure coefficient and the wall azimuthal vorticity along the sphere surface at $Re = 100$ and shows that the comparison between the current and previous results [46, 52] is very satisfactory.

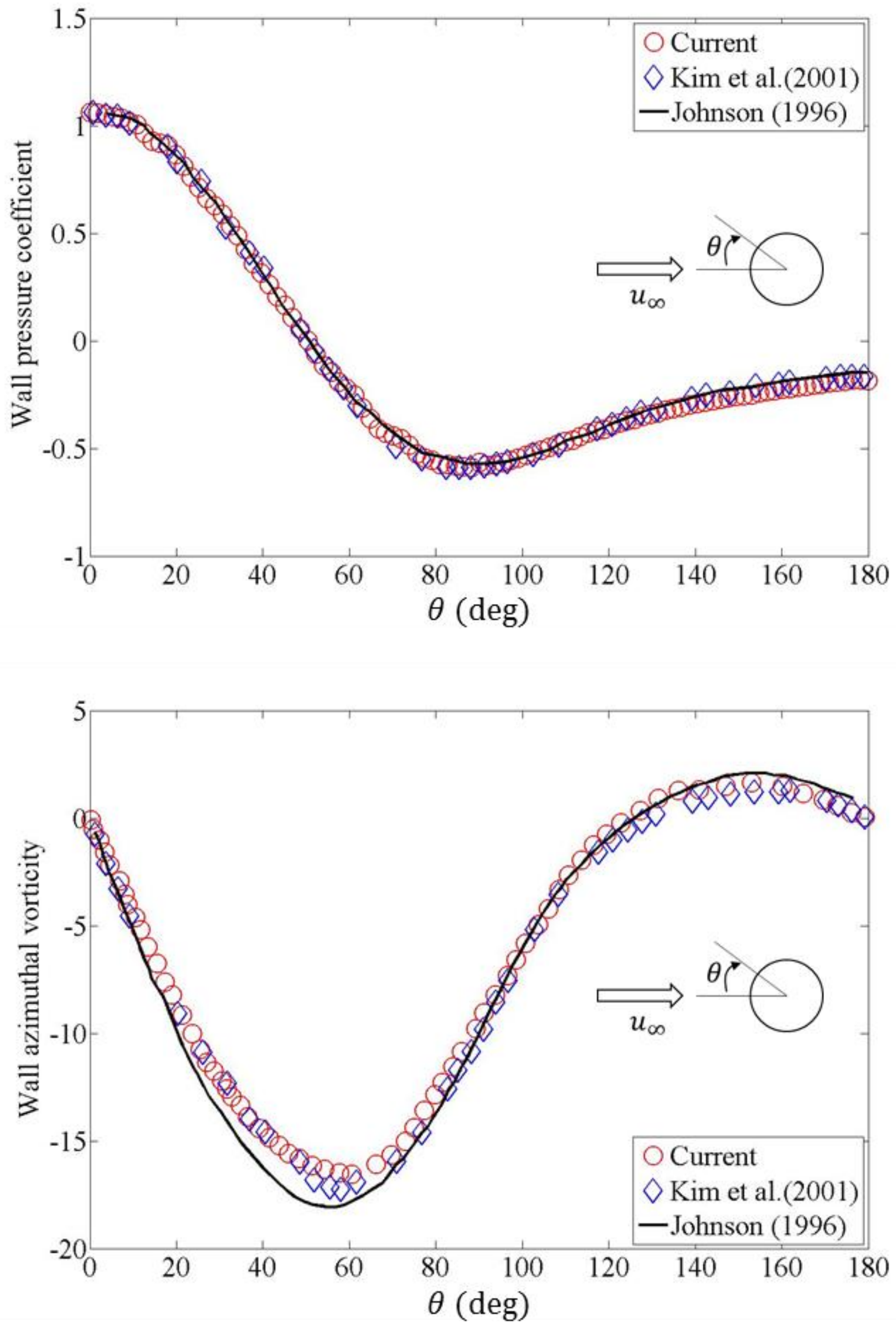


Fig. 7.2. (a) Wall pressure coefficient and (b) wall azimuthal vorticity along the sphere surface at $Re = 100$

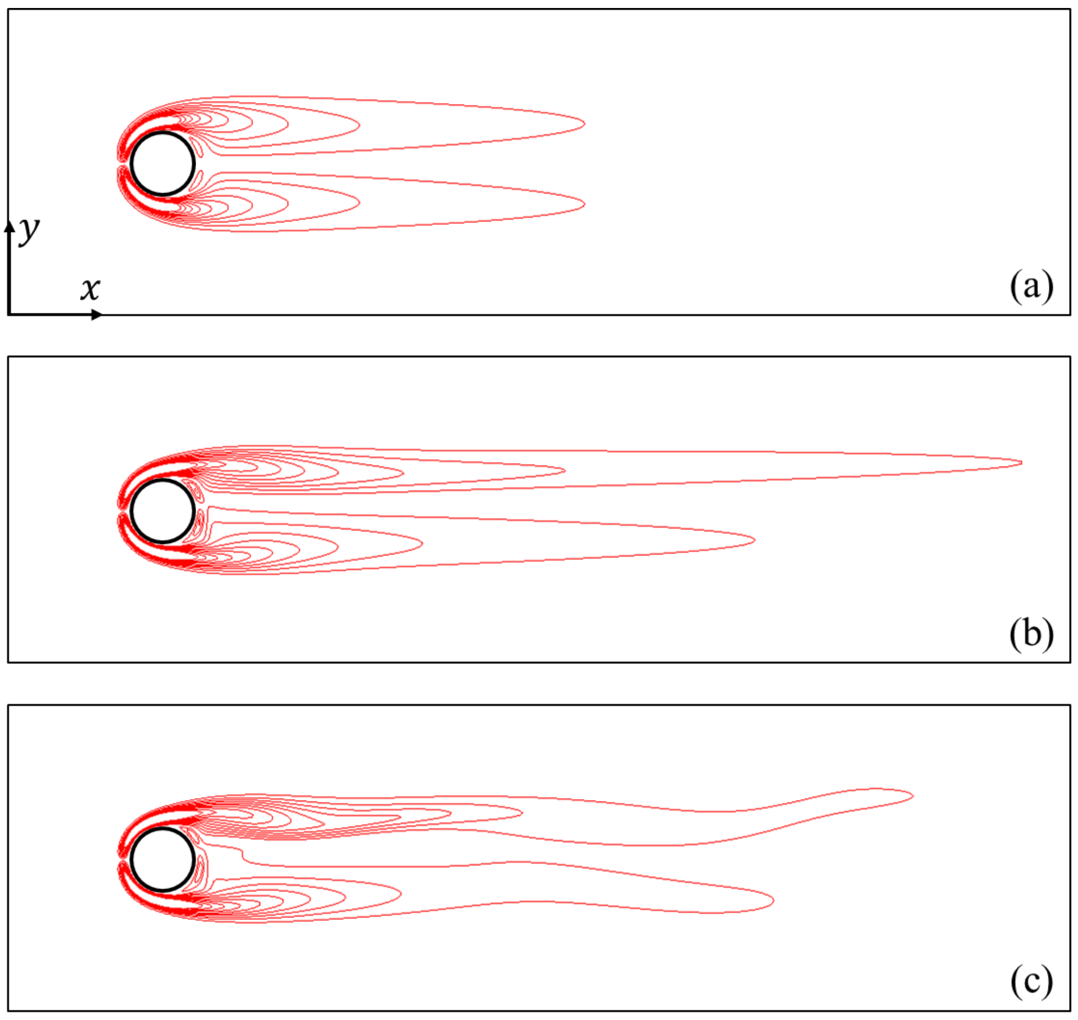


Fig. 7.3. Contours of z-directional vorticity in xy-plane: (a) $Re = 100$; (b) $Re = 250$; (c) $Re = 300$. The size of the presented domain is $[-2D, 12D] \times [-2D \times 2D]$. Contours are presented from -$5u_\infty/D$ to $5u_\infty/D$ with 21 intervals.

It is well known that laminar flow can be divided into three different flow regimes [52]: steady axisymmetric flow $(Re \leq 200)$, steady non-axisymmetric flow $(210 \leq Re \leq 270)$, and unsteady flow $(Re \geq 280)$. To show that the current simulation can represent the variation of these flow characteristics well, the contour of the z-directional vorticity (ω_z) is presented in Fig. 7.3. Since unsteady flow occurs at $Re = 300$, the instantaneous contour of the vorticity is captured at the half of the shedding period. Fig. 7.3 shows that flow characteristics corresponding to each Re are well represented. In addition, these contours are in good agreement with the results presented in Johnson and Patel [47].

The vortical structure based on the λ_2-definition [44] is presented in Fig. 7.4. At $Re = 100$, because the flow around the sphere is axisymmetric, the vortical structure is attached only near the sphere. In contrast, two slender vortical structures are produced behind the sphere at $Re = 250$ due to the non-symmetricity. At $Re = 300$, the elongated vortex is detached from the sphere, and eventually the shedding vortical structure is observed. The variation of vortical structures with respect to time at $Re = 300$ is visually presented at Fig. 7.5. These variations of the vortical structure with respect to Re are nearly identical to the results shown in Kim et al. [52] and Johnson and Patel [47].

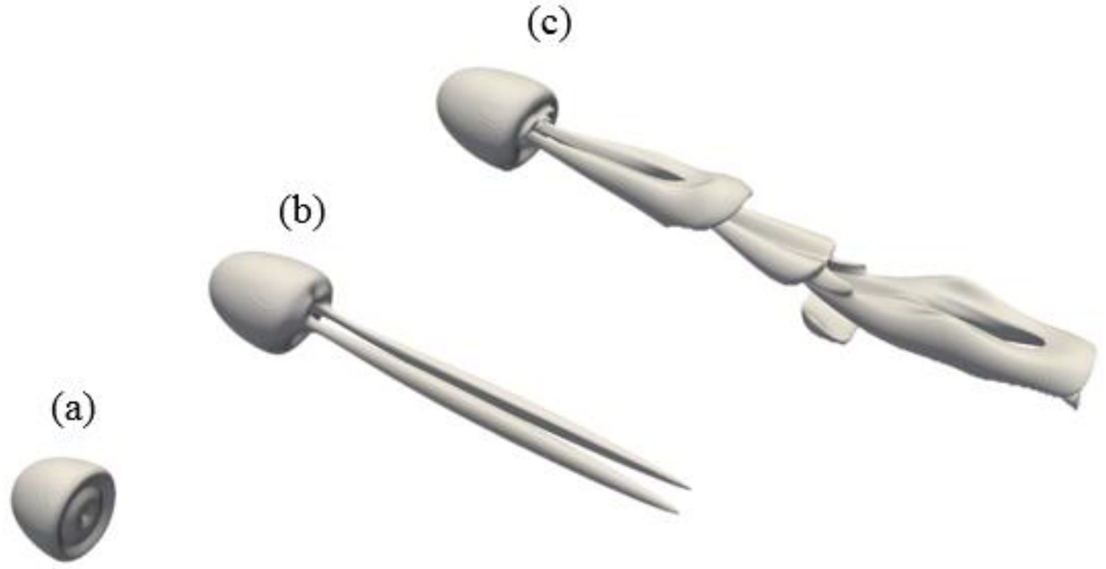


Fig. 7.4. Perspective view of the vortical structure based on λ_2 definition: (a) $Re = 100$; (b) $Re = 250$; (c) $Re = 300$. For $Re = 300$, the instantaneous result at half of shedding period is shown due to unsteadiness.

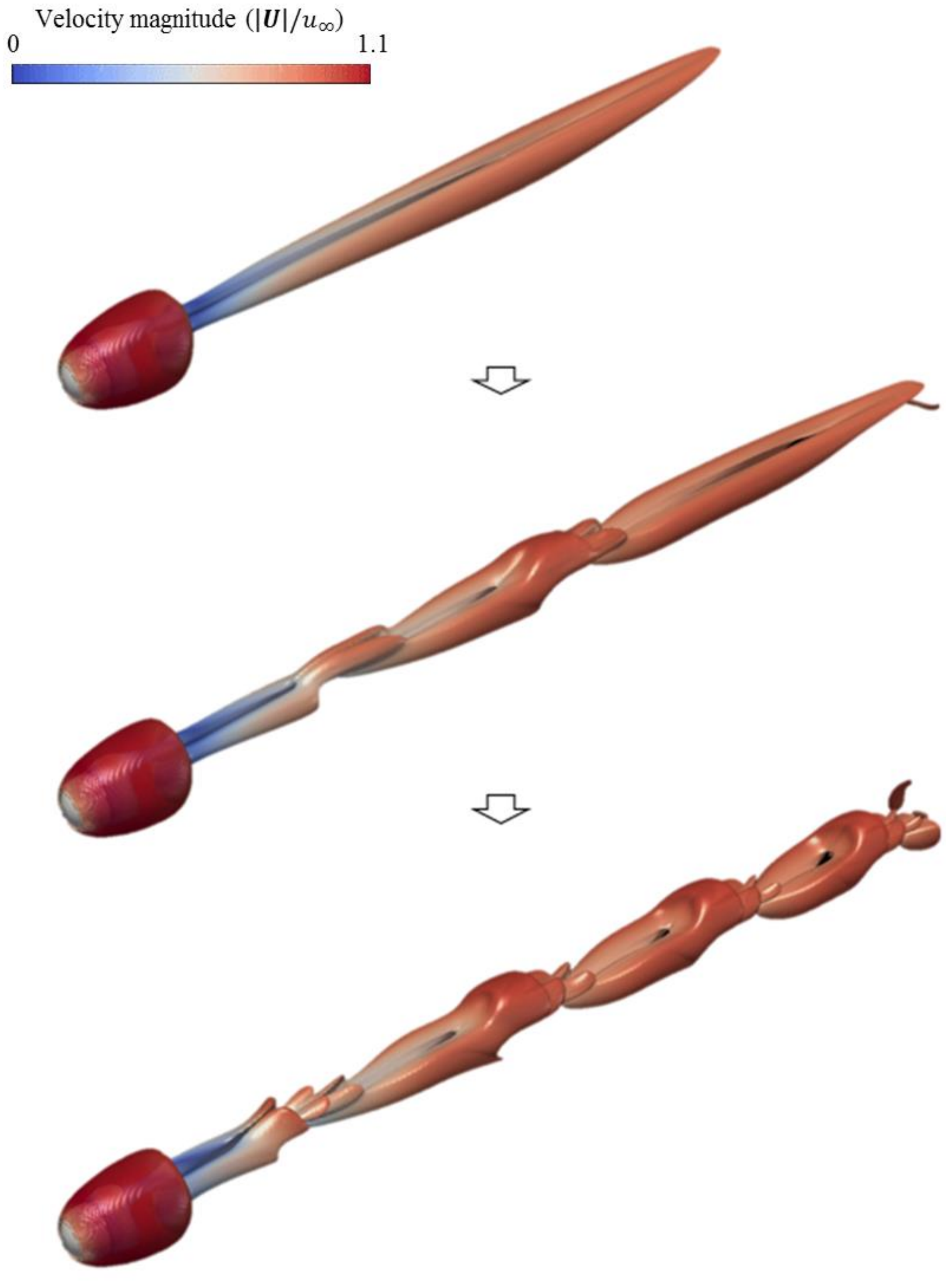
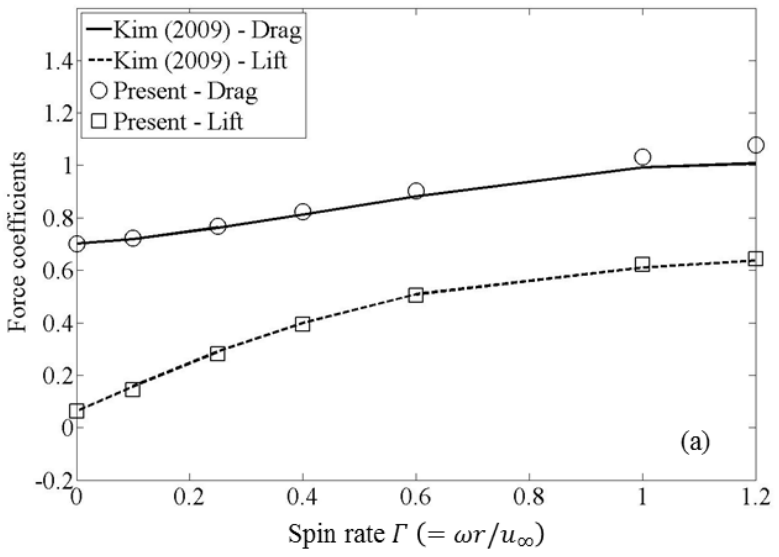


Fig. 7.5. Visualization of vortical structures with respect to time evolution at $Re = 300$. Vortical structures are colored by velocity magnitude.

Finally, we also consider the flow past a rotating sphere. In this problem, all simulation condition is same with the previous case for the stationary sphere, except for the spin rate of the sphere. The spin rate (Γ) of the sphere is defined as $\Gamma = \omega R / u_\infty$ where ω is the angular velocity, R is the radius of the sphere, and u_∞ means the free-stream speed. The Reynolds number is set to 250 and 300, and the spin rate is varied from 0.0 to 1.2 with 6 intervals. Fig. 7.6 shows the variation of force coefficients, i.e. drag and lift coefficients, with respect to the spin rate at different Reynolds numbers. The current simulation results are in good agreement with previous numerical results [55] based on the direct forcing immersed boundary method using the cylindrical coordinate system.



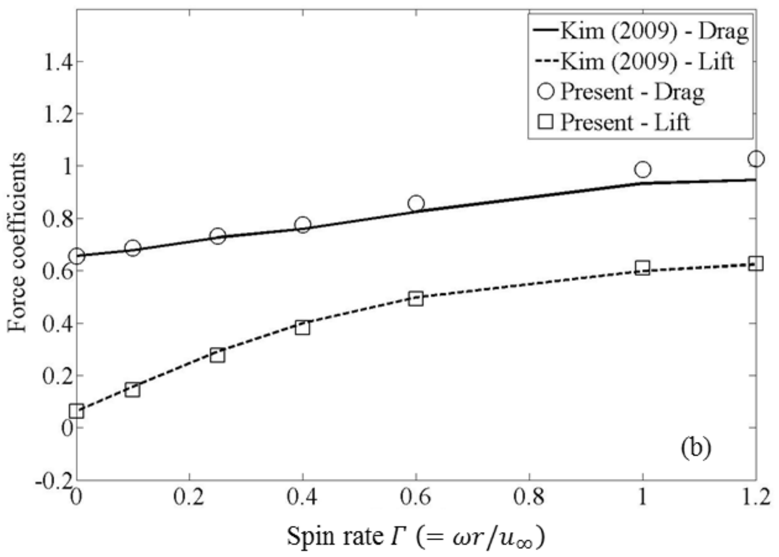


Fig. 7.6. Variation of drag and lift coefficients with respect to the spin rate for the spinning sphere: (a) $Re = 250$; (b) $Re = 300$. Kim (2009) [55] is based on the direct forcing immersed boundary method using the cylindrical coordinate system.

7.1.2. Subcritical flow

Because the flow transition occurs in the wake region of the sphere at $Re = 3700$, the larger computation domain and the greater number of grids are required than previous laminar flow cases. For this reason, in order to analyze the subcritical flow around a sphere where $Re = 3700$, the computational domain (see Fig. 7.7) is set to $[-4D, 28D] \times [-8D, 8D]^2$ where D means the diameter of the sphere, which is positioned at the origin, and the number of grids is set to $2{,}048 \times 1{,}024 \times 1{,}024$ $(= 2.1 \times 10^9)$ where the 64 grids are positioned along the diameter of the sphere. As shown in Fig. 7.7, boundary conditions are equally imposed with previous laminar flow cases. The CFL number based on the free-stream velocity is set to 2.0.

The computational domain is divided into $16 \times 8 \times 8$ uniform blocks for conducting hybrid MPI/OpenMP parallel computation, thus $128 \times 128 \times 128$ grids are assigned to each MPI process. For each MPI process, 8 threads are assigned to apply OpenMP parallel computation. Therefore, total 8,192 cores (1,024 MPI process $\times$ 8 OpenMP threads) are employed to compute this problem. The computation is carried out on the *Nurion* supercomputer by using 128 nodes. For the single node, 8 MPI processes are allocated.

In the current research, a subgrid-scale (SGS) is treated by numerical dissipation induced by numerical discretization scheme, and this approach is known as implicit large eddy simulation (LES) [11, 32, 96, 93]. This approach is especially convenient in flow regimes where the derivation or the computation of SGS is difficult. If an appropriate discretization scheme for implicit LES is applied, any additional procedure would not be required according to variation of the Reynolds number. For this reason, the potential of implicit LES approach has been gaining popularity in recent studies.

Table 7.2
Flow parameters of subcritical flow past a sphere

	$\overline{C_D}$	$\overline{C_{pb}}$	St	$\overline{L_{rec}}/D$
Kim & Durbin [50] (Exp.)	-	-0.224	0.225	-
Sakamoto & Haniu [87] (Exp.)	-	-	0.204	-
Schlichting [86] (Exp.)	0.39	-	-	-
Yun et al. [105] (LES)	0.355	-0.194	0.21	2.622
Rodriguez et al. [83] (DNS)	0.394	-0.207	0.215	2.28
Present	0.41	204	0.20	2.61

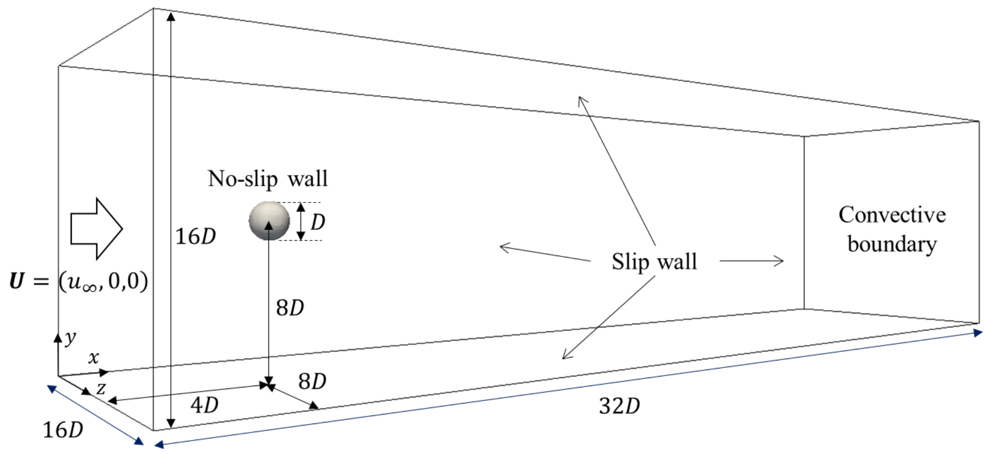


Fig. 7.7. Layout of the computational domain for subcritical flow past a sphere problem.

Flow parameters of the current simulation are summarized in Table 7.2, where the time-averaged drag coefficients ($\overline{C_D}$), base pressure coefficient ($\overline{C_{pb}}$), Strouhal number based on the vortex shedding frequency, and non-dimensional recirculation length ($\overline{L_{rec}}/D$) are presented together with previous experimental and numerical (DNS, LES) results [50, 83, 86, 87, 105]. All results are in good agreement with previous data. Fig. 7.8 shows the comparison results of the mean pressure coefficient on the sphere surface along the azimuthal angle (θ). Although there is minor discrepancy at minimum value, overall tendency is very similar with previous results. The base pressure coefficient (C_{pb}) presented in Table 7.2 is the value at $\theta = 180°$.

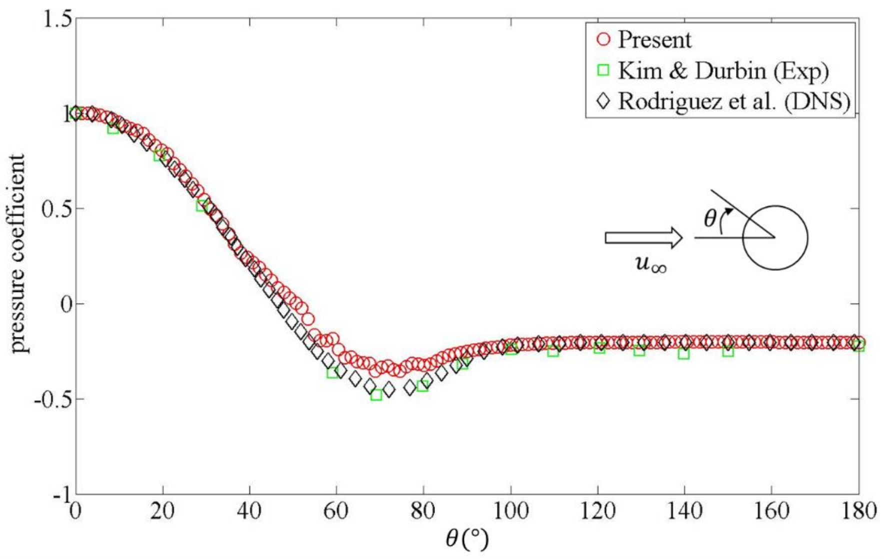


Fig. 7.8. Comparison of the mean pressure coefficient on the sphere surface: ○, present results; □, experimental results by Kim and Durbin [50]; △, DNS results by Rodriguez et al. [83];

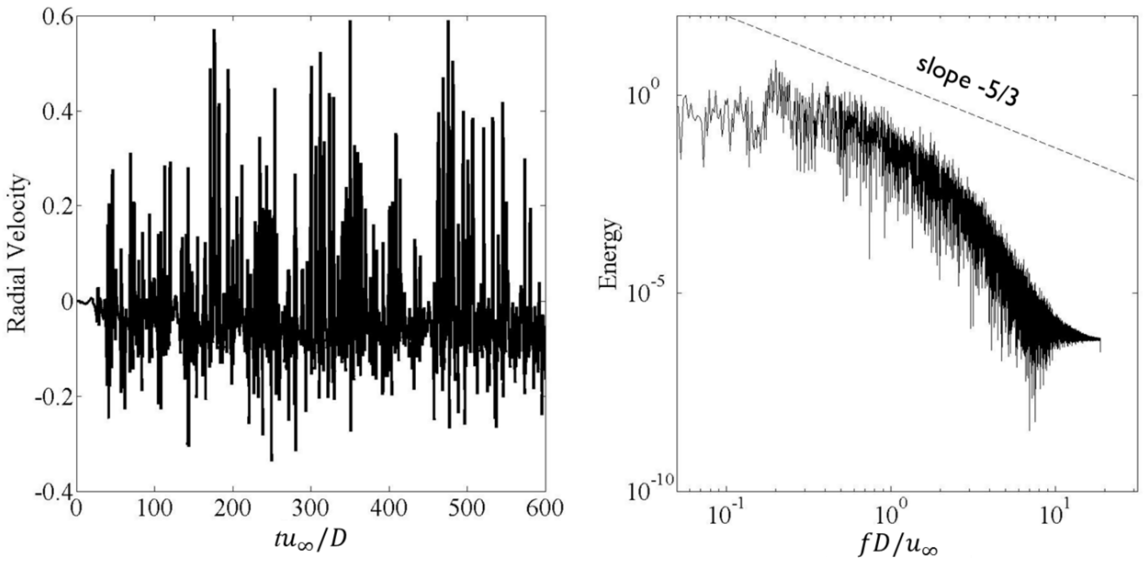


Fig. 7.9. (a) Time history and (b) Energy spectrum of the radial velocity at $x/D = 5.0$ and $r/D = 0.6$.

Fig. 7.9 shows the time history and energy spectrum of the radial velocity at $x/D = 5.0$ and $r/D = 0.6$. Energy spectrum is obtained by applying fast Fourier transform to time-history of the radial velocity and presented by the log-log scale. As shown in Fig. 7.9-(b), maximum peak energy is observed at frequency $fD/u_\infty = 0.20$, and this value can be considered as vortex-shedding frequency.

In the inertial subrange, the net energy delivered from the energy-containing eddies is in equilibrium with the net energy cascading to smaller scale eddies where it is dissipated, thus the slope of the energy spectrum in this range remains constant as -5/3, which is proved by Kolmogorov [49]. In Fig. 7.9-(b), however, the -5/3 Kolmogorov law is clearly confirmed in a particular range.

The mean streamwise velocity profiles (Fig. 7.10) along the y-direction are obtained at three different position: $x/D = 0.2$; $x/D = 1.6$; $x/D = 3.0$. The results of velocity profiles are compared with previous experimental and numerical results. Although, there are very minor discrepancies between $y/D = 0.5$ and 1.0, overall trend and the values of other region are in good agreement with previous data, especially explicit LES results of Yun et al. [105].

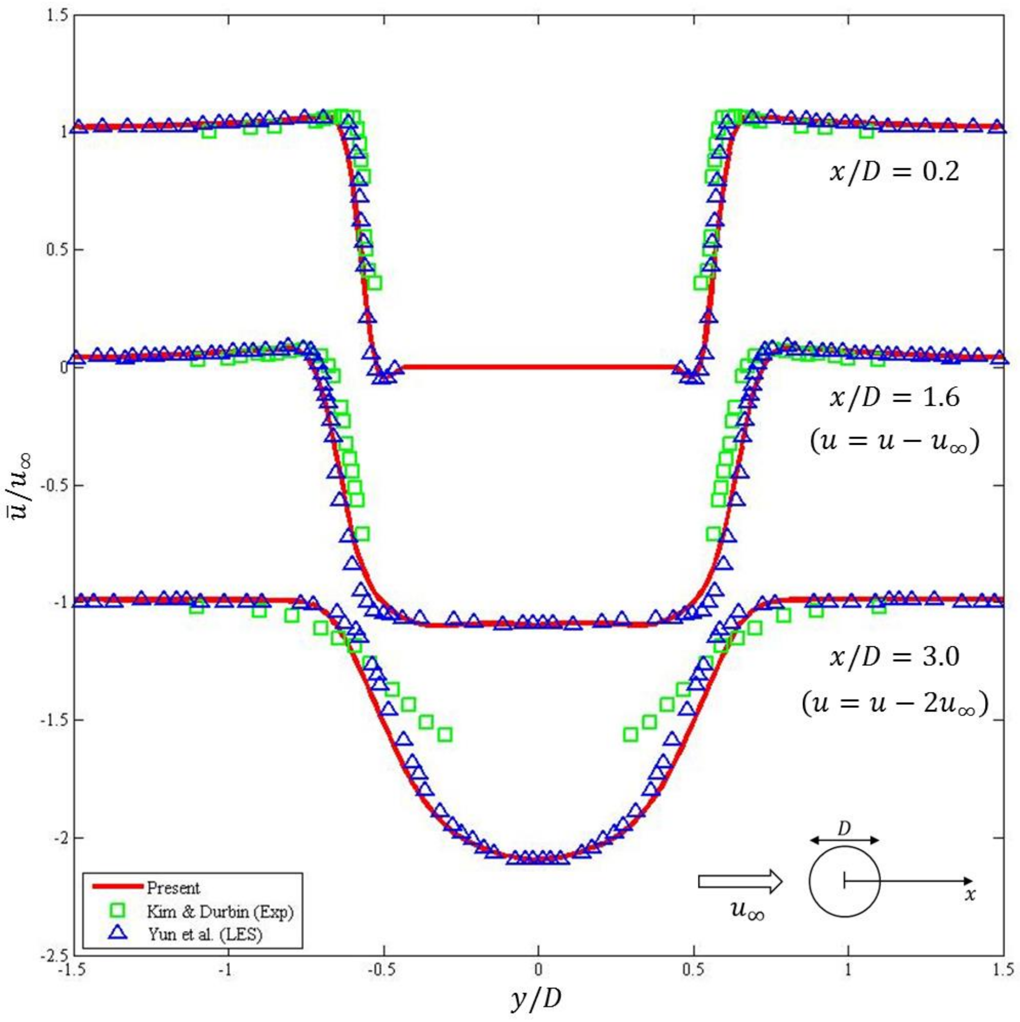


Fig. 7.10. Mean streamwise velocity profiles along the y-direction at different positions

The current ten-billions order large-scale computation produces huge output data, thus there are some problems such as data transmission, storage, and visualization. For this reason, the relatively small-scale problem using 67.1×10^6 grids is computed to visualize the simulation results. Based on low-resolution simulation results, the instantaneous vortical structure based on λ_2-definition is illustrated in Fig. 7.11 with three different views. In this figure, flow transition is clearly observable.

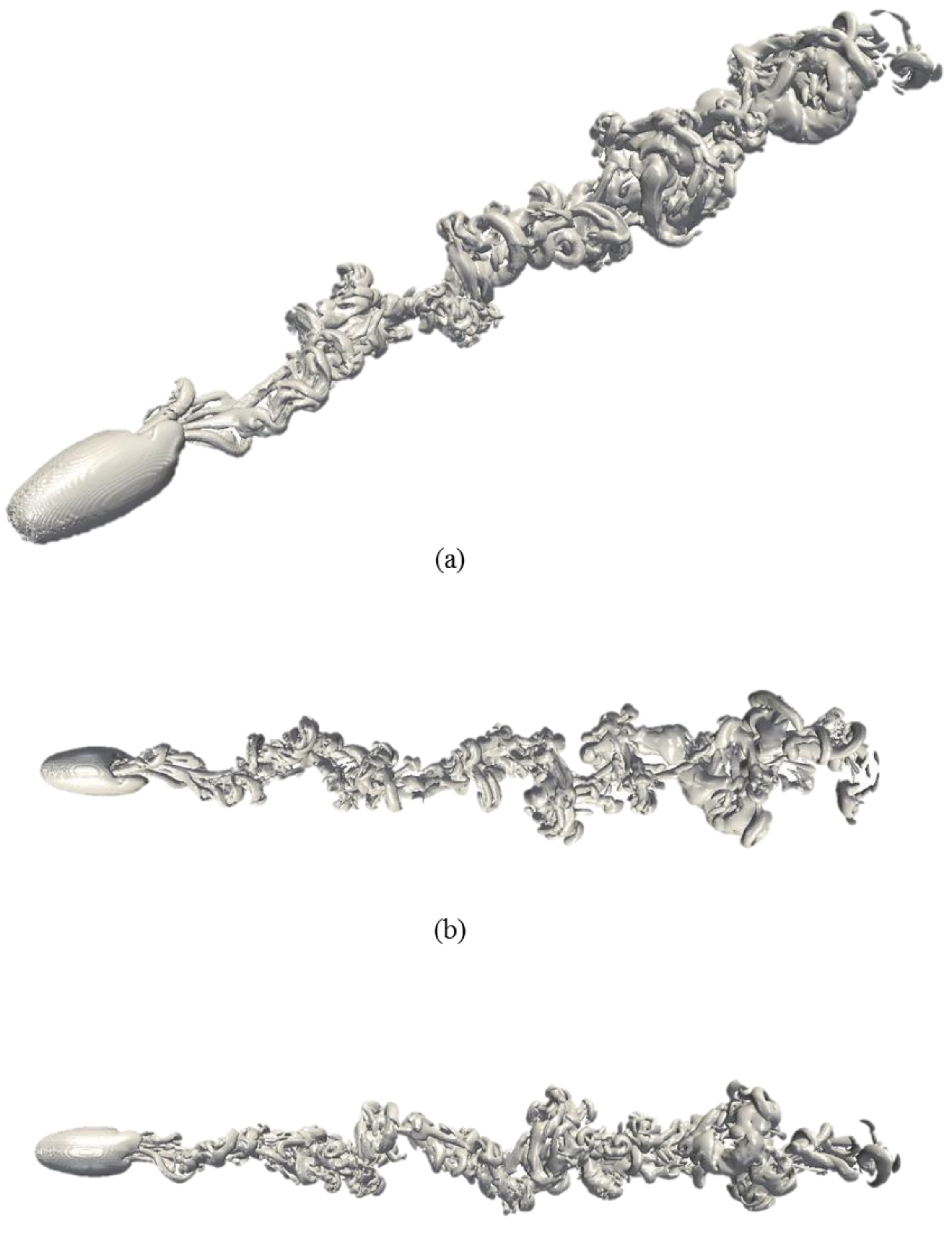


Fig. 7.11. Instantaneous vortical structures based on λ_2-definition at $Re = 3700$: (a) perspective view; (b) top view ($-y$ direction); (c) side view ($-z$ direction). These are results of the low-resolution simulation using 67.1millions uniform Cartesian grids.

7.2. Flow past a circular cylinder

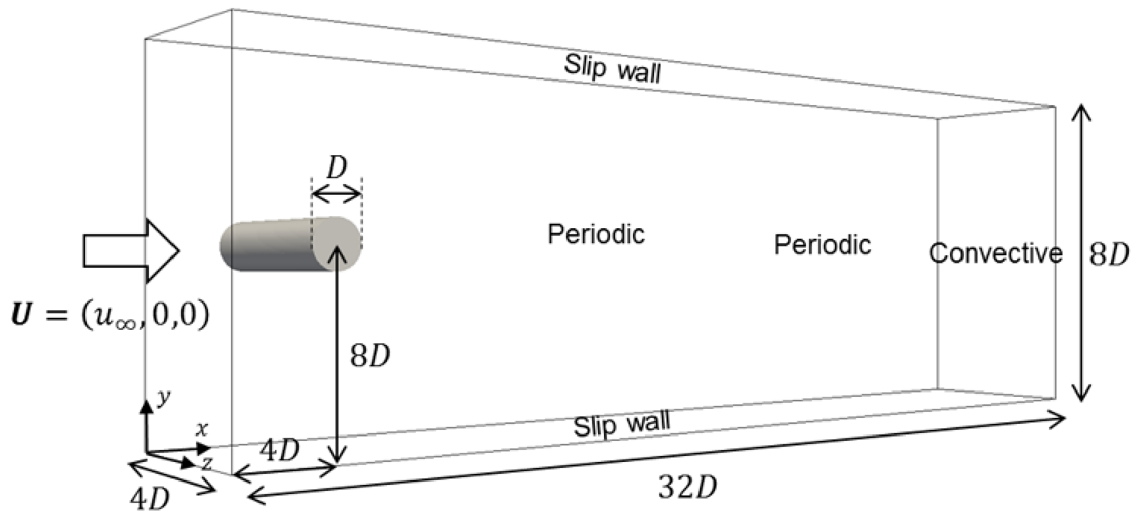


Fig. 7.12. Layout of the computational domain for flow past a circular cylinder problem.

The computational domain (see Fig. 7.12) is set to $[-4D, 28D] \times [-8D, 8D] \times [-2D, 2D]$ where D means the diameter of the sphere, which is positioned at the origin, and the number of grids is set to $2{,}048 \times 1{,}024 \times 256$ $(= 0.54 \times 10^9)$ where the 64 grids are positioned along the diameter of the sphere. As shown in Fig. 7.12, except the boundary of $z = -2D$ and $2D$, boundary conditions are equally imposed with previous sphere problems. The periodic boundary condition is imposed along z-direction. The Reynolds numbers (Re) based on the sphere diameter D and the free-stream velocity u_∞ are set to 3900 and 10000. The CFL number based on the free-stream velocity is set to 2.0.

The computational domain is divided into $32 \times 16 \times 4$ uniform blocks for conducting hybrid MPI/OpenMP parallel computation, thus $64 \times 64 \times 64$ grids are assigned to each MPI process. For each MPI process, 8 threads are assigned to apply OpenMP parallel computation. Therefore, total 16,384 cores (2,048 MPI process $\times$ 8 OpenMP threads) are employed to compute this problem. The computation is carried out on the *Nurion* supercomputer by using 256 nodes. For the single node, 8 MPI processes are allocated.

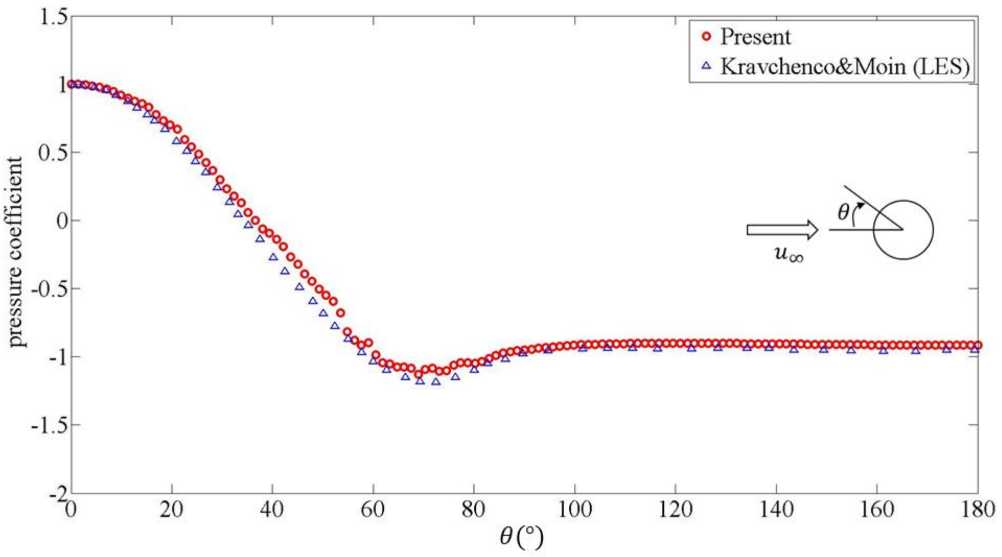
Table 7.3
Flow parameters of subcritical flow past a circular cylinder at $Re = 3900$

	$\overline{C_D}$	$\overline{C_{pb}}$	St	$\overline{L_{rec}}/D$
Bcaudan & Moin [109] (LES)	1.00	-0.95	0.203	1.36
Mittal & Moin [70] (LES)	1.00	-0.93	0.207	1.40
Kravchenko & Moin [114] (LES)	1.04	-0.94	0.210	1.35
Parnaudeau et al. [115] (Exp)	-	-	-	1.51
Parnaudeau et al. [115] (LES)	-	-	0.208	1.56
Present	1.09	-0.92	0.216	1.56

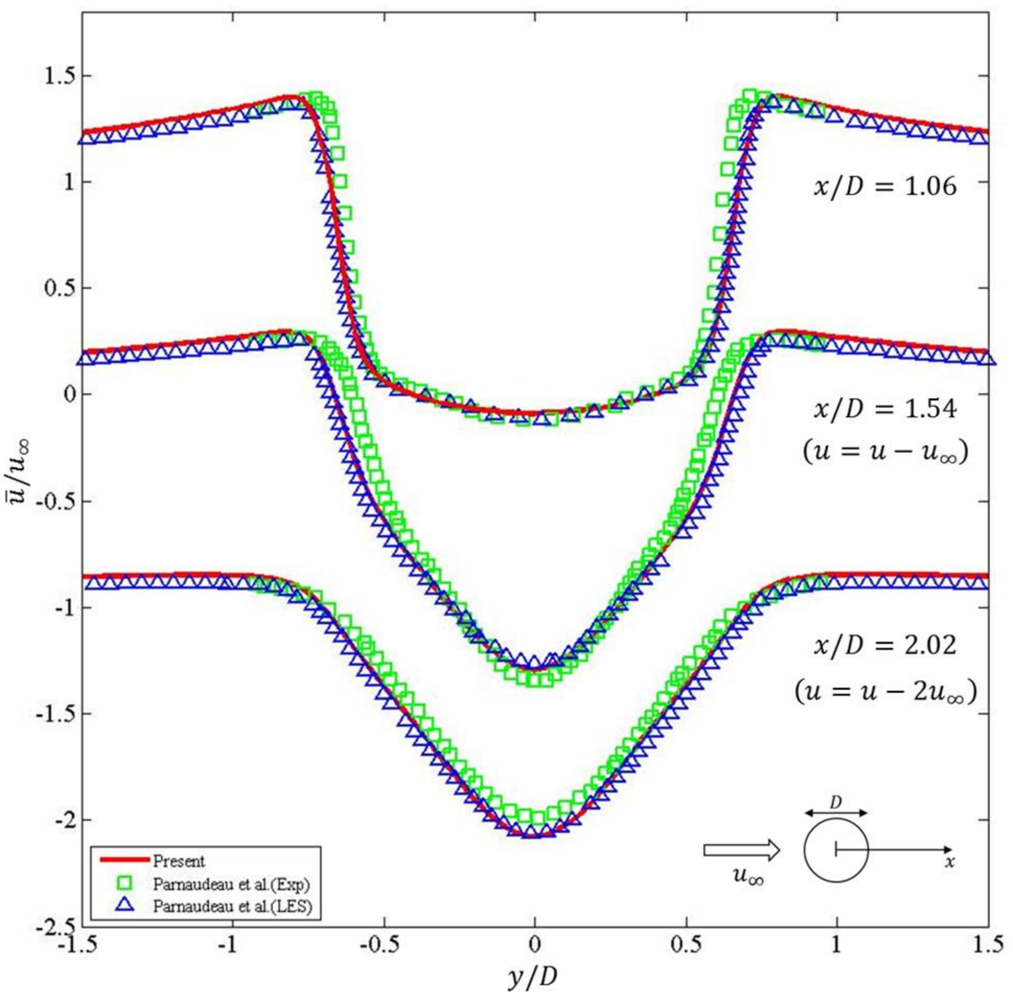
Flow parameters of the current simulation are summarized in Table 7.3, where the time-averaged drag coefficients ($\overline{C_D}$), base pressure coefficient ($\overline{C_{pb}}$), Strouhal number based on the vortex shedding frequency, and non-dimensional recirculation length ($\overline{L_{rec}}/D$) are presented together with previous experimental and numerical (LES) results [70, 109, 114, 115]. All results are in good agreement with previous data.

Fig. 7.13-(a) shows the comparison results of the mean pressure coefficient on the sphere surface along the azimuthal angle (θ). The current result is in good agreement with other result [114]. As shown in Fig. 7.13-(b), the mean streamwise velocity profiles along the y-direction are obtained at three different position: $x/D = 1.06$; $x/D = 1.54$; $x/D = 2.02$. The results of velocity profiles are compared with previous experimental and numerical results [115]. The comparison is considered to be very satisfactory.

Fig. 7.14 shows the time history and energy spectrum of the vertical velocity at $x/D = 10.0$ and $y/D = 0.0$. Energy spectrum is obtained by applying fast Fourier transform to time-history of the vertical velocity and presented by the log-log scale. In the energy spectrum, $-5/3$ Kolmogorov law is clearly observable. Fig. 7.15 shows the instantaneous vortical structure of flow around a circular cylinder.



(a)



(b)

Fig. 7.13. (a) Wall pressure coefficient and (b) distribution of the streamwise velocity along vertical direction at $Re = 3900$.

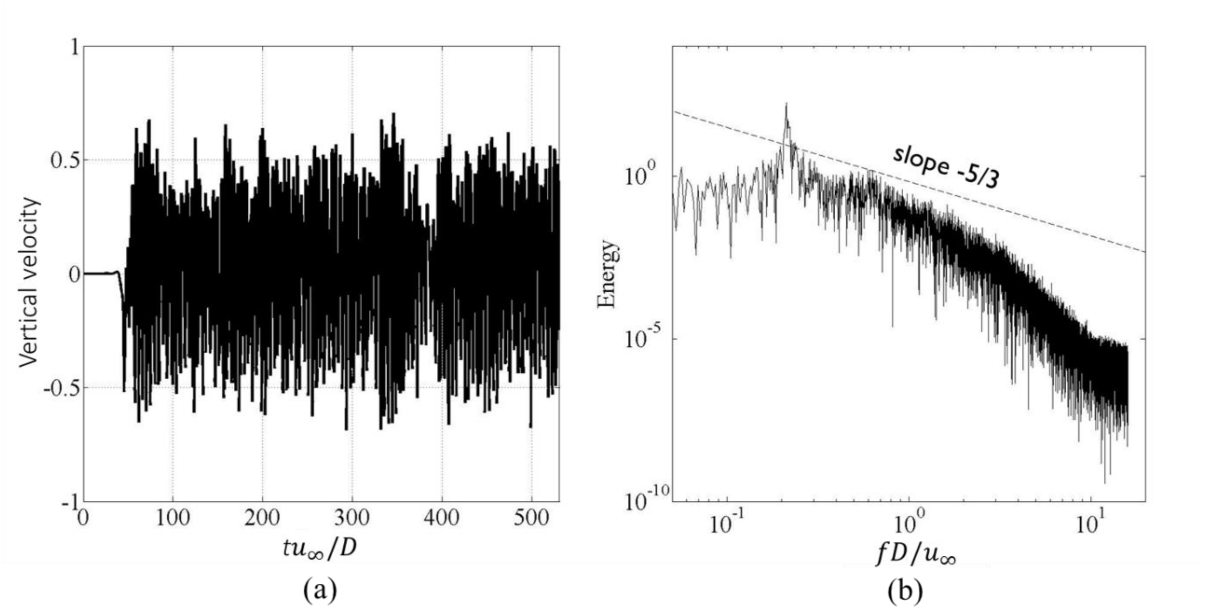


Fig. 7.14. (a) Time history and (b) Energy spectrum of the vertical velocity at $x/D = 10$ and $y/D = 0$. Here, the Reynolds number is 3900.

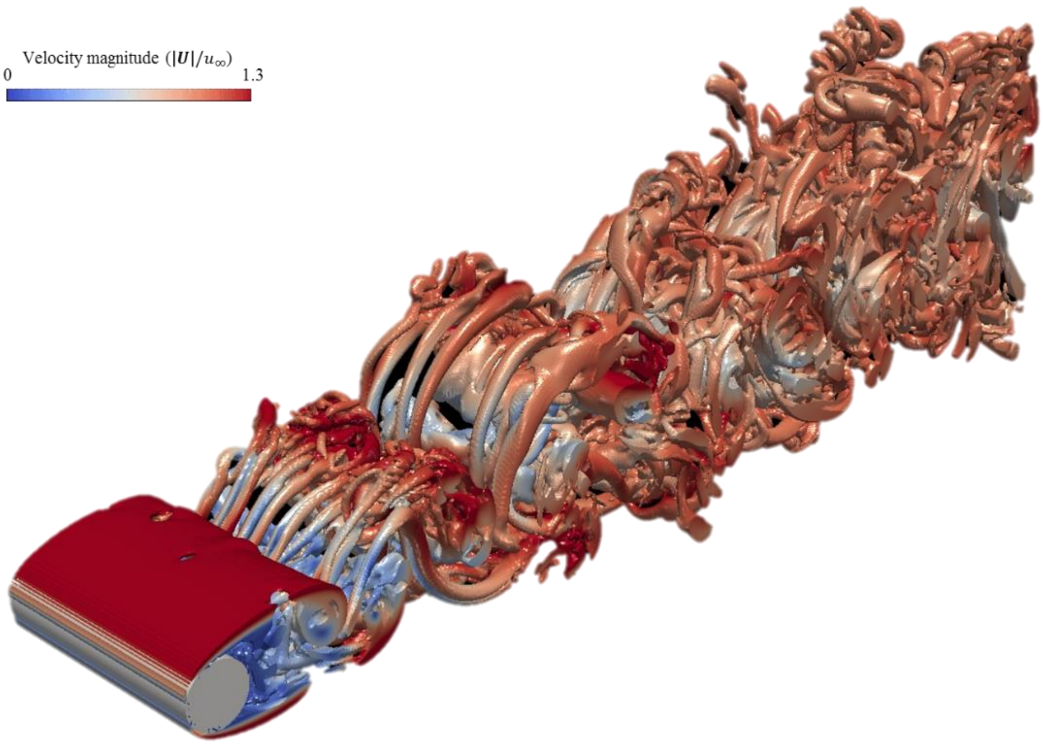


Fig. 7.15. Instantaneous λ_2-vortical structure of flow around a circular cylinder at $Re = 3900$

From now, flow over a circular cylinder at $Re = 10000$ is considered. Fig. 7.16 shows the wall pressure coefficient on the cylinder surface, and the current result is in good agreement with previous experimental [112] and numerical [111] results. Fig. 7.17 shows the time history and energy spectrum of the vertical velocity at $x/D = 10.0$ and $y/D = 0.0$. Energy spectrum is obtained by applying fast Fourier transform to time-history of the vertical velocity and presented by the log-log scale. In the energy spectrum, $-5/3$ Kolmogorov law is clearly confirmed.

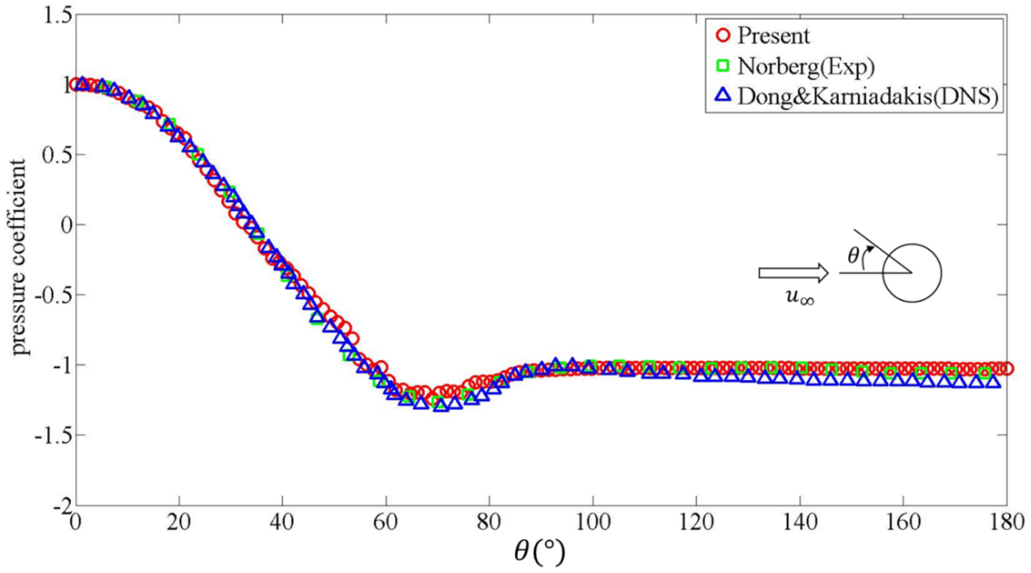
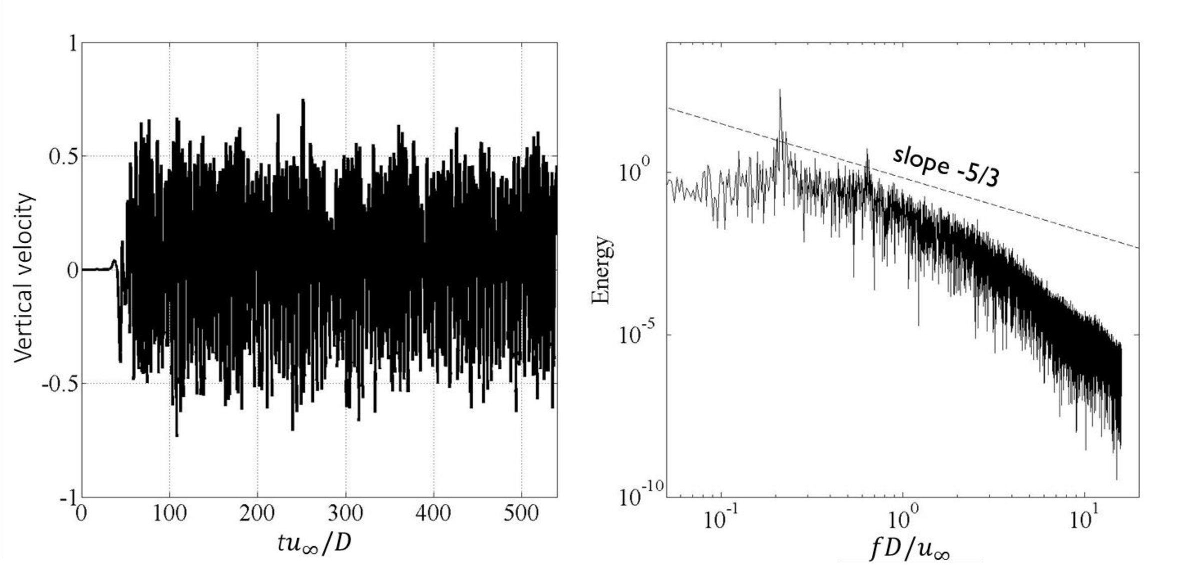


Fig. 7.16. Wall pressure coefficient on the cylinder surface at $Re = 10000$



(a) (b)

Fig. 7.17. (a) Time history and (b) Energy spectrum of the vertical velocity at $x/D = 10$ and $y/D = 0$. Here, the Reynolds number is 10000.

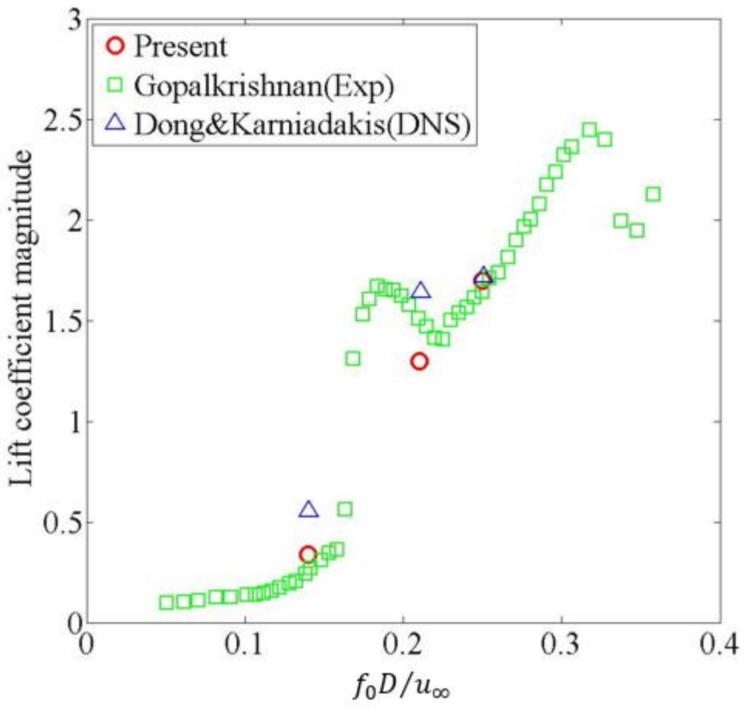


Fig. 7.18. Lift coefficient magnitudes versus non-dimensional frequency

At $Re = 10000$, oscillating cylinder is also considered. The cylinder oscillation in the cross flow (y) direction is given by $y/D = Y_0 \cos(2\pi f_0 t)$, where y is the cylinder displacement at time t, Y_0 is the amplitude, and f_0 is the frequency of the cylinder oscillation. In this case, Y_0 is 3.0 and f_0 is set to $0.14u_\infty/D$, $0.21u_\infty/D$, $0.25u_\infty/D$. Fig. 7.18 shows the lift coefficient magnitudes as a function of non-dimensional frequency. At $f_0D/u_\infty = 0.14, 0.25$, the current results are in good agreement with experimental results. However, due to resonance at around $f_0D/u_\infty = 0.21$, the discrepancy occurs between all results (current, previous experimental [112] and numerical [111] results).

Fig. 19 shows the λ_2-vortical structures with respect to time evolution at $Re = 10000$. Here, interestingly, three different wake patterns are observable. First, the vortex roller observed in laminar region is produced. Next, the flow transition occurs around the vortex roller. Finally, the vortex roller is collapsed, and then the fully developed turbulent flow is observed.

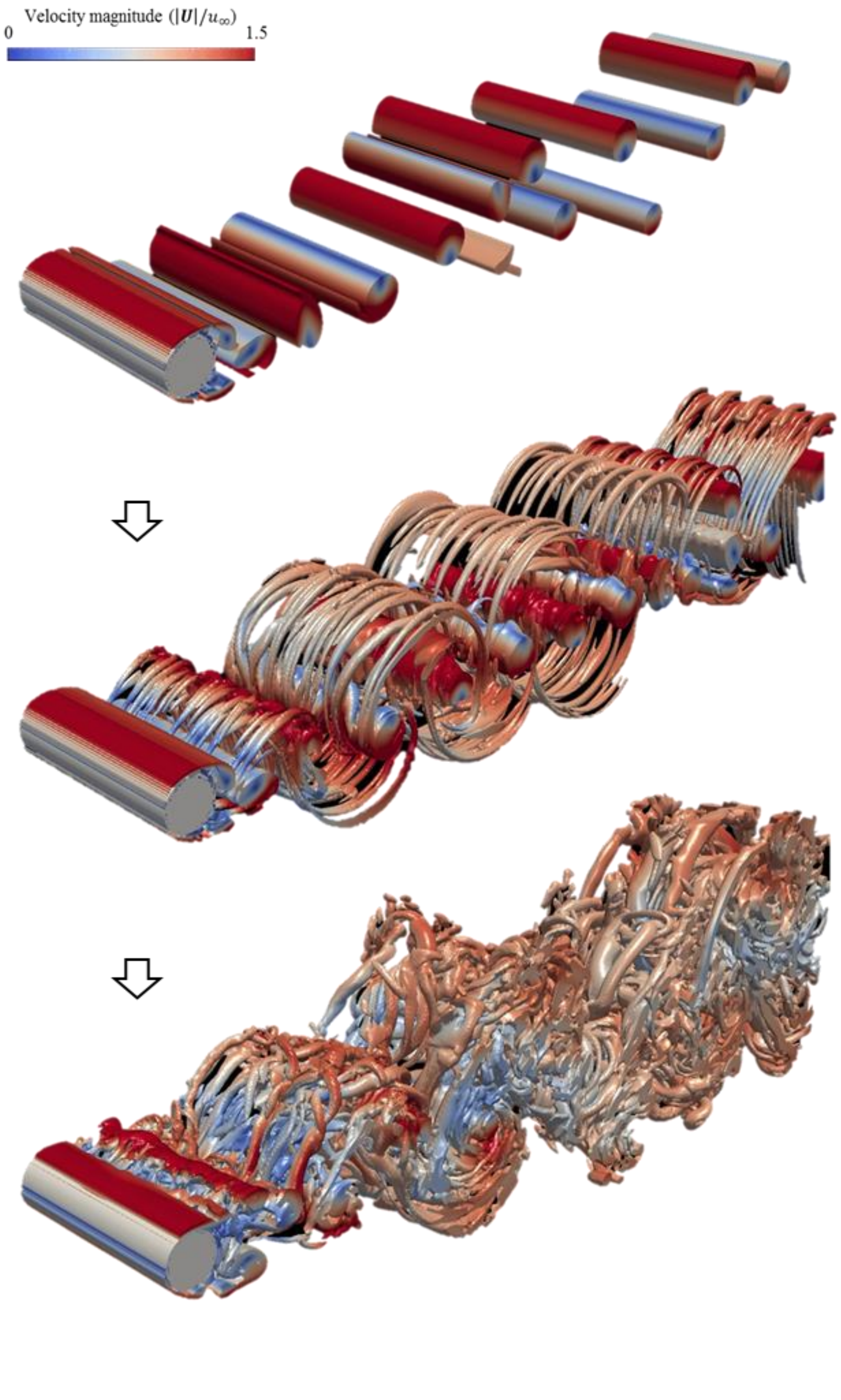


Fig. 7.19. Visualization of λ_2-vortical structures with respect to time evolution at $Re = 10000$. Three different wake patterns are observable: vortex roller, flow transition around vortex roller, and fully flow transition after collapsing vortex roller.

7.3. Flow past an underwater robot

Finally, the simulation results about flow past Crabster [108] are presented. Crabster is considered to be a perfect example of a realistic engineering object for demonstrating the effectiveness and robustness of the current methodology. In other words, the two major contributions of this dissertation, i.e. AMR-based SDF computation and the MG method for the irregular domain, can be effectively demonstrated through this demanding simulation.

Crabster is originally designed to carry out its mission by walking on the sea floor. When Crabster encounters a strong head current, it has the ability to change its posture by putting its head down to maintain its position. In terms of maintaining position, an accurate prediction of hydrodynamics resulting from this process is very important for the safe control of the robot. For this reason, we try to conduct a flow simulation around Crabster, which has three representative postures. An open-source 3D graphics software *Blender* (http://www.blender.org) is utilized to transform the original neutral posture of the Crabster to different heading postures. Fig. 7.20 shows the three different postures of Crabster with perspective and side views: head-up $10°$, neutral, and head-down $16°$.

The configuration of the computational domain is depicted in Fig. 7.21. In this figure, the background grid lines refer to the results of AMR for computing the SDF of a neutral position. Additionally, Fig. 7.22 illustrates the close-up view of AMR results for three different postures of Crabster. The entire flow domain size is $[-8B, 24B] \times [-4B, 4B]^2$ where B refers to the breadth of Crabster`s main body (see Fig. 7.21), and the $1024 \times 256 \times 256$ uniform Cartesian mesh system is utilized in the entire computational domain. The center of gravity for Crabster is located at $(0,0,0)$. As shown in Fig. 7.21, boundary conditions are equally set to the previous sphere problem. For all simulations, Reynolds number (Re) is based on the breadth B and the free-stream velocity u_∞, and the CFL number based on the free-stream velocity u_∞ is set to 4.

This problem is computed on the KISTI`s *Nurion* supercomputer. The computational domain is divided into $16 \times 4 \times 4$ uniform Cartesian blocks for MPI parallelization, thus total 256 MPI processes are assigned and 16 OpenMP threads are utilized at each MPI process. 64 KNL nodes of the *Nurion* supercomputer are used for computing this problem.

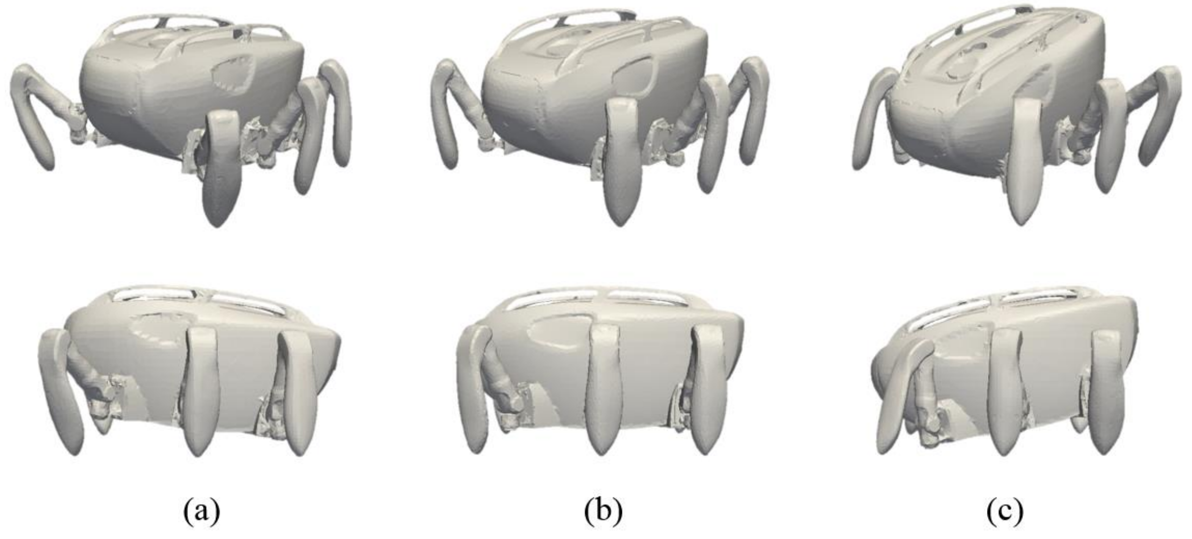


Fig. 7.20. Three different postures of Crabster: (a) head-up $10°$, (b) neutral, (c) head-down $16°$. Upper row shows perspective views, and lower row represents side views.

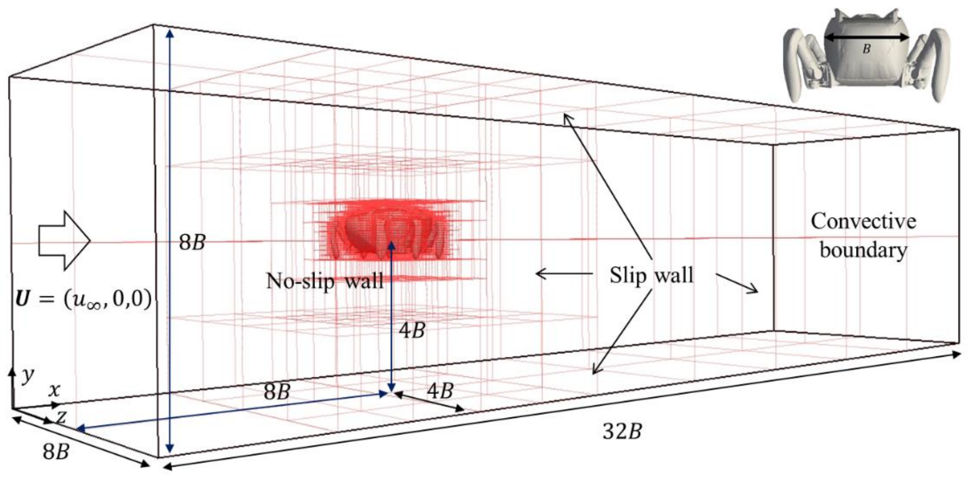
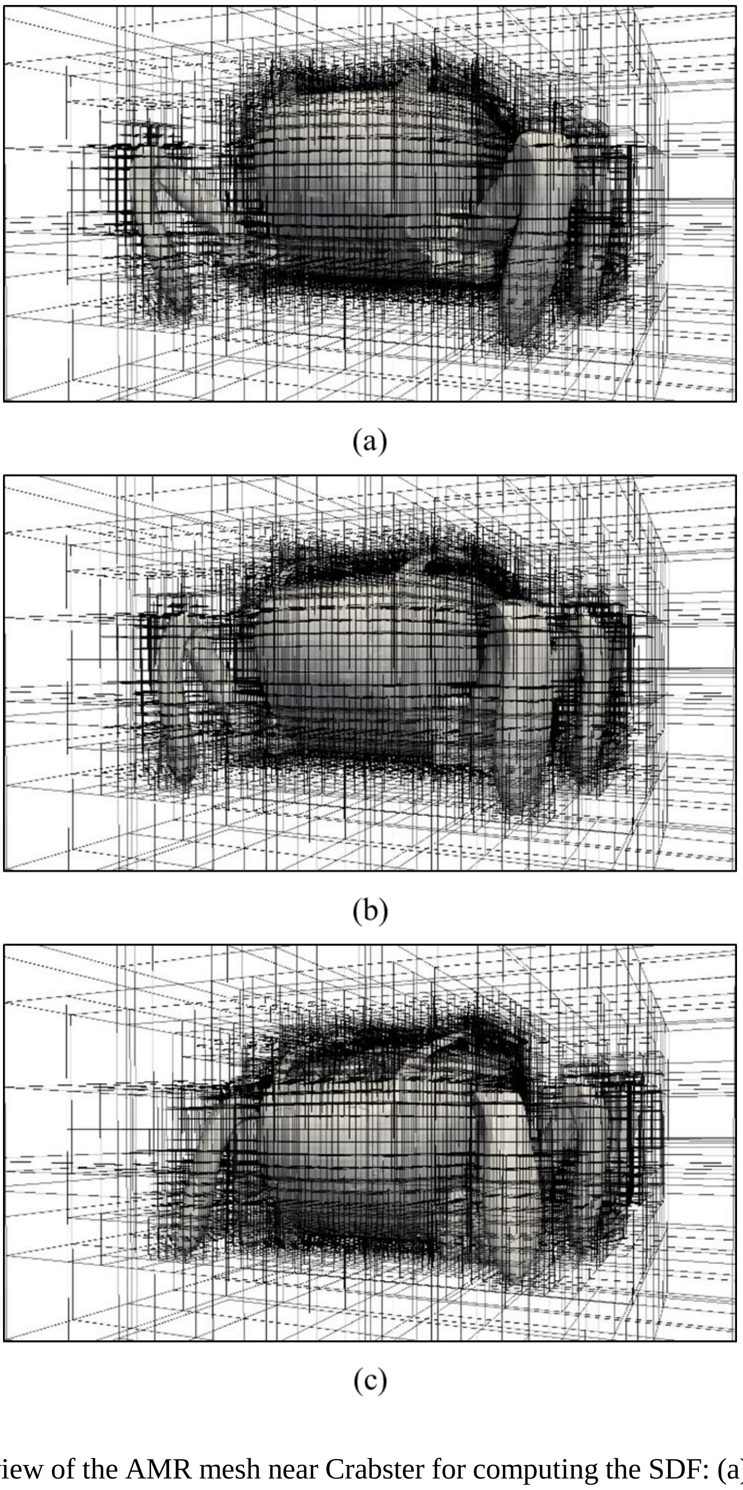


Fig. 7.21. Computational domain for flow past Crabster. Mesh line indicates the resulting process of AMR for determining the SDF.


(a)

(b)

(c)

Fig. 7.22. Close-up view of the AMR mesh near Crabster for computing the SDF: (a) head-up $10°$, (b) neutral, and (c) head-down $16°$.

7.3.1. Steady flow case

Since no available experimental data exists for Crabster, the simulation result is compared to the result of control volume analysis inspired by the wake-survey procedure [12] of experimental fluid mechanics. To validate the current simulation results, the hydrodynamic forces computed by the surface integral as shown in Eq. (6.6) are compared with the forces estimated by the wake-survey. Here, the entire computational domain is considered as the control volume employed for the wake-survey. Based on the Reynolds transport theorem [102], the total force acting on the fluid passing through the entire computational domain can be estimated as follows:

$$\boldsymbol{F} = \int \frac{\partial(\rho \boldsymbol{U})}{\partial t} d\Omega_f + \oint \rho \boldsymbol{U}(\boldsymbol{U} \cdot \boldsymbol{n_f})\, d\Gamma_f, \tag{7.1}$$

where $\boldsymbol{F}$ is the total force acting on the fluid, $\boldsymbol{U}$ is the fluid velocity, and $\boldsymbol{n_f}$ is the outward unit normal vector of the fluid region Ω_f, limited by its boundary Γ_f composed of an exterior and interior (body) boundary.

Here, $\boldsymbol{F}$ can be divided into

$$\boldsymbol{F} = \boldsymbol{F}_p^e + \boldsymbol{F}_v^e + \boldsymbol{F}_b, \tag{7.2}$$

where $\boldsymbol{F}_p^e$ and $\boldsymbol{F}_v^e$ refer to the pressure and viscous forces on the exterior boundary, and $\boldsymbol{F}_b$ indicates the force exerted by the interior body. The viscous forces on the exterior boundary can be assumed to be zero by considering the current boundary condition.

Here, the flow simulation over Crabster is conducted at $Re = 40$, where the flow field is steady. In this situation, the unsteady term of Eq. (7.1) can be safely ignored, and the wake-survey can be easily conducted. After all, the forces acting on the fluid exerted by Crabster at $Re = 40$ can be written as

$$\boldsymbol{F}_b = \oint \rho \boldsymbol{U}(\boldsymbol{U} \cdot \boldsymbol{n_f})\, d\Gamma_f - \boldsymbol{F}_p^e. \tag{7.3}$$

Due to Newton`s third law, namely the law of action-reaction, the hydrodynamic forces acting on Crabster can be estimated simply by $-\boldsymbol{F}_b$.

Fig. 7.23 depicts the vortical structures around Crabster at three different postures. Table 7.4 shows the force coefficients (drag and lift) predicted by the direct surface integral, i.e., Eq. (6.6), and the wake-survey, i.e., $-\boldsymbol{F}_b$. The comparison between these two methods is considered to be very satisfactory. Through this comparative study, the current approach can be indirectly validated.

Table 7.4
Comparison of force coefficients computed from the direct surface integral and the wake-survey for the flow past Crabster at $Re = 40$.

	Drag Coefficient C_D		Lift Coefficient C_L	
	Direct	Wake-survey	Direct	Wake-survey
Head-up	4.726	4.793	0.453	0.400
Neutral	4.626	4.666	0.148	0.122
Head-down	4.630	4.735	-0.395	-0.359

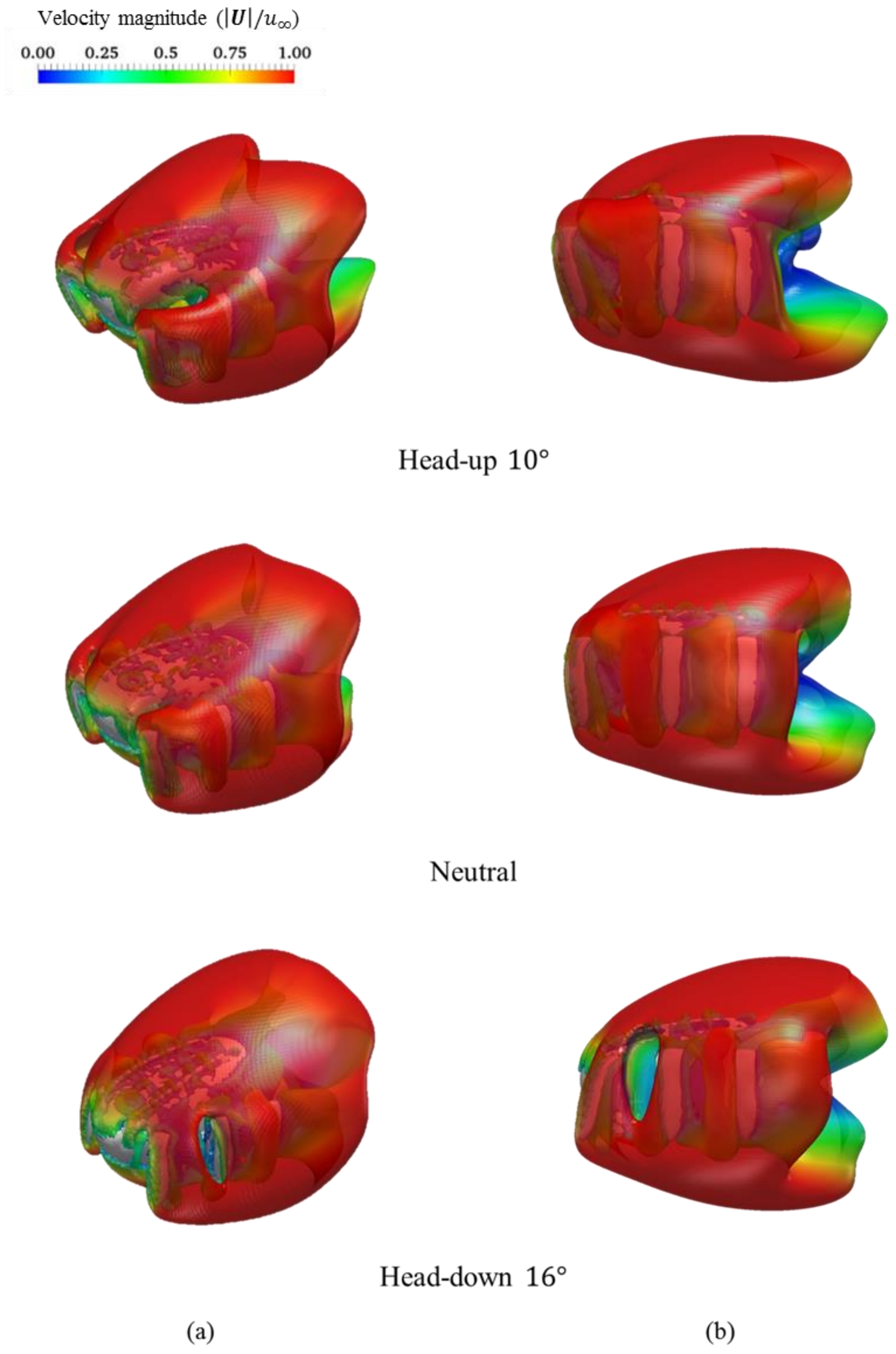


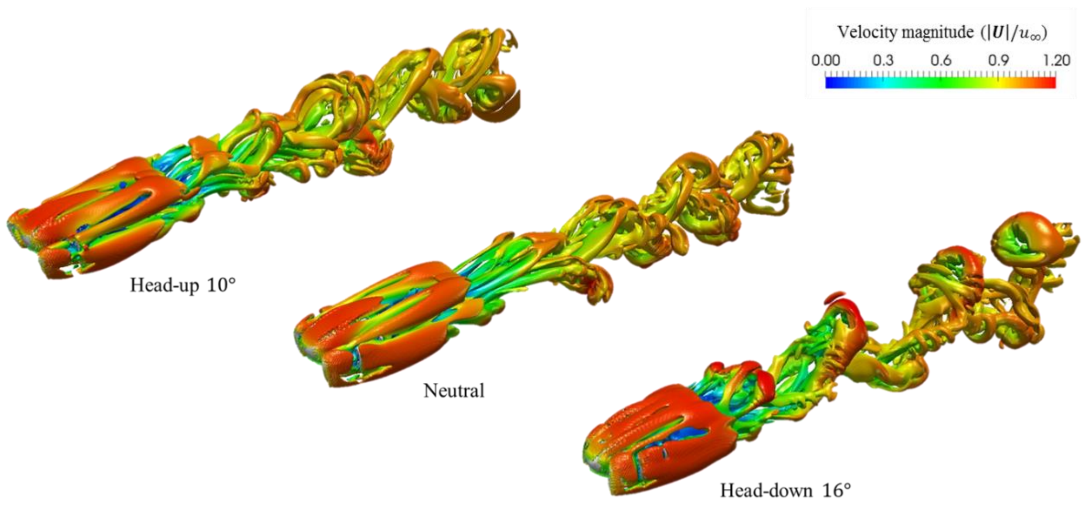
Fig. 7.23. Iso-surfaces of the vortical structure for flow around Crabster at $Re = 40$: (a) perspective view; (b) side view. Iso-surfaces are colored by velocity magnitude, which is divided by the free-stream velocity u_∞.

7.3.2. Unsteady flow case

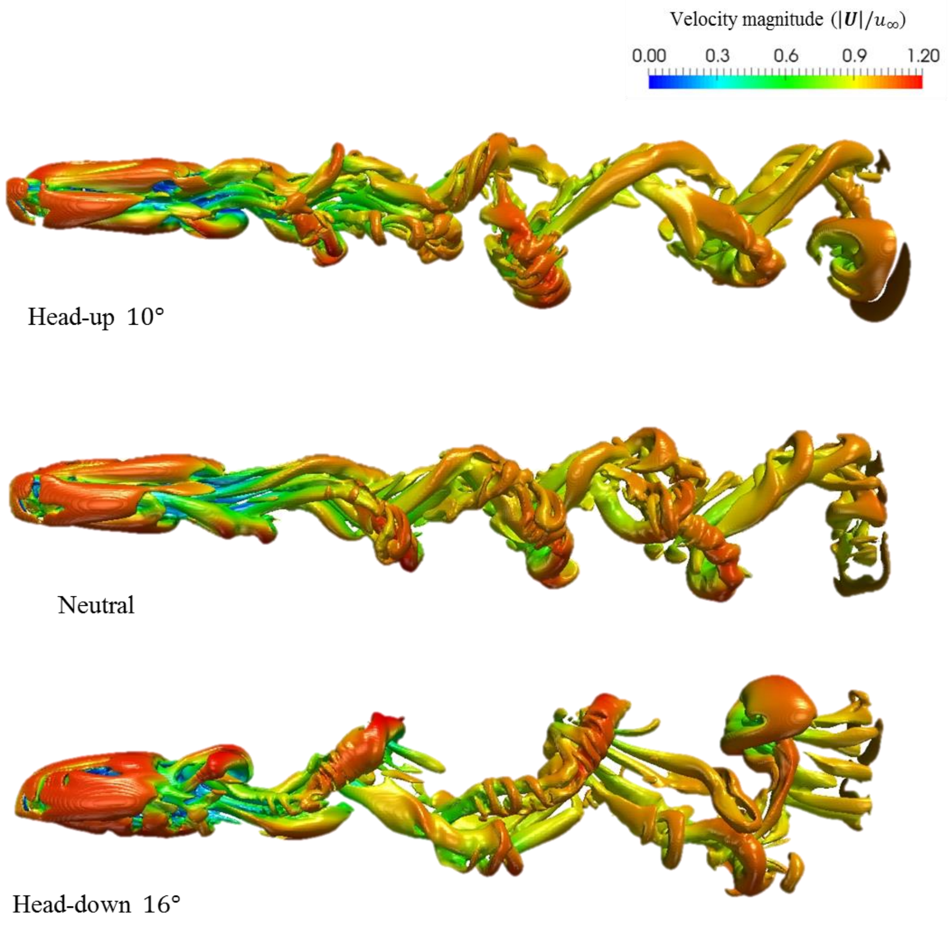
In order to analyze the unsteady flow around Crabster, simulations are performed with $Re = 1000$ based on the breadth of Crabster`s main body. Fig. 7.24 shows the simulation results with iso-surfaces of the vortical structure introduced by Jeong and Hussain [44], namely λ_2-definition, and this coherent structure is colored by velocity magnitude. Based on these results, we clearly demonstrate that the current MG method for irregular domain works perfectly well on a very complex irregular domain. Crabster occupies just a small portion of the computational domain and has detailed geometric features, including six slender legs. Nevertheless, the current Heaviside function restriction algorithm allows our MG method to successfully tackle the flow simulation around the complex irregular shape.

As shown in Fig. 7.24, the famous hairpin vortex [4] is observed at all cases. Note that the hairpin vortex consists of a head, neck, and leg, and the head of the hairpin vortex is the vortex core. As the profile of the Crabster changes, the directions of the hairpins also change. The head of the hairpin vortex is developed in the downward direction in both the head-up and neutral postures, whereas it is generated in the upward direction in the head-down case. This indicates the possibility of upward and downward lift depending on the heading angle of Crabster. After the rigorous validation study, this quantitative prediction data can certainly be utilized for design modifications and developing operational guidelines.

We also conduct flow over Crabster on bottom plate. As shown in Fig. 7.25, the bottom boundary condition is changed from slip to no-slip wall in this case. Other simulation conditions are same with the previous case. Actually, Crabster is the underwater walking robot on the sea bed, so this simulation, i.e. flow past Crabster on bottom plate, is more desired case for delivering helpful information towards actual operation of Crabster. Fig. 7.26 shows the snapshots of instantaneous vortical structures around Crabster and the possibility of the application of the current research for Cranbster in actual operating condition.



(a)



(b)

Fig. 7.24. Instantaneous iso-surfaces of the vortical structure for flow around Crabster at $Re = 1000$. Iso-surfaces are colored by velocity magnitude, which is divided by the free-stream velocity u_∞.

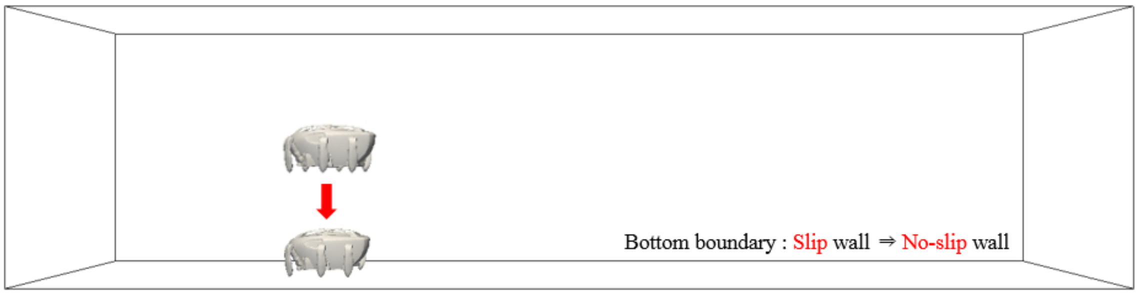


Fig. 7.25. Position variation of Crabster for analyzing flow around Crabster on the bottom no-slip wall.

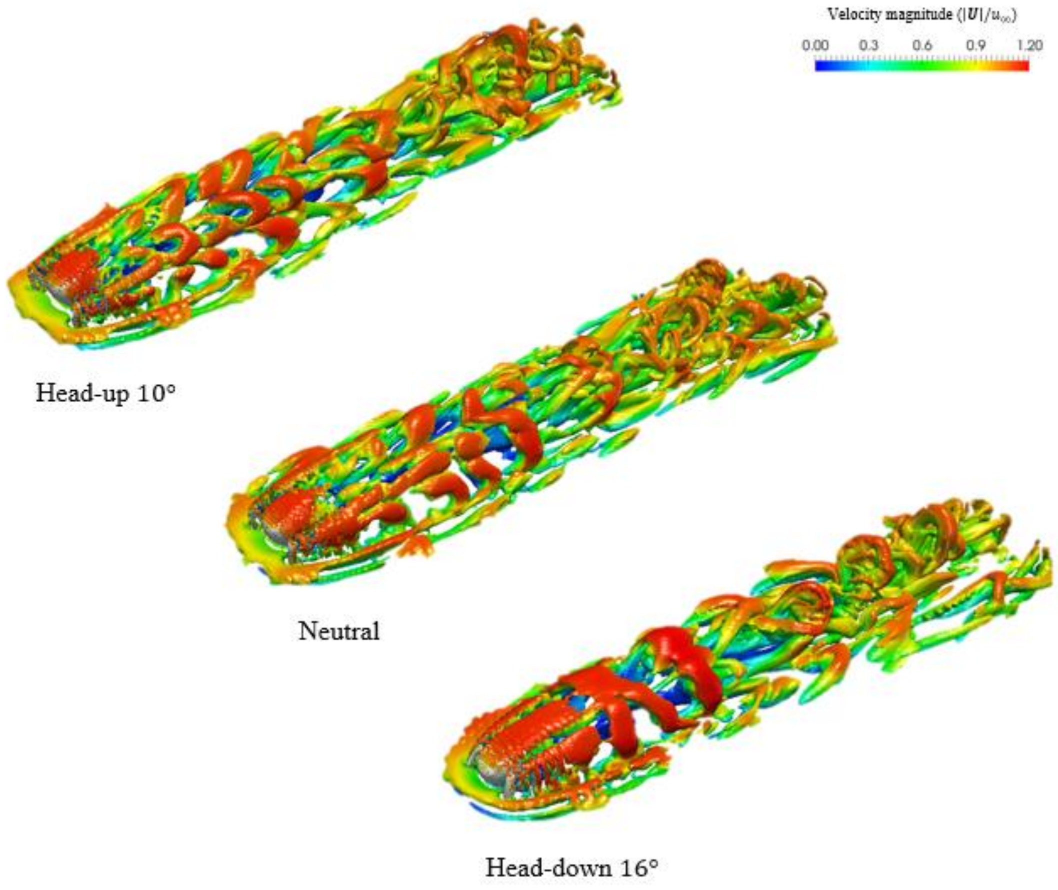


Fig. 7.26. Instantaneous iso-surface of the λ_2 vortical structure for flow past Crabster on the bottom boundary with respect to its attitudes. Iso-surfaces are colored by velocity magnitude, which is divided by the free-stream velocity u_∞.

7.4. Golf ball wake simulation

In this sub-chapter, numerical investigation of wake regions for flow past a stationary and a back-spinning golf ball is presented. There in an interesting physical phenomenon of fluid flow around a golf ball, namely drag reduction. It is well-known that the drag force of a golf ball is reduced by dimples of a golf ball at the particular state, which is at around $Re = 1.1 \times 10^5$, compared to a smooth sphere. The reason of this drag reduction is that flow separation at a golf ball occurs more behind than a smooth golf ball. Note that the drag is reduced as a region of flow separation decreases.

There are some successful studies based on experimental [8, 20] and numerical [91, 64, 23] approaches, which can predict drag reduction of a golf ball. However, wake regions behind a golf ball have not been clearly investigated by both experimental and numerical methods. The current uniform Cartesian mesh -based simulation is the most appropriate to investigate wake regions, so qualitative results of golf ball wake simulation is considered as primary concerns in this sub-chapter.

Fig. 7.27 depicts the layout of the computational domain for golf ball wake simulation. The computational domain is set to $[-4D, 28D] \times [-8D, 8D]^2$, where D means the diameter of the golf ball and the center of the golf ball is located at the origin. Boundary conditions are equally set to previous sphere cases. The Reynolds number of 1.1×10^5 is considered, and the back-spinning motion is given by spin rate (Γ) of 0.1. Here, spin rate is defined as $\Gamma = \omega R / u_\infty$ where ω means the angular velocity and R refers to the radius of the golf ball. The Reynolds number and spin rate chosen in this dissertation are based on the typical flying condition of the golf ball in real golf games.

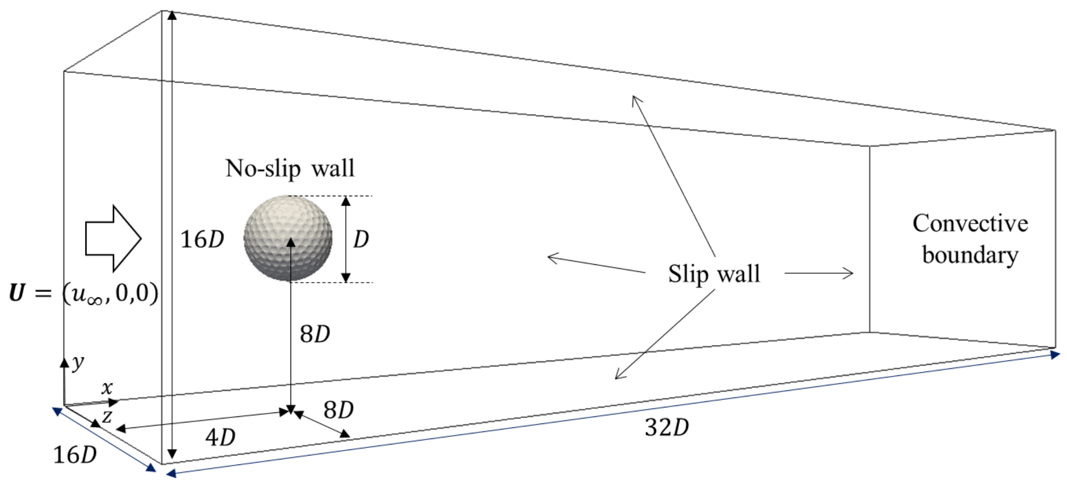


Fig. 7.27. Layout of the computational domain for golf ball wake simulation.

The number of grids is set to $4{,}096 \times 2{,}048^2$ ($\cong 17.2 \times 10^9$), resulting in 128 grids are located along the diameter of the golf ball. Based on the total number of grids, i.e. 17.2billions, this problem is the most large-scale simulation in this dissertation. By using hybrid MPI/OpenMP parallelization strategies, this very large-scale problem is computed on the KISTI`s *Nurion* supercomputer. Here, total 8,192 MPI processes (the computational domain is divided into $32 \times 16 \times 16$ uniform Cartesian blocks for MPI parallelization) are assigned at 512 Knight Landing (KNL) nodes, so 128^3 grids are included in each MPI process and 16 MPI processes are employed at each KNL node. 4 OpenMP threads are assigned at each MPI process, thus total 32,768 cores are utilized to conduct golf ball wake simulation with and without back-spin.

In the current research, the golf ball which has 392 dimples is considered. To represent the SDF of the golf ball with 392 dimples, the AMR-based SDF computation algorithm described in Chapter 4.1 is used. Fig. 7.28 shows the mesh refinement procedures for computing SDF of the golf ball. Refinement is carried out until the size of the coarsest grid is same with the size of grids for fluid simulation. SDF transformation strategies described in Chapter 4.7 is utilized to represent the motion of the back-spinning golf ball during unsteady fluid simulation on fixed Cartesian meshes.

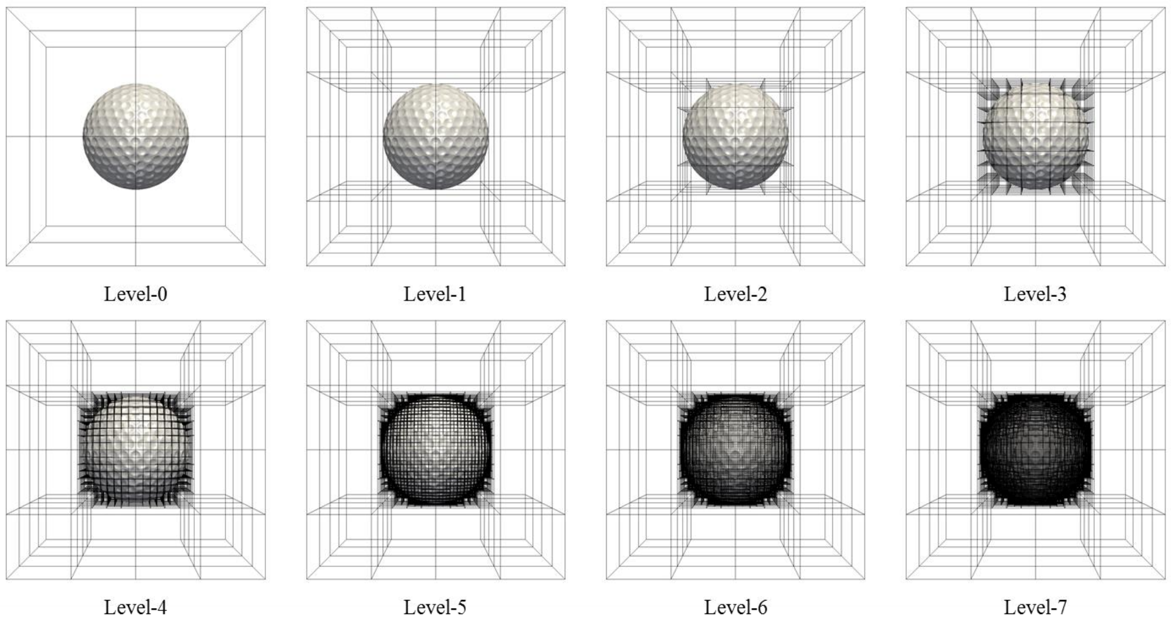


Fig. 7.28. AMR procedure for computing SDF of a golf ball with 392 dimples.

Because the problem size is too large, a huge amount of output files (vtk file format) for visualization is produced. Therefore, parallel visualization is required for this 10billions order problem to overcome memory limitation (more critical reason) and reduce operation time. In the current research, parallel visualization works are supported by visualization team of KISTI. An open-source visualization software, namely Paraview, is utilized. Visualization is also conducted on the *Nurion* supercomputer using 256 KNL nodes. Here, 256 MPI processes are utilized, and 68 OpenMP threads are assigned at each MPI process.

Figs. 7.29 and 7.30 respectively show the perspective and the side views of instantaneous vortical structures based on λ_2-definition of the golf ball, which is with and without back-spin. Based on knowledge of the fluid mechanics, it is known that the upward (positive) lift force can be induced by back-spinning effect and the wake region of a back-spinning sphere-type object is bent in downward direction. The current results are in good agreement with this fact. Differences between wake formation of the stationary and the back-spinning golf ball can be rather unclear because the spin rate is relatively small compared to the Reynolds number. However, different wake patterns are observable in Fig. 7.30. Flow patterns are straight in the wake region of the stationary golf ball, whereas, in the case of the back-spinning golf ball, flow is relatively bent in downward direction and more violent shedding is observed.

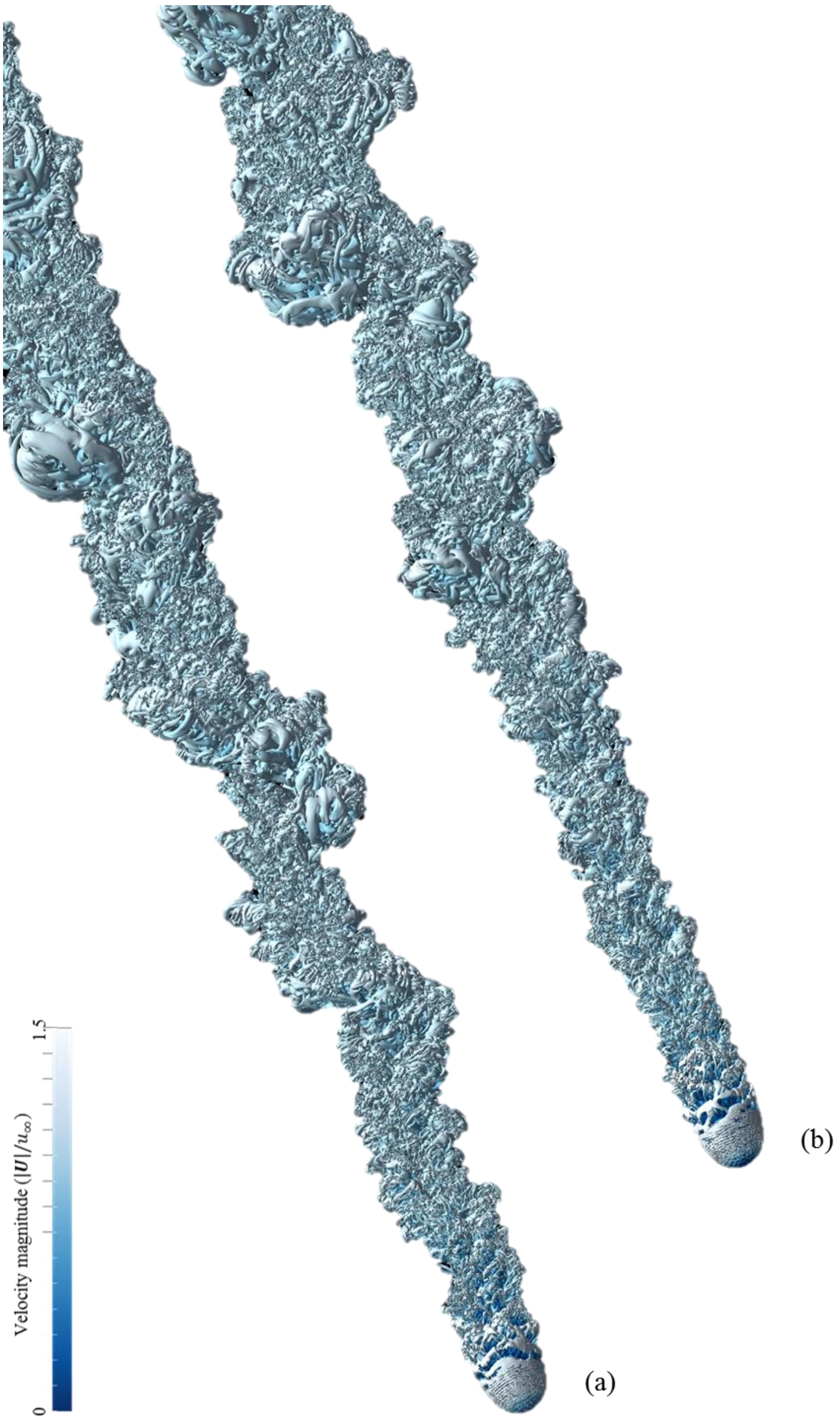


Fig. 7.29. Perspective views of instantaneous vortical structures based on λ_2-definition behind the golf ball: (a) stationary golf ball; (b) back-spinning golf ball.

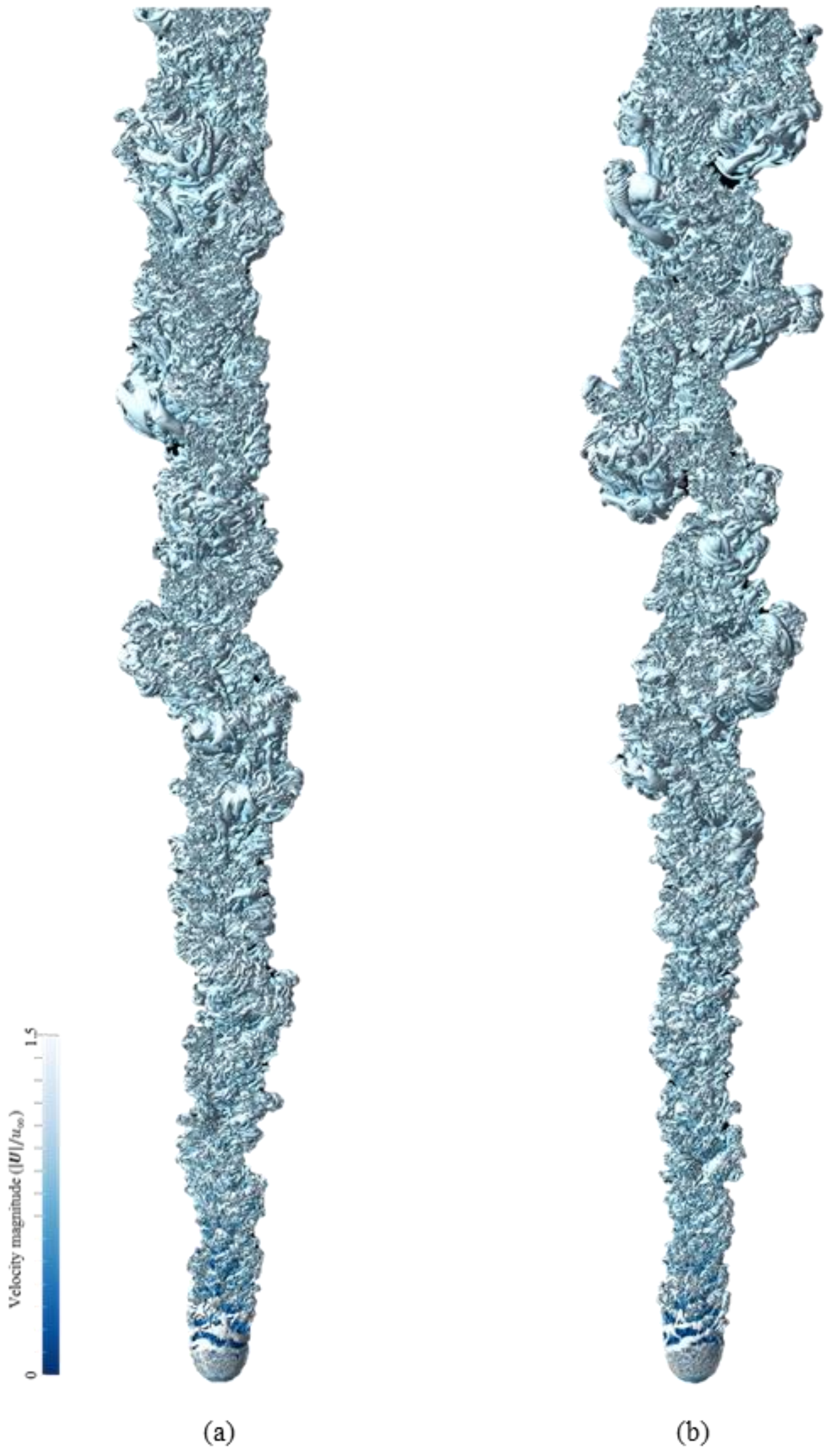


Fig. 7.30. Side views of instantaneous vortical structures based on λ_2-definition behind the golf ball: (a) stationary golf ball; (b) back-spinning golf ball.

Chapter 8

Conclusion and Future Work

Throughout this dissertation, two algorithms for efficient Cartesian mesh-based incompressible flow simulations over complex geometries were presented. Furthermore, the validity of the current approaches was demonstrated by both rigorous verification and validation studies. In this chapter, conclusions of the current research are drawn, and recommended future works are presented.

8.1. Conclusion

As the first contribution of this dissertation, the geometric MG algorithm on an irregular domain was presented. Although the fastest iterative solver of Poisson-type equations on the regular domain has been known as geometric MG, it has been rather unclear that MG is indeed the optimal solver for an irregular domain problem. Using a special treatment presented in the current study, namely the Heaviside function restriction, the optimal performance of MG was also obtained for irregular domain problems. The validity of current algorithm was demonstrated by solving an analytically defined model test problem on an irregular domain. Based on these results, we confirmed that geometric MG may also be the fastest (optimal) iterative solver of Poisson-type equations in an irregular domain.

As the second contribution of this dissertation, a novel efficient, robust, and reliable AMR-based SDF computation procedure was presented. The current refinement criterion is based on whether the cell boundary is intersected by the reference geometry or not, namely the edge/face-based criterion. This criterion guarantees that complicated engineering objects characterized by non-convexity, sharp corners, and multiple disjointed pieces can be exactly detected. Through the AMR procedure, the target region, i.e. the cell over the interface, is automatically detected, and only these interface cells are refined further. Therefore, the SDF computation process for the unconcerned pure cell region can be minimized, and the entire operation is computationally optimized. The current AMR-based algorithm was proved to be an order of magnitude faster than a naive SDF computation on a uniform mesh system by comparing the number of the sign determination procedure. Based on the volume convergence test, we confirmed that the computed SDF results in accurate volume estimation, which is second-order in 3D.

Furthermore, an efficient and accurate moving body representation technique, which is essential for the flow simulation over a rigid moving body, was presented. The significance of the current approach is that any redundant SDF computation, such as the re-initialization of the SDF, can be completely avoided regardless of body motion. The global SDF around a moving body is accurately and efficiently updated by a rigid body transformation using the initial SDF constructed by the AMR and a high-order representation within each Cartesian cell.

In order to solve a large-scale problem using the current algorithm for incompressible flows, parallelization strategies were presented. The hybrid MPI/OpenMP approach, which has been known as the most effective parallelization, was described together with the multi-level V-cycle algorithm, which is essential for the parallel multigrid method. Based on scalability evaluated by parallel computation on KISTI`s *Nurion* supercomputer, it was proved that the current incompressible flow solver is appropriate for parallel computation. Here, efficiencies of strong and weak scaling are 0.67 and 0.75, respectively.

The two contributions of this dissertation, i.e. MG method for an irregular domain and the AMR-based SDF computation and its extension for moving body representation, were applied to various flow simulations not only involving complex objects but also using the ten-billions order grid system. These simulation results were in good agreement with previous simulation results, experimental data, and physical intuition. Based on all these results, we concluded that if the geometry is given, regardless of its complexity and motion, the fluid simulation using Cartesian mesh system can be efficiently performed with the optimal iterative solver, namely the irregular domain MG method, and the parallel computation.

8.2. Future work

Based on the current research, following future works are highly recommended.

Through the current study, it was confirmed that incompressible single-phase flow simulation over arbitrarily complex geometry, which is stationary or has forced motion, can be efficiently and accurately carried out by using current algorithms. To apply the current incompressible solver for a wider range of engineering fields, extension to fluid-structure interaction (FSI) and multi-phase flow is highly required. The current solution algorithm for incompressible flows using Heaviside function were originally introduced for solving FSI problems, thus it is guaranteed to extension to FSI problems. However, how to efficiently treat SDF for a deformable body should be investigated in the future. There are many studies for analyzing multi-phase flow, such as volume-of-fluid (VOF) [40] and moment-of-fluid (MOF) [6, 7] methods. The current solution algorithm is unconditionally stable regardless of a size of a time step, so the stable algorithm for multi-phase flow is highly desired to keep the stability. For this reason, adaptive MOF method, which is stable and accurate, introduced by Ahn and Shashkov [7] can be considered as a potential candidate.

The current incompressible flow solver can be applied to solve very large-scale problems through multidimensional MPI parallelization and its combination with OpenMP, namely hybrid MPI/OpenMP parallelization. Actually, a problem size is not obstacle of the current method in terms of computation. However, post-processing can be considered as a potential barrier for computing a very large-scale problem. Flow visualization is very important to enhance understanding of flow physics, so researches for the parallel post-processing is recommend in the future. Furthermore, as a size of problems grows, a size of output data proportionally increases. For this reason, efficient and effective data management also should be considered as future works.

References

[1] H.T. Ahn, *A New Incompressible Navier-Stokes Method with General Hybrid Meshes and Its Application to Flow/Structure Interactions*, Ph.D. dissertation, University of Texas at Austin, 2005.

[2] H.T. Ahn, Y. Kallinderis, Strongly coupled flow/structure interaction with a geometrically conservation ALE scheme on general hybrid meshes, *Journal of Computational Physics* 219 (2) (2006) 671-696.

[3] H.T. Ahn, M. Shashkov, Multi-material interface reconstruction on generalized polyhedral meshes, *Journal of Computational Physics* 226 (2) (2007) 2096-2132.

[4] R.J. Adrian, Hairpin vortex organization in wall turbulence, *Physics of Fluids* 19 (2007) 041301, 1-16.

[5] H.T. Ahn, M. Shashkov, Geometric algorithms for 3D interface reconstruction, *in Proceeding of the 16th Int. Meshing Roundtable* 2008, 405-422.

[6] H.T. Ahn, M. Shashkov, M.A. Christon, The moment-of-fluid in action, *Communications in Numerical Methods in Engineering* 25 (2009) 1009-1018.

[7] H.T. Ahn, M. Shashkov, Adaptive moment-of-fluid method, *Journal of Computational Physics*. 228 (8) (2009) 2792-2821.

[8] P. Bearman, J. Harvey, Golf ball aerodynamics, *The Aeronautical Quarterly* 27 (2) (1976) 112-122.

[9] J. Benek, P. Buning, J. Steger, A 3-D chimera grid embedding technique, *in: Proceeding of the 7th AIAA Computational Fluid Dynamics Conference* 1985, 322-331.

[10] M. Braza, P. Chassaing, H.H. Minh, Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, *Journal of Fluid Mechanics* 165 (1986) 79–130.

[11] J.P. Boris, F. F. Grinstein, E.S. Oran, R.L Kolbe, New insights into large eddy simulation, *Fluid Dynamics Research* 10 (1992) 199-228.

[12] G.W. Brune, Quantitative Low-Speed Wake Surveys, *Journal of Aircraft* 31 (2) (1994) 249-255.

[13] W.L. Briggs, V.E. Henson, S.F. McCormick, *A Multigrid Tutorial: 2nd Ed.*, SIAM, Philadelphia, 2000.

[14] E. Balaras, Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations, *Computers & Fluids* 33 (3) (2004) 375-404.

[15] J.A. Baerentzen, H. Aanaes, Signed Distance Computation Using the Angle Weighted Pseudonormal, *IEEE Transactions on Visualization and Computer Graphics* 11 (3) (2005) 243-253.

[16] S.C. Brenner, L.R. Scott, *The mathematical theory of finite element methods: 3rd Ed.*, Springer, New York, 2008.

[17] I. Borazjani, L. Ge, F. Sotiropoulos, Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies, *Journal of Computational Physics* 227 (16) (2008) 7587-7620.

[18] A. Chorin, A numerical method for solving incompressible viscous flow problems, *Journal of Computational Physics* 2 (1967) 12-26.

[19] D. Calhoun, A Cartesian grid method for solving the two-dimensional stream function-vorticity equations in irregular region, *Journal of Computational Physics* 176 (2002) 231–275.

[20] J. Choi, W.P. Jeon, H. Choi, Mechanism of drag reduction by dimples on a sphere, *Physics of Fluids* 18 (2006) 0421702.1-4.

[21] J.I. Choi, R.C. Oberoi, J.R. Edwards, J.A. Rosati, An immersed boundary method for complex incompressible flows, *Journal of Computational Physics* 224(2) (2007) 757-784.

[22] S. Cai, A. Ouahsine, J. Favier, Y. Hoarau, Moving immersed boundary method, *International Journal for Numerical Methods in Fluids* 85 (2017) 288-323.

[23] J. Crabill, F. Witherden, A. Jameson, High-order computational fluid dynamics simulations of a spinning golf ball, *Sports Engineering*, in press.

[24] H. Dütsch, F. Durst, S. Becker, H. Lienhart, Low-Reynolds-number flow around an oscillating circular cylinder at low Keulegan-Carpenter numbers, *Journal of Fluid Mechanics* 360 (1998) 249-271.

[25] M. Detrixhe, F. Gibou, Hybrid massively parallel fast sweeping method for static Hamilton–Jacobi equations, *Journal of Computational Physics* 322 (2016) 199-223.

[26] H. Elman, D. Silvester, A. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, UK, 2005.

[27] D. Eberly, Distance Between Point and Triangle in 3D. https://www.geometrictools.com/Documentation/DistancePoint3Triangle3.pdf, 2017 (accessed 13 March 2018).

[28] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, *Journal of Computational Physics* 161 (1) (2000) 35-60.

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

[30] F. Gibou, R.P. Fedkiw, L.T. Cheng, M. Kang, A second-order-accurate symmetric discretization of the Poisson equation on irregular domains, *Journal of Computational Physics* 176 (1) (2002) 205-227.

[31] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, *Journal of Computational Physics* 207 (2) (2005) 457-492.

[32] F.F. Grinstein, L.G. Margolin, W.J. Rider, *Implicit Large Eddy Simulation: Computing Turbulent Fluid Dynamics*, Cambridge University Press, 2007.

[33] F. Gibou, C. Min, Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions, *Journal of Computational Physics* 231 (8) (2012) 3246-3263.

[34] G. Go, *An Efficient Algorithm for Free Surface Flow Simulation on Cartesian Meshes*, M.S. dissertation, University of Ulsan, South Korea, 2013.

[35] G. Go, H.T. Ahn, Fluid-Body Interaction Analysis of Floating Body in Three Dimensions, *Journal of Computational Fluids Engineering* 20 (2) (2015) 103-108.

[36] G. Go, H.T. Ahn, A Geometric Multigrid Accelerated Incompressible Flow Simulation Around an Arbitrarily Complex Object on Cartesian Grids, *under review*.

[37] G. Go, H.T. Ahn, Cartesian mesh-based flow simulation around a moving body using an efficient and accurate SDF transformation, *under review*.

[38] F. Gibou, R. Fedkiw, S. Osher, A review of level-set methods and some recent applications, *Journal of Computational Physics* 353 (2018) 82-109.

[39] F.H. Harlow, J.E. Welch, Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface, *Physics of Fluids* 8 (1965) 2182-2189.

[40] C. Hirt, B.D. Nicholls, Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries, *Journal of Computational Physics* 39 (1981) 201-225.

[41] W.D. Henshaw, A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids, *Journal of Computational Physics*. 113 (1) (1994) 13-25.

[42] G.D. Ilio, D. Chiappini, S. Ubertini, G. Bella, S. Succi, Fluid flow around NACA0012 airfoil at low-Reynolds numbers with hybrid lattice Boltzmann method, *Computers & Fluids* 166 (30) (2018) 200-208.

[43] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, *Applied Mechanics Reviews* 56 (3) (2003) 331-347.

[44] J. Jeong, F. Hussain, On the identification of a vortex, *Journal of Fluid Mechanics* 285 (1995) 69-94.

[45] G.S. Jiang, C.W. Shu, Efficient Implementation of Weighted ENO Schemes, *Journal of Computational Physics* 126 (1) (1996) 202-228.

[46] T.A. Johnson, *Numerical and Experimental Investigation of Flow past a Sphere up to a Reynolds Number of 300*, Ph.D. dissertation, University of Iowa, USA, 1996.

[47] T.A. Johnson, V.C. Patel, Flow past a sphere up to a Reynolds number of 300, *Journal of Fluid Mechanics* 378 (1999) 19-70.

[48] G. Jo, D.Y. Kwak, Geometric multigrid algorithms for elliptic interface problems using structured grids, published online, *Numerical Algorithms* (2018) 1-25. https://doi.org/ 10.1007/s11075-018-0544-9.

[49] A.N. Kolmogorov, The Local Structure of Turbulence in Incompressible Viscous Fluid for Very Large Reynolds Number, *Doklady Akademiia Nauk SSSR* 30 (1941) 301-305.

[50] H.J. Kim, P.A. Durbin, Observations of the frequencies in a sphere wake and of drag increase by acoustic excitation, *Physics of Fluids* 31 (11) (1988) 3260-3265.

[51] C.T. Kelly, *Iterative Methods for Linear and Nonlinear Equations*, Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[52] J. Kim, D. Kim, H. Choi, An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries, *Journal of Computational Physics*. 171(1) (2001) 132-150.

[53] Y. Kallinderis, H.T. Ahn, Incompressible Navier-Stokes method with general hybrid meshes, *Journal of Computational Physics* 210 (1) (2005) 75-108.

[54] Y. Kallinderis, C. Kavouklis, A dynamic adaptation scheme for general 3-D hybrid meshes, *Computer Methods in Applied Mechanics and Engineering* 194 (48-49) (2005) 5019-5050.

[55] D. Kim, Laminar flow past a sphere rotating in the transverse direction, *Journal of Mechanical Science and Technology* 23 (2009) 578-589.

[56] C. Kavouklis, Y. Kallinderis, Parallel adaptation of general three-dimensional hybrid meshes, *Journal of Computational Physics* 229 (9) (2010) 3454-3473.

[57] D. Kurtulus, On the unsteady behavior of the flow around NACA0012 airfoil with steady external conditions at Re=1000, *International Journal of Micro Air Vehicles* 7 (2015) 301-326.

[58] M.E. Khalili, M. Larsson, B. Müller, Immersed boundary method for viscous compressible flows around moving bodies, *Computers & Fluids* 170 (2018) 77-92.

[59] C. Kavouklis, P. Colella, Computation of Volume Potentials on Structured Grids with the Method of Local Corrections, *Communications in Applied Mathematics and Computational Science,* in press.

[60] C. Liu, X. Zheng, C.H. Sung, Preconditioned Multigrid Methods for Unsteady Incompressible Flows, *Journal of Computational Physics* 139 (1998) 35-57.

[61] Y. Liu, K. Li, J. Zhang, H. Wang, L. Liu, Numerical bifurcation analysis of static stall of airfoil and dynamic stall under unsteady perturbation, *Communications in Nonlinear Science and Numerical Simulation* 17 (2012) 3427-3434.

[62] H. Luo, H. Dai, P.J.S.A. Ferreira de Sousa, B. Yin, On the numerical oscillation of thedirect-forcing immersed-boundary method for moving boundaries, *Computers & Fluids* 56 (2012) 61-76.

[63] C. Liu, C. Hu, An efficient immersed boundary treatment for complex moving object, *Journal of Computational Physics* 274 (2014) 654-680.

[64] J. Li, M. Tsubokura, M. Tsunoda, Numerical Investigation of the Flow Past a Rotating Golf Ball and Its Comparison with a Rotating Smooth Sphere, *Flow, Turbulence and Combustion* 99 (3-4) (2017) 837-864.

[65] E. Lee, H.T. Ahn, H. Luo, Cell-centered high-order hyperbolic finite volume method for diffusion equation on unstructured grids, *Journal of Computational Physics* 335 (2018) 464-491.

[66] E. Lee, H.T. Ahn, A reconstruction-based cell-centered high-order finite volume method for incompressible viscous flow simulation on unstructured meshes, *Computers & Fluids* 170 (2018) 187-196.

[67] R. Mittal, G. Iaccarino, Immersed boundary methods, *Annual Review of Fluid Mechanics*, 37 (2005) 239-261.

[68] J. Mohd-Yusof, Combined immersed boundaries/B-splines methods for simulation of flows in complex geometries, CTR Annual Research Briefs, NASA Ames/Stanford University, (1997) 317-327.

[69] T. Möller, B. Trumbore, Fast, Minimum Storage Ray-Triangle Intersection, *Journal of Graphics Tools* 2 (1) (1997) 21-28.

[70] R. Mittal, P. Moin, Suitability of upwind-biased finite-difference schemes for large-eddy simulation of turbulent flows, *AIAA Journal* 35 (1997) 1415-1417.

[71] C. Min, F. Gibou, A second order accurate projection method for the incompressible Navier-Stokes equations on non-graded adaptive grids, *Journal of Computational Physics* 219 (2) (2006) 912-929.

[72] Y.T. Ng, C. Min, F. Gibou, An efficient fluid-solid coupling algorithm for single-phase flows, *Journal of Computational Physics* 228 (23) (2009) 8807-8829.

[73] C. Norberg, Flow around a Circular Cylinder: Aspects of Fluctuating Lift, *Journal of Fluids and Structures* 15 (3-4) (2001) 459-469.

[74] K. Nakajima, OpenMP/MPI Hybrid Parallel Multigrid Method on Fujitsu FX10 Supercomputer System, *in: Proceeding of IEEE International Conference on Cluster Computing Workshops* 2012.

[75] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, New York, 2002, pp.17-22.

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

[77] O. Pironneau, On the Transpor-Diffusion Algorithm and Its Applications to the Navier-Stokes Equations, *Numerical Mathematics* 38 (1982) 309-332.

[78] H. Park, X. Pan, C. Lee, J.I. Choi, A pre-conditioned implicit direct forcing based immersed boundary method for incompressible viscous flows, *Journal of Computational Physics* 314 (2016) 774-799.

[79] P.O. Persson, C. Fidkowski, Test Case CL1 - Heaving and Pitching Airfoil. https://acdl.mit.edu/HOW5/WorkshopPresentations/CL1_HeavingAndPitchingAirfoil/CL1_0_IntroSummary.pdf, 2018 (accessed 13 January 2019).

[80] P.L. Roe, Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes, *Journal of Computational Physics* 43 (1981) 357-372.

[81] D. Russell, Z.J. Wang, A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow, *Journal of Computational Physics* 191 (2003) 177–205.

[82] R. Rabenseifner, G. Hager, G. Jost, Hybrid MPI/OpenMP parallel programing on clusters of multi core SMP nodes, *in: Proceeding of the 17 Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, Weimar, Germany, 2009, 427-436.

[83] I. Rodriguez, R. Borell, O. Lehmkuhl, C.D.P. Segarra, A. Oliva, Direct numerical simulation of the flow over a sphere at $Re = 3700$, *Journal of Fluid Mechanics* 679 (2011) 263-287.

[84] J. Senocak, M. Sandusky, R. Deleon, D. Wade, K. Felzein, M. Budnikova, An immersed boundary geometric preprocessor for arbitrarily complex terrain and geometry, *Journal of Atmospheric and Oceanic Technology* 32 (11) (2015) 2075-2087.

[85] M. Shimrat, Algorithm 112: Position of point relative to polygon, *Communications of the ACM* 5 (8) (1962) 425.

[86] H. Schlichting, *Boundary Layer Theory*, 7th Edition, McGraw-Hill, 1979.

[87] H, Sakamoto, H. Haniu, A study on vortex shedding from spheres in a uniform flow, *Journal of Fluids Engineering* 112 (1998) 386-392.

[88] J. Stam, Stable fluids, *in: Proceeding of ACM Siggraph* 1999, 121-128.

[89] G. Strang, Multigrid Methods, Section 6.3 in lecture note of Mathematical Methods for EngineerisⅡ, Department of Mathematics, MIT, USA, 2006 (DOI: http://math.mit.edu/classes/18.086/2006/).

[90] S. Sen, S. Mittal, G. Biswas, Steady separated flow past a circular cylinder at low Reynolds numbers, *Journal of Fluid Mechanics* 620 (2009) 89-119.

[91] C.E. Smith, N. Beratlis, E. Balaras, K. Squires, M. Tsunoda, Numerical investigation of the flow over a golf ball in the subcritical and supercritical regimes, *International Journal of Heat and Fluid Flow* 31 (3) (2010) 262-273.

[92] R. Stephens, Find the shortest distance between a point and a line segment in C#. http://csharphelper.com/blog/2016/09/find-the-shortest-distance-between-a-point-and-a-line-segment-in-c/, 2016 (accessed 13 March 2018).

[93] G. Sun, J.A. Domaradzki, Implicit LES using adaptive filtering, *Journal of Computational Physics* 359 (2018) 380-408.

[94] M. Thoma, How to check if two line segments intersect. https://martin-thoma.com/how-to-check-if-two-line-segments-intersect/, 2013 (accessed 13 March 2018).

[95] G.I. Taylor, A.E. Green, Mechanism of the production of small eddies from large ones, *in: Proceeding of the Royal Society of London A* 1937, 158 499-521.

[96] B. Thornber, A. Mosedale, D. Drikakis, On the implicit large eddy simulations of homogeneous decaying turbulence, *Journal of Computational Physics* 226 (2) (2007) 1902-1929.

[97] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, *Journal of Computational Physics* 209 (2) (2005) 448-476.

[98] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A Sharp Interface Cartesian Grid Method for Simulating Flows with Complex Moving Boundaries, *Journal of Computational Physics* 174 (2001) 345-380.

[99] C.H.K. Williamson, Oblique and parallel models of vortex shedding in the wake of a circular cylinder at low Reynolds numbers, *Journal of Fluid Mechanics* 206 (1989) 579–627.

[100] Z.J. Wang, Two dimensional mechanism for insect hovering, *Physical Review Letters* 85 (10) (2000) 2216-2219.

[101] P. Wesseling, C.W. Oosterlee, Geometric multigrid with applications to computational fluid dynamics, *Journal of Computational and Applied Mathematics* 128 (1-2) (2001) 311-334.

[102] F.M. White, *Fluid Mechanics: 7th Ed.*, McGraw-Hill, New York, 2011.

[103] F. Xiao, T. Yabe, Completely conservative and oscillationless semi-Lagrangian schemes for advection transportation, *Journal of Computational Physics* 170 (2) (2001) 498-522.

[104] D. Xiu, G.E. Karniadakis, A Semi-Lagrangian High-Order Method for Navier-Stokes Equations, *Journal of Computational Physics* 172 (2) (2001) 658-684.

[105] G. Yun, D. Kim, H. Choi, Vortical structures behind a sphere at subcritical Reynolds numbers, *Physics of Fluids* 187 (2003) 343-368.

[106] J. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, *Journal of Computational Physics* 215 (1) (2006) 12-40.

[107] J. Yang, F. Stern, A simple and efficient direct forcing immersed boundary framework for fluid-structure interactions, *Journal of Computational Physics* 231 (15) (2012) 5029-5061.

[108] S. Yoo, B.H. Jun, H. Shim, P. Lee, Design and analysis of carbon fiber reinforced plastic body frame for multi-legged subsea walking robot, Crabster, *Ocean Engineering* 102 (2015) 78-86.

[109] P. Bcaudan, P. Moin, *Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number*, Report No. TF-62, Department of Mechanical Engineering, Stanford University, 1994.

[110] L. Botto, A geometric multigrid Poisson solver for domains containing solid inclusions, *Computer Physics Communication* 184 (2013) 1033-1044.

[111] S. Dong, G.E. Karniadakis, DNS of flow past a stationary and oscillating cylinder at Re=10000, *Journal of Fluids and Structures* 20 (2005) 519-531.

[112] R. Gopalkrishnan, *Vortex-induced forces on oscillating bluff cylinders*, Ph.D. dissertation, MIT, Cambridge, 2005.

[113] H. Johansen, P. Colella, A Cartesian Grid Embedded Boundary Method for Poisson`s Equation on Irregular Domains, *Journal of Computational Physics* 147 (1998) 60-85.

[114] A.G. Kravchenko, P. Moin, Numerical studies of flow over a circular cylinder at $\mathrm{Re}_D = 3900$, *Physics of Fluids* 12 (2) (2000) 403-417.

[115] P. Parnaudeau, J. Carlier, D. Heitz, E. Lamballais, Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3900, *Physics of Fluids* 20 (2008) 085101.

[116] D. Weber, J. Mueller-Roemer, A. Stork, D. Fellner, A Cut-Cell Geometric Multigrid Poisson Solver for Fluid Simulation, *Computer Graphics Forum* 34 (2015) 481-491.

Abstract in Korean

본 논문에서는 직교격자 상에서 효율적인 복잡형상 주위 비압축성 유동해석을 위한 두 가지 알고리즘이 제시된다. 첫번째로, 불규칙한 도메인상에서 정의된 포아송 방정식에 쉽게 적용할 수 있는 다중격자법(Multigrid)이 제시된다. Projection 법을 이용한 유동해석에서 계산시간이 가장 많이 소요되는 부분은 압력 값을 구하기 위해 포아송 방정식을 푸는 과정이며, 이를 가장 빠르게 해석할 수 있는 방법은 다중격자법이다. 그러나 유체영역 내부에 물체가 포함될 경우 유체 영역은 불규칙 도메인이 되며, 연속적으로 형성된 성긴 격자시스템을 활용하는 다중격자법은 이러한 불규칙 도메인 문제에 적용되는데 많은 어려움이 있다. 본 논문에서는 물체의 형상 정보를 포함하는 헤비사이드 함수를 성긴 격자 시스템으로 직접 Restriction하여 불규칙 도메인 문제에 적용될 수 있는 다중격자법이 제시된다. 이 방법의 경우 실제 구현하기 쉽고, 병렬화 과정이 간단하며, 불규칙 도메인 문제에서도 다중격자법의 최적 성능이 보장된다. 두번째로, Adaptive Mesh Refinement(AMR)를 활용하여 물체 형상에 대한 Signed Distance Function (SDF)을 효율적으로 계산할 수 있는 기법이 제시된다. 직교격자상에서 물체 형상정보는 SDF를 통해 표현되며, 해석적인 형태의 SDF가 존재하지 않은 복잡한 형상의 경우 기하학적 연산을 통해 SDF를 직접 계산해야 된다. 이 과정은 수없이 많은 반복 연산을 수반하기 때문에 많은 계산시간이 요구된다. 본 논문에서는 AMR 기법을 통해 실제 정확한 SDF가 요구되는 영역만을 효과적으로 선별하여 전체적인 SDF 계산과정을 최적화할 수 있는 방법이 제시된다.

앞서 기술된 두가지 기법을 비압축성 유동해석 알고리즘과 결합하여, 복잡형상 주위 유동을 효율적으로 해석할 수 있는 자체코드(in-house code)를 개발하였다. 개발된 코드는 Hybrid MPI/OpenMP 기법으로 병렬화 되었으며, 병렬계산은 국가슈퍼컴퓨터 5호기 누리온에서 수행되었다. 본 논문에 제시된 알고리즘들의 타당성을 검증하기 위하여 2차원 및 3차원 공간상에서 정의된 다양한 벤치마크 문제를 해석하였으며, 그 결과를 이전 실험 및 수치 시뮬레이션 결과와 비교 및 검증하였다. 또한, 수중로봇 Crabster 및 골프공 주위 유동해석을 통해 현실적인 문제에 대한 적용 가능성을 검토해보았다.