Computational Methods in Engineering & the Sciences

Somnath Roy
Ashoke De
Elias Balaras *Editors*

Immersed Boundary Method

Development and Applications

Springer

Computational Methods in Engineering & the Sciences

Series Editor

Klaus-Jürgen Bathe, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA

This Series publishes books on all aspects of computational methods used in engineering and the sciences. With emphasis on simulation through mathematical modelling, the Series accepts high quality content books across different domains of engineering, materials, and other applied sciences. The Series publishes monographs, contributed volumes, professional books, and handbooks, spanning across cutting edge research as well as basics of professional practice. The topics of interest include the development and applications of computational simulations in the broad fields of Solid & Structural Mechanics, Fluid Dynamics, Heat Transfer, Electromagnetics, Multiphysics, Optimization, Stochastics with simulations in and for Structural Health Monitoring, Energy Systems, Aerospace Systems, Machines and Turbines. Climate Prediction, Effects of Earthquakes, Geotechnical Systems, Chemical and Biomolecular Systems, Molecular Biology, Nano and Microfluidics, Materials Science, Nanotechnology, Manufacturing and 3D printing, Artificial Intelligence, Internet-of-Things.

More information about this series at http://www.springer.com/series/16459

Somnath Roy · Ashoke De · Elias Balaras
Editors

Immersed Boundary Method

Development and Applications

Springer

Editors
Somnath Roy
Department of Mechanical Engineering
Indian Institute of Technology Kharagpur
Kharagpur, West Bengal, India

Ashoke De
Department of Aerospace Engineering
Indian Institute of Technology Kanpur
Kanpur, Uttar Pradesh, India

Elias Balaras
Department of Mechanical and Aerospace Engineering
George Washington University
Washington, DC, USA

ISSN 2662-4869 ISSN 2662-4877 (electronic)
Computational Methods in Engineering & the Sciences
ISBN 978-981-15-3939-8 ISBN 978-981-15-3940-4 (eBook)
https://doi.org/10.1007/978-981-15-3940-4

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

Over the past two decades, immersed boundary (IB) methods have been constantly gaining popularity and are increasingly expanding to new areas of applications in computational mechanics. A common feature of all IB formulations is that the requirement for the grid lines to conform to the boundary is relaxed. This greatly simplifies grid generation but at the same time renders the implementation of boundary conditions non-trivial. The no-slip boundary condition, for example, in most IB variants is approximated by assuming either elastic or viscoelastic properties on incompressible solids and deriving their equations of motion in a continuum setting or imposing kinematic constraints on the surrounding Eulerian points or the surface points themselves. The required forces to impose boundary conditions are computed directly from the momentum equations or can be obtained using Lagrange multipliers on the set of the equations governing the problem. An attractive feature of IB methods in complex geometries is that the need for the tedious grid generation step in boundary-conforming formulations is eliminated. This is particularly beneficial for flows over moving/deforming boundaries, which can now be tackled on structured grids utilizing highly efficient solvers with optimal conservation properties.

Historically, the IB formulation was designed to simulate the complex fluid–structure interaction (FSI) problem in the human heart by Prof. C. S. Peskin in early 1970s. In this pioneering implementation, the principle of virtual work in conservative systems for a continuous elastic material subjected to the incompressibility constraint was considered. The material was defined in curvilinear coordinates and assumed elastic, while the equations of motion were derived in Eulerian coordinates considering constrained virtual velocities. A limitation of this formulation is that it requires the definition of elastic properties for an immersed solid, which can be problematic when used in rigid body dynamical systems. Prof. Peskin's group proposed extension of the IB methods to rigid body systems in their later work at the beginning of 21st century. Several variances of this class of methods, broadly

known as continuous forcing schemes, have been proposed since. A new class of methods was proposed at the same time by research groups at Sapienza University of Rome, Polytechnic University of Bari and Los Alamos National Laboratory, where they considered directly the discretized momentum equations to introduce forcing such that the IB no-slip and no-penetration condition was approximated. Practically, the approach is equivalent to a local reconstruction of the velocity at points on the Eulerian grid near the boundary. Numerous variances of this formulation have been proposed in the literature since, including methods where the direct forcing function is computed on the Lagrangian grid defining the boundary and then transferred to the Eulerian grid nodes, sharing some features with the continuous forcing schemes.

Today, IB methods are considered a viable alternative to classical boundary-conforming formulations, especially in cases of moving/deforming boundaries. Depending on the application, however, there are still challenges to be addressed, which are unique to the particular class of methods. Some formulations, for example, trade the ease of implementation in existing structured solvers with a limited class of boundary conditions one can consider. In a similar manner, certain formulations are more appropriate for moving boundary, fluid–structure interaction problems, or facilitate an accurate computation of the local traction forces, which is non-trivial in IB methods. The plurality of IB formulations certainly drove the constant proliferation of the approach into new application areas, but at the same time it created a vast amount of the literature that one needs to consider when selecting a cost-efficient formulation for a particular class of problems. This book aims to be of the first of its kind to distill the vast information available and discuss different IB implementations and applications for a range of problems.

In particular, the reader will find a balanced distribution of chapters covering both incompressible and compressible flow implementations. For the former case, a number of chapters are devoted to addressing fundamental issues, such as projection methods, mass conservation, spurious pressure oscillations, modeling of the turbulent boundary layer, and FSI. Practical aspects related to computational efficiency and utilization of modern hybrid computing platforms are discussed in a separate chapter. Specific implementations for curvilinear mesh solvers and lattice Boltzmann methods are also included. For the case of compressible flows, the emphasis is placed on high-speed turbulent flows. The implementation of reconstruction functions that mimic the turbulent wall laws is discussed, together with strategies for the reconstruction of the temperature field, as well as sharp interface formulations. Finally, practical examples for a wide range of applications are included and cover areas as diverse as human locomotion, insect flight, and high-speed compressible flow around an aircraft.

We believe that this book is unique in nature and covers an extensive gamut of topics on IB methods and applications from a group of authors that are internationally recognized in their respective fields. We hope that this will be of help to graduate and undergraduate students, researchers, and managers in their quest to explore and utilize IB methods.

Kharagpur, India
Somnath Roy
somnath.roy@mech.iitkgp.ac.in

Kanpur, India
Ashoke De
ashoke@iitk.ac.in

Washington, DC, USA
Elias Balaras
balaras@gwu.edu

Contents

About the Editors

Somnath Roy is an Associate Professor at Indian Institute of Technology Kharagpur. Dr. Roy received his Masters' degree in Mechanical Engineering from IIT Kanpur in 2004, and Ph.D. degree in Mechanical Engineering from Louisiana State University, USA in 2010. Before joining IIT Kharagpur, he worked as a research associate and visiting Assistant Professor at Louisiana State University, USA and as an Assistant Professor at Indian Institute of Technology Kharagpur. His research interest is turbulence, arterial flows, moving boundary flow simulation, high performance computing using cluster and GPGPUs. His primary area of work is computational fluid dynamics (CFD). He has hosted two GIAN programs on computational methods and has also offered online course (NPTEL) on matrix solvers. His group works on developing immersed boundary method (IBM) based computationally efficient algorithms to solve moving boundary problems and on utilizing these implementations to predict flow and heat transfer in engineering and biological applications.

Ashoke De is currently working as Associate Professor in the Department of Aerospace Engineering at Indian Institute of Technology Kanpur. Dr. De received his Masters' degree in Aerospace Engineering from IIT Kanpur in 2004, and Ph.D. degree in Mechanical Engineering from Louisiana State University, USA in 2009. Before joining IIT Kanpur, he worked as a post-doctoral scholar at Technical University of Delft (TU-Delft), Netherlands and as Research Engineer in GE Global Research at Bangalore. He is the recipient of many awards and fellowships including the Humboldt Research fellowship (2018), DAAD Fellowship (2016) and GE Global Research's Expertise Award (2010), among others. Dr. De leads large-scale initiatives in the modeling of turbulent reacting and non-reacting flows at IIT Kanpur. His current research interests include combustion modeling, hybrid RANS/LES model development, supersonic flows and Fluid-Structure interactions (FSI). His primary research focus is the emerging field of computational mechanics with particular interest in combustion and turbulent flows.

Elias Balaras is a Professor at the Department of Mechanical and Aerospace Engineering at George Washington University. Prof. Balaras received his Ph.D. from the Swiss Federal Institute of Technology (EPFL) in Lausanne, Switzerland in 1995. He was formerly a visiting scientist at the National Institute for Standards and Technology and a faculty at the University of Maryland. Prof. Balaras's current research program aims at the development of robust numerical techniques for parallel, large-scale simulations of multiscale, multiphysics problems in physical and biological systems. Emphasis is given at large-eddy and direct numerical simulations, fluid-structure interactions and biological fluid dynamics. He has been the recipient of several awards including the Marie-Curie fellowship from the European Commission in 1994 and the career award from the National Science Foundation in 2003. He is currently an Associate Editor at the ASME J. Fluids Engineering and has served as a reviewer for numerous journals and government programs related to fluid mechanics, biological flows, high performance computing and turbulence.

Part I
Incompressible Flow Modeling

Chapter 1
Immersed Boundary Projection Methods

Benedikt Dorschner and Tim Colonius

1.1 Overview of the Immersed Boundary Method

Most conventional numerical methods to simulate complex fluid–structure interaction problems utilize body-conforming discretizations, where the fluid–solid interface conditions are imposed as boundary conditions. In its most general formulation, the fluid–structure interface can both be moving and deforming as a result of the two-way coupling between fluid and structure. Common body-fitted approaches include arbitrary Lagrangian–Eulerian formulations (Hirt et al. 1974; Ahn and Kallinderis 2006) or space–time finite element methods (Tezduyar et al. 1992, 2006).

The generation of body-fitted meshes for complex, possibly moving and deforming geometries, is computationally expensive and requires sophisticated procedures to avoid severe mesh distortion and preserve accuracy (Thompson et al. 1998; Hermansson and Hansbo 2003; Tezduyar et al. 2006; Nakata and Liu 2012). An alternative is the use of non-conforming meshes, the most widely used example being the immersed boundary method (IB). The IB method was first proposed in Peskin (1972) to simulate blood flow inside a heart with flexible valves. In the IB method, the flow field is described on a non-conforming Eulerian grid. The immersed surface is represented in a Lagrangian framework, and the surface traction is determined by imposing the no-slip boundary condition on the Eulerian velocity field interpolated to the surface. In the continuous setting, the surface traction is a singular function (defined only on the surface) and is discretized by a smeared, discrete delta function that regularizes the forcing effect over the neighboring Eulerian grid cells.

B. Dorschner · T. Colonius (✉)
Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125, USA
e-mail: colonius@caltech.edu

B. Dorschner
e-mail: bdorschn@caltech.edu

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_1

In the original IB method (Peskin 1972), the heart valves were modeled as flexible membranes, and Hooke's law was used as a constitutive relation to relate the forcing function to the motion of the Lagrangian points. Later, this scheme was extended to rigid bodies by taking large values for the spring constants (Beyer and LeVeque 1992; Lai and Peskin 2000). In addition, the concept of feedback control to compute the force on the rigid immersed surface was introduced by Goldstein et al. (1993), where the difference between the velocity solution and the boundary velocity is used in a proportional-integral controller. Note that for techniques using constitutive relations to model the flow over rigid bodies, the choice of gain (stiffness) is a tuning parameter whose value must be heuristically chosen to simultaneously avoid a restrictive time step size (large stiffness) and slip error (small stiffness).

Constitutive relations are eliminated in direct forcing methods and its variants (Mohd-Yusof 1997; Fadlun et al. 2000), where the momentum forcing is obtained by penalization of the slip at the surface. However, the no-slip condition is only enforced on an intermediate velocity field and hence requires iterations to approximate the no-slip condition on the final velocity fields. While the slip has been reported to be small (Fadlun et al. 2000), it cannot be estimated in a systematic fashion.

An alternative approach is to consider the boundary force as a Lagrange multiplier that is determined to enforce the no-slip condition (Glowinski et al. 1998; Taira and Colonius 2007; Colonius and Taira 2008; Kallemov et al. 2016). In this formulation, the discretized, incompressible Navier–Stokes equations (NSE) can be formulated in an analogous manner to the classical fractional step method by introducing appropriate regularization and interpolation operators. In addition, a modified Poisson equation, where the force and the pressure are lumped together, can be solved to determine the pressure and force unknowns. We refer to these methods as immersed boundary projection methods (IBPM). The advantage of IBPM is that continuity and no-slip conditions can be satisfied implicitly and with arbitrary accuracy at each time step. The Courant number is further only limited by the choice of the time-marching algorithm. With typical splitting methods (fractional step methods), one can achieve second-order accuracy uniformly in time and the matrices arising from the implicit treatment of the viscous terms as well as the modified Poisson equation can be made symmetric and positive definite. The resulting linear system can be solved with an efficient conjugate-gradient solver.

While the standard implementation with discrete delta functions is only first-order accurate in space, there have been efforts to improve the accuracy of IB methods in order to efficiently tackle higher Reynolds number flows. These include so-called sharp-interface and cut-cell approaches (Seo and Mittal 2011). In particular, standard IBs such as ghost-cell methods do not in general conserve mass or momentum at the interface, which manifests in spurious pressure oscillations (Mittal and Iaccarino 2005) and becomes particularly problematic for compressible flows or when coupled with large-eddy simulations and alike. A remedy was found in cut-cell methods, which strictly enforce conservation by reshaping finite volume boundary cells to locally conform with the geometry. A drawback of the cut-cell approach is that the fluid volume fractions of cut cells can become small, necessitating stabilization of the underlying time-stepping scheme using cell-merging (Ye et al. 1999), cell-linking

(Kirkpatrick et al. 2003) or flux redistribution techniques (Hu et al. 2006; Colella et al. 2006). Additionally, cut-cell methods inherit many of the complications of body-fitted meshes and must be adapted dynamically for fluid–structure interaction problems.

In discussing order of accuracy, we must tackle a misunderstanding that has permeated at least through part of the IB literature. Even for rigid bodies, the velocity gradients are not continuous at an immersed surface, and the discretization must account for the derivative singularity to achieve high-order accuracy. If the singular traction is regularized without respect to the discretization (for example by using a discrete delta function), the regularized solution will converge to the continuous one at first order irrespective of the order of accuracy of the schemes used to treat the derivative operators. The regularization error may in principle be made small independently of the discretization error, but then the region over which the surface is smeared must be made arbitrarily thin *compared to the grid spacing*. If the first-order regularization error does not satisfy this restriction, then these "high-order" methods simply converge to the incorrect, smeared solution faster. An interesting approach toward higher-order IB methods is presented in Stein et al. (2017).

In this review, we take the alternative approach of accepting the first-order error near the immersed surface, as this allows mimetic discretizations that achieve other desirable properties, such as stability, discrete conservation, and computational efficiency. It may be possible to achieve second-order accuracy while maintaining these other characteristics in the future.

In what follows, we will introduce the immersed boundary projection method as proposed in Taira and Colonius (2007), Colonius and Taira (2008). We will also present a strongly coupled fluid–structure interaction algorithm as in Goza and Colonius (2017), which is then applied to simulate the flow past an inverted flexible flag. Subsequently, an immersed boundary method based on Lattice Green's functions is introduced, and examples ranging from inclined rotating disks to turbulent flow past a sphere are shown. Finally, some perspectives to further increase the Reynolds number and inclusion of explicit turbulence models within the filtered NSE are provided.

1.2 Immersed Boundary Projection Method

The IB formulation for the incompressible Navier–Stokes equations with an additional singular boundary force $\boldsymbol{f}_\Gamma$ in the momentum equation reads as :

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \int_{\Gamma(t)} \boldsymbol{f}_\Gamma(\boldsymbol{X}(\boldsymbol{\xi}, t), t) \delta(\boldsymbol{X}(\boldsymbol{\xi}, t) - \boldsymbol{x}) \mathrm{d}\boldsymbol{\xi}, \quad (1.1a)$$

$$\nabla \cdot \boldsymbol{u} = 0, \quad (1.1b)$$

$$\int_\Omega \boldsymbol{u}(\boldsymbol{x}) \delta(\boldsymbol{x} - \boldsymbol{X}(\boldsymbol{\xi}, t)) \mathrm{d}\boldsymbol{x} = u_\Gamma(\boldsymbol{\xi}, t) = \frac{\partial \boldsymbol{X}(\boldsymbol{\xi}, t)}{\partial t}, \quad (1.1c)$$

where $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t)$, $p(\boldsymbol{x}, t)$, and Re denote the fluid velocity, pressure, and the Reynolds number, respectively. We assume that the equations have already been made non-dimensional with respect to characteristic length and velocity scales, and the (constant) fluid density.

The Dirac delta function is indicated by δ and Ω is the Eulerian domain. The immersed body surface Γ is described in Lagrangian coordinates $\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{\xi}, t)$, where $\boldsymbol{\xi}$ is the surface parametrization, and $u_\Gamma(\boldsymbol{\xi}, t) = \frac{\partial \boldsymbol{X}(\boldsymbol{\xi}, t)}{\partial t}$ is the boundary velocity. The body force $\boldsymbol{f}_\Gamma$ is chosen such that the no-slip condition on the immersed surface, as prescribed by Eq. (1.1c), is satisfied. This step is agnostic to any model for the motion of the surface based on the fluid forces acting on it, which in general determines the *position* of the surface, which can be moving with respect to the underling Eulerian domain.

The convolution with the Dirac delta function Eq. (1.1a) and Eq. (1.1c) couples the immersed surface with the Eulerian grid Ω. The velocity field $\boldsymbol{u}$ as well as the pressure are defined for all $\boldsymbol{x} \in \Omega$ and satisfy the far field boundary conditions $\boldsymbol{u}(\boldsymbol{x}, t) \to \boldsymbol{u}_\infty(t)$ as $|\boldsymbol{x}| \to \infty$.

An issue with this formulation based on surfaces is that it is assumed that fluid resides on either side of the immersed surface. When this is not the case, i.e., the immersed surface comprises a substantial closed volume, there are wasted points in the "ghost fluid" inside the body. Moreover, it is important to remember that the fluid inside the body can exert a force on the body when the surface is accelerated. For example, the added mass on a fluid-filled hollow sphere is different from that of a solid sphere. To the extent that the fluid inside the surface is moving as a rigid body, the additional force can readily be tabulated and used to correct the IB results to non-hollow bodies. For the more general case involving deformation of the IB, this formulation is only directly applicable to thin structures.

1.2.1 Discretization

The spatial discretization may be obtained via a second-order mimetic finite volume method (Nicolaides and Wu 1997; Nicolaides 1992; Perot 2000; Zhang et al. 2002) on a staggered mesh $Q := \{\mathcal{V}, \mathcal{E}, \mathcal{F}, C\}$, which consists of vertices $\mathcal{V}$, edges $\mathcal{E}$, faces $\mathcal{F}$, and cells C (see Fig. 1.1). Scalar quantities reside at cell centers and vertices, while faces and edges contain vector flow quantities. Grid functions with values on Q are denoted by $\mathbb{R}^Q$ and functions with vector values at the Lagrangian grid points are denoted by $\mathbb{R}^\Gamma$. The semi-discrete form of Eq. (1.1) reads as

$$M\frac{\mathrm{d}u}{\mathrm{d}t} + N(u, t) = -Gp + \frac{1}{\mathrm{Re}}L_{\mathcal{F}}u + R(t)f \tag{1.2a}$$

$$\bar{D}u = 0 \tag{1.2b}$$

$$R(t)u = u_\Gamma, \tag{1.2c}$$

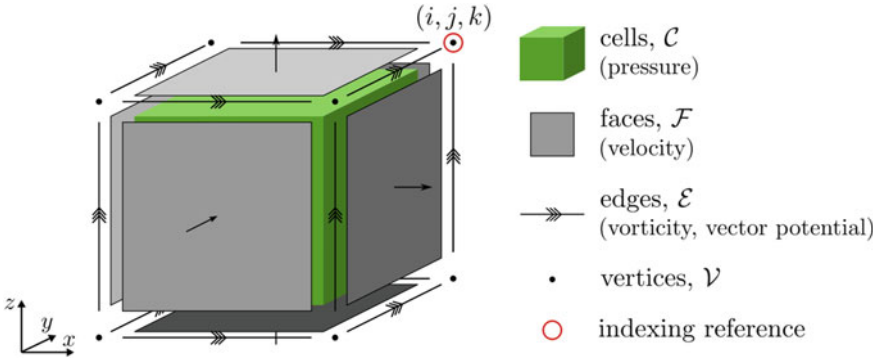


Fig. 1.1 Staggered cell object as used in the IBLGF method. Reprinted from Liska and Colonius (2016) with permission

where $u \in \mathbb{R}^{\mathcal{F}}$ and $p \in \mathbb{R}^{C}$ are the discrete velocity and pressure variables at time $t > 0$. The mass matrix is denoted by M. The discrete gradient, divergence, and Laplace operators are denoted by G, D, and L and, if ambiguous, a subscript is used to identify storage location. The set of discrete vector operators used in the following is given by:

$$\text{Gradient} \qquad G : \mathbb{R}^{\mathcal{C}} \to \mathbb{R}^{\mathcal{F}}, \bar{G} : \mathbb{R}^{\mathcal{C}} \to \mathbb{R}^{\mathcal{F}}, \tag{1.3}$$

$$\text{Curl} \qquad C : \mathbb{R}^{\mathcal{F}} \to \mathbb{R}^{\mathcal{E}}, \bar{C} : \mathbb{R}^{\mathcal{E}} \to \mathbb{R}^{\mathcal{F}}. \tag{1.4}$$

$$\text{Divergence} \qquad D : \mathbb{R}^{\mathcal{E}} \to \mathbb{R}^{\mathcal{V}}, \bar{D} : \mathbb{R}^{\mathcal{F}} \to \mathbb{R}^{\mathcal{C}}, \tag{1.5}$$

$$\text{Laplace} \qquad L : \mathbb{R}^{Q} \to \mathbb{R}^{Q}. \tag{1.6}$$

The convection term $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ is approximated by the nonlinear operator $N(u, t)$. Different choices for this discretization lead to different (conservation) properties (Perot 2000). The discrete surface functions $f(i, t)$ and $u_\Gamma(i, t)$ denote the force and the velocity of the ith Lagrangian marker at $\boldsymbol{X}(\boldsymbol{\xi_i}, \boldsymbol{t})$ where $i \in [1, N_L]$. The interpolation and regularization operators $E(t)$ and $R(t)$ are time-dependent and constructed by regularizing the δ-function convolutions of Eq. (1.1a) and Eq. (1.1c), i.e., $R(\cdot)$ and $E(\cdot)$ are discretizations of $\int_\Gamma (\cdot)\delta_h(X(\boldsymbol{\xi}, t) - \boldsymbol{x})\mathrm{d}\boldsymbol{\xi}$ and $\int_\Omega (\cdot)\delta_h(X(\boldsymbol{\xi}, t) - \boldsymbol{x})\mathrm{d}\boldsymbol{x}$, respectively (see also Sect. 1.2.3). The interpolation and regularization operators are adjoints under the standard inner product such that $E = (\Delta x)^3 R^\dagger$.

The scheme is second-order accurate and by using a staggered Cartesian grid conserves momentum and either kinetic energy or circulation, depending on the discretization for $N(u, t)$, in the limit of vanishing viscosity and time-stepping errors (Lilly 1965; Morinishi et al. 1998; Perot 2000). Note that the operators G and D can be formulated such that $G = -D^\dagger$. Explicit expression for all operators can be found in Liska and Colonius (2017), Colonius and Taira (2008). For a uniform mesh, the mass matrix M is a constant multiple of the identity.

With these definitions, we can write Eq. (1.2) as a system of algebraic equations as

$$\begin{bmatrix} A & G & -R \\ D & 0 & 0 \\ E & 0 & 0 \end{bmatrix} \begin{bmatrix} u^{n+1} \\ p \\ f \end{bmatrix} = \begin{bmatrix} r^n \\ 0 \\ u_\Gamma^{n+1} \end{bmatrix} + \begin{bmatrix} \mathrm{bc}_1 \\ \mathrm{bc}_2 \\ 0 \end{bmatrix}, \tag{1.7}$$

where the submatrix A is the result of the implicit velocity treatment. Here, it is obtained by the implicit trapezoidal rule on the viscous term yielding $A = \frac{1}{\Delta t} M - \frac{1}{2} L$. The convection term is discretized by the second-order Adams–Bashforth method, leading to the right-hand side $r^n = [\frac{1}{\Delta t} M - \frac{1}{2} L] u^n + \frac{3}{2} N(u^n) - \frac{1}{2} N(u^{n-1})$. The inhomogeneous terms bc_1, bc_2 depend on the particular boundary conditions, which are discussed later. Using the above properties of the submatrices, Eq. (1.7) can be rewritten as

$$\begin{bmatrix} A & G & E^\dagger \\ G^\dagger & 0 & 0 \\ E & 0 & 0 \end{bmatrix} \begin{bmatrix} u^{n+1} \\ p \\ \tilde{f} \end{bmatrix} = \begin{bmatrix} r^n + \mathrm{bc}_1 \\ -\mathrm{bc}_2 \\ u_\Gamma^{n+1} \end{bmatrix}, \tag{1.8}$$

where $\tilde{f}$ is the scaled boundary force, which accounts for the scaling factor when expressing R with $E^\dagger$. The form of Eq. (1.8) is the Karush–Kahn–Tucker (KKT) system, where $(p, \tilde{f})$ is the set of Lagrange multipliers to satisfy a set of kinematic constraints. These constraints are purely algebraic, and there is no need for the pressure and boundary force to be distinguished anymore. Thus, we can group the Lagrange multipliers and the submatrices as $\lambda = [p, \tilde{f}]$ and $Q = [G, E^\dagger]$. The above system is algebraically identical to traditional discretizations of the NSE and allows the use of standard solvers. Here, the (projection) fractional step algorithm is applied to Eq. (1.8), which can be expressed as an approximate LU decomposition of the left side matrix (Perot 2000), which yields the immersed boundary projection method (IBPM):

$$Au^* = r_1, \quad \text{(Solve for intermediate velocity)} \tag{1.9}$$

$$Q^\dagger A_j^\ddagger Q\lambda = Q^\dagger u^* - r_2, \quad \text{(Solve modified Poisson equation)} \tag{1.10}$$

$$u^{n+1} = u^* - A_j^\ddagger Q\lambda \quad \text{(Projection step)}, \tag{1.11}$$

where $A_j^\ddagger$ is the jth order Taylor series expansion of A^{-1} with respect to Δt, and the explicit terms on the right-hand side are denoted by r_1 and r_2. Note that in Taira and Colonius (2007), A and $Q^\dagger A_j^\ddagger Q$ are constructed to be symmetric positive-definite operators such that the system can be solved efficiently with the conjugate-gradient method. Thus, the no-slip boundary condition is enforced on the solution by projecting the intermediate velocity field into the solution space that satisfies both divergence-free and no-slip constraints.

The IBPM is found to be second-order in time and first-order accurate in space.[1] There is no need for a constitutive relation to compute the boundary force. The IBPM therefore does not have any stability restrictions associated with the immersed surface, and the time step restrictions are imposed only by the choice of the marching scheme.

1.2.2 Nullspace Method for the Immersed Boundary Method

The nullspace or discrete streamfunction approach was originally proposed for solving Eq. (1.2) without the immersed boundary (Hall 1985; Chang et al. 2002), where only the incompressibility constraint needs to be satisfied. Using the discrete streamfunction s such that

$$u = Cs, \tag{1.12}$$

where the discrete curl operator C is constructed with column vectors corresponding to the basis of the nullspace of D. It follows that

$$DC = 0, \tag{1.13}$$

which automatically satisfies the incompressibility constraint for all times.

In addition, left-multiplication of the momentum equation with $C^\dagger$ removes the pressure term and thus reduces to a single equation to be solved per time step

$$C^\dagger ACs^{n+1} = C^\dagger(r_1^n + \mathrm{bc}_1). \tag{1.14}$$

Note that solution of the pressure Poisson equation is not required here and therefore the most expensive part of the fractional step method is eliminated, while exactly satisfying the continuity equation. In addition, the errors from the approximate LU decomposition are eliminated, which is why this scheme is also called the *exact fractional step method* (Chang et al. 2002).

Furthermore, a second-order approximation of the circulation is obtained by $\gamma = C^\dagger q$.

Especially in two-dimensional problems, where the streamfunction and vorticity have a single nonzero component, this can lead to a more efficient algorithm, even in the presence of an IB. For 2D problems, including the immersed boundary formalism into the nullspace approach leads to the KKT system (Colonius and Taira 2008)

$$\begin{bmatrix} C^\dagger AC & C^\dagger E^\dagger \\ EC & 0 \end{bmatrix} \begin{bmatrix} s^{n+1} \\ \tilde{f} \end{bmatrix} = \begin{bmatrix} C^\dagger r_1^n \\ u_\Gamma^{n+1} \end{bmatrix}. \tag{1.15}$$

[1] Typically, the first-order errors that are associated with the regularization of the delta functions are limited to a finite region near the surface, and this results in better than first-order accuracy in the L_2 norm.

The left-hand side matrix is symmetric but in general indefinite, which limits the efficiency for direct solutions. However, with the projection (fractional step) approach, we obtain

$$C^\dagger ACs^* = C^\dagger R_\mathrm{I}^n \tag{1.16}$$

$$EC(C^\dagger AC)^{-1}(EC)^\dagger \tilde{f} = ECs^* - u_\Gamma^{n+1} \tag{1.17}$$

$$s^{n+1} = s^* - (C^\dagger AC)^{-1}(EC)^\dagger \tilde{f}. \tag{1.18}$$

A direct solution of the above system requires a nested iteration to solve the modified Poisson equation. For stationary bodies, however, one can compute a Cholesky factorization of $EC(C^\dagger AC)^{-1}(EC)^\dagger$ once, since the system size scales with the number of the immersed boundary points. Then, a system of equations of the form $C^\dagger ACx = b$ needs to be solved only once per Lagrangian force.

In Colonius and Taira (2008), it was shown that for a uniform grid with simple boundary conditions, a similar system to Eq. (1.9) can be solved efficiently using fast sine transforms. Assuming that the velocity outside the computational domain is known, simple Dirichlet boundary conditions can be applied to the velocity normal to the sides of the domain, while Neumann boundary conditions are imposed on the velocity tangent to the sides.

No-penetration boundary conditions for the normal component of the velocity and a zero vorticity (or no-stress) condition for the tangent components are natural boundary conditions for external flows, given a sufficiently large domain. With these simplifications, one can write the semi-discrete momentum equation as

$$\frac{\mathrm{d}\gamma}{\mathrm{d}t} = C^\dagger E^\dagger \tilde{f} = -\beta C^\dagger C\gamma + C^\dagger N(u) + \mathrm{bc} + \gamma, \tag{1.19}$$

where $Lq = -\beta CC^\dagger u = -\beta C\gamma$ has been used. Here, $\beta = 1/\Delta x^2 \mathrm{Re}$ is constant. Under the aforementioned assumptions, the matrix $-\beta C^\dagger C$ corresponds to the standard discrete Laplace operator with zero Dirichlet boundary conditions for γ. This discrete Laplacian can be diagonalized by a sine transform, where the sine transform pair is denoted by

$$\hat{\gamma} = S\gamma \leftrightarrow \gamma = S\hat{\gamma}, \tag{1.20}$$

and $\hat{(\cdot)}$ indicates Fourier coefficients. In addition, we use $\Lambda = SC^\dagger CS$, where Λ is a diagonal matrix with the eigenvalues of $C^\dagger C$, which are positive and known analytically.

Using the same time-marching scheme as above, the system becomes

$$S\left(I + \frac{\beta\Delta t}{2}\Lambda\right)S\gamma^* = \left(I - \frac{\beta\Delta t}{2}C^\dagger C\right)\gamma^n + \frac{\Delta t}{2}\left(3C^\dagger N(u^n) - C^\dagger N(u^{n-1})\right) + \Delta t\mathrm{bc}_\gamma, \tag{1.21}$$

$$EC\left(S\Lambda^{-1}\left(I+\frac{\beta\Delta t}{2}\Lambda\right)^{-1}S\right)(EC)^{\dagger}\tilde{f}=ECS\Lambda^{-1}S\gamma^{*}-u_{\Gamma}^{n+1}, \tag{1.22}$$

$$\gamma^{n+1}=\gamma^{*}-S\left(I+\frac{\beta\Delta t}{2}\Lambda\right)^{-1}S(EC)^{\dagger}\tilde{f}. \tag{1.23}$$

The velocity u^n can then be found by

$$u^n = Cs^n + \mathrm{bc}_u, \quad s^n = S\Lambda^{-1}S\gamma^n = \mathrm{bc}_s, \tag{1.24}$$

where each of the boundary conditions involve the assumed known values at the velocity edge.

Note that in the transformed system only one linear system associated with a symmetric positive-definite operator, Eq. (1.22), needs to be solved. In addition, the matrix dimensions are now $N_f \times N_f$, a drastic reduction compared to the original modified Poisson equation Eq. (1.10). A corresponding order of magnitude speedup was measured numerically in Colonius and Taira (2008). Further, if the body is stationary, the modified Poisson equation for the force can be solved efficiently using a triangular Cholesky decomposition.

To conclude, for a uniform grid and simple boundary conditions it is preferable to solve Eqs. (1.21)–(1.23). However, for simulations of external flows the simplified boundary conditions require large computational domains. Since the grid is also required to be uniform, this constraint quickly outweighs its benefits. However, the multi-domain approach as proposed in Colonius and Taira (2008) was found to be an effective solution to approximate the free-space boundary conditions. In Sect. 1.3, an IB method based on lattice Green's function is presented, which alleviates the need of far-field approximation and satisfies the free-space boundaries exactly.

1.2.3 Accurate Calculation of Surface Stresses and Forces

In this section, we will present a procedure to accurately calculate surface stresses and forces in the context of IB methods as proposed in Goza et al. (2016). In particular, we will focus on the set of IB methods that solve for surface stresses by imposing velocity boundary conditions. This is, for example, the case in the IBPM above, but in contrast to the original IB method of Peskin (1972), where the surface stresses are derived from specific constitutive laws. Velocity-based IB methods have been shown to suffer from inaccurate surface stresses and may also exhibit spurious oscillations in time traces, which originate from the ill-posedness of the first-kind integral equation for the surface stresses. It is important to note that the velocity is typically convergent in spite of the poor accuracy of the surface stresses. Thus, the following procedure is important for either post-processing the stress data or when the IB method is used in conjunction with a structural solver (see Sect. 1.2.4).

To understand the origin of the spurious oscillations, we consider the Poisson equation in two dimensions with an unknown singular source term f that takes nonzero values only on the immersed surface Γ as a model problem:

$$\begin{aligned} \nabla^2\varphi(\boldsymbol{x}) &= -\int_\Gamma f(\boldsymbol{X}(\boldsymbol{\xi}))\delta(\boldsymbol{x}-\boldsymbol{X}(\boldsymbol{\xi}))\mathrm{d}\xi, \\ \varphi(\boldsymbol{x}) &= \varphi_{\partial\Omega}(\boldsymbol{x}),\ \ \boldsymbol{x}\in\partial\Omega, \\ \int_\Omega \varphi(\boldsymbol{x})\delta(\boldsymbol{x}-\boldsymbol{X}(\boldsymbol{\xi}))\mathrm{d}\boldsymbol{x} &= \varphi_\Gamma(\boldsymbol{X}(\boldsymbol{\xi})). \end{aligned} \tag{1.25}$$

While numerical solutions of this equation and their errors have been analyzed for prescribed source terms f (Tornberg and Engquist 2004; Zahedi and Tornberg 2010), it is explicitly solved for in what follows by incorporating the third equation as a boundary constraint in order to mimic velocity-based IB method as closely as possible.

The immersed boundary Γ is taken to be a circle of radius $1/2$ in a unit square and is centered at $\boldsymbol{x}=0$,$\varphi_{\partial\Omega}(\boldsymbol{x})=1-\frac{1}{2}\log(2|\boldsymbol{x}|)$, and $\varphi_\Gamma(\boldsymbol{X})=1$. The exact solution to (1.25) is given by

$$\varphi_{\mathrm{ex}}(\boldsymbol{x}) = \begin{cases} 1 & |\boldsymbol{x}|\le\frac{1}{2}, \\ 1-\frac{1}{2}\log(2|\boldsymbol{x}|) & |\boldsymbol{x}|>\frac{1}{2}, \end{cases} \tag{1.26}$$

$$f_{\mathrm{ex}}(\boldsymbol{X}) = 1. \tag{1.27}$$

Similar to the integrated surface force in the context of IB, we also define $F_{\mathrm{ex}}=\int_\Gamma f_{\mathrm{ex}}(\boldsymbol{X}(\boldsymbol{\xi}))\mathrm{d}\boldsymbol{\xi}=\pi$. As was done in previous sections, the delta function is replaced with a smeared delta function, $\delta_h(\boldsymbol{x}-\boldsymbol{X}(\boldsymbol{\xi}))$, which is continuous with nonzero but compact support and is defined in terms of the grid spacing Δx. The numerical solution for a given grid spacing approximates

$$\varphi(\boldsymbol{x}) = -\int_\Omega\int_\Gamma f(\boldsymbol{X}(\boldsymbol{\xi}'))\delta_h(\boldsymbol{x}'-\boldsymbol{X}(\boldsymbol{\xi}'))G^L(\boldsymbol{x};\boldsymbol{x}')\mathrm{d}\boldsymbol{\xi}'\mathrm{d}\boldsymbol{x}', \tag{1.28}$$

where $G^L(\boldsymbol{x};\boldsymbol{x}')$ denotes the Green's function of the Poisson problem, and δ_h indicates the smeared delta function (see also Sect. 1.3 for further details). To obtain the unknown source term f, Eq. (1.28) is multiplied by δ_h and integrated over the domain Ω:

$$\int_\Omega\int_\Omega\int_\Gamma f(\boldsymbol{X}(\boldsymbol{\xi}'))\delta_h(\boldsymbol{x}'-\boldsymbol{X}(\boldsymbol{\xi}'))G^L(\boldsymbol{x};\boldsymbol{x}')\delta_h(\boldsymbol{x}-\boldsymbol{X}(\boldsymbol{\xi}))\mathrm{d}\boldsymbol{\xi}'\mathrm{d}\boldsymbol{x}'d\boldsymbol{x} = -\varphi_\Gamma(\boldsymbol{X}(\boldsymbol{\xi})), \tag{1.29}$$

and $\varphi(\boldsymbol{x})$ may be obtain upon substitution f into Eq. (1.28). Note that since δ_h is continuous, the kernel in the integral equation Eq. (1.29) is continuous and has finite

support, which makes the integral operator compact and with a formally unbounded inverse (Kress 2014). A direct consequence is that discretizations of this equation lead to inaccurate surface source terms.

Examples of smeared delta functions include

- A 2-point hat function:

$$\delta_h^{\text{hat}}(r) = \begin{cases} \frac{1}{\Delta x} - \frac{|r|}{\Delta x^2}, & |r| \le \Delta x \\ 0, & |r| > \Delta x \end{cases} \tag{1.30}$$

- A 3-point function:

$$\delta_h^3(r) = \begin{cases} \frac{1}{3\Delta x}\left(1 + \sqrt{1 - 3\left(\frac{r}{\Delta x}\right)^2}\right), & |r| \le \frac{\Delta x}{2} \\ \frac{1}{6\Delta x}\left(5 - \frac{3|r|}{\Delta x} - \sqrt{1 - 3\left(1 - \frac{|r|}{\Delta x}\right)^2}\right), & \frac{\Delta x}{2} \le |r| \le \frac{3\Delta x}{2} \\ 0, & |r| > \frac{3\Delta x}{2} \end{cases} \tag{1.31}$$

- A 4-point cosine function:

$$\delta_h^{\cos}(r) = \begin{cases} \frac{1}{4\Delta x}\left(1 + \cos\left(\frac{\pi r}{2\Delta x}\right)\right), & |r| \le 2\Delta x \\ 0, & |r| > 2\Delta x \end{cases} \tag{1.32}$$

- A Gaussian function:

$$\delta_h^G(r) = \begin{cases} \sqrt{\frac{\pi}{36\Delta x^2}}\, e^{\frac{-\pi^2 r^2}{36\Delta x^2}}, & |r| \le 14\Delta x \\ 0, & |r| > 14\Delta x \end{cases} \tag{1.33}$$

Discretization of Eq. (1.25) yields

$$L\varphi = -Rf + b_L, \tag{1.34}$$

$$E\varphi = \varphi_\Gamma, \tag{1.35}$$

and combining Eq. (1.34) and Eq. 1.35 results in

$$EL^{-1}Rf = -\varphi_\Gamma + EL^{-1}b_L, \tag{1.36}$$

which is a discretization of the integral equation (1.29).

In Goza et al. (2016), Eq. (1.36) was solved numerically using a finite difference approximation. In the following, n_b and n_g are used to denote the number of points on the immersed body and the computational domain, respectively. In Fig. 1.2, it is apparent that the source term f does not converge with grid refinement, whereas the integrated source term F and the solution φ do converge at first order to their

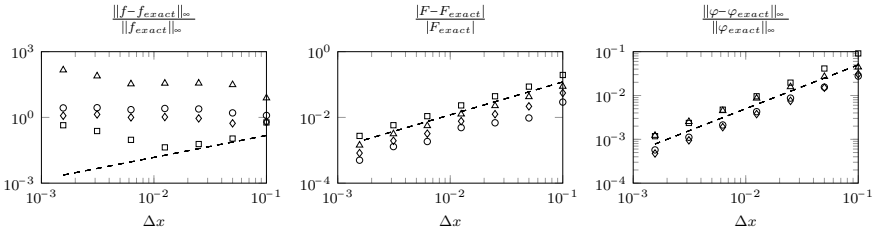


Fig. 1.2 Errors in $\tilde{f}$, F, and φ versus grid spacing (h) for the Poisson problem. $\circ$: δ_h^{hat}, $\diamond$: δ_h^3, $\triangle$: $\delta_h^{\cos}$, $\square$: δ_h^G, --: first-order convergence. Reprinted from Goza et al. (2016) with permission

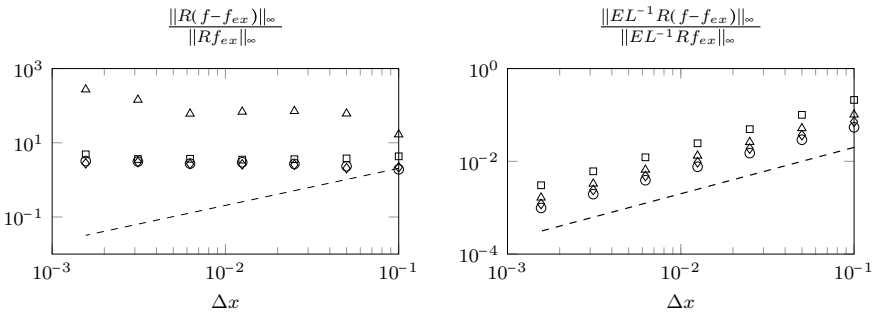


Fig. 1.3 Errors in Rf and $EL^{-1}Rf$ versus grid spacing (h) for the Poisson model problem. $\circ$:δ_h^{hat}, $\diamond$: δ_h^3, $\triangle$: $\delta_h^{\cos}$, $\square$: δ_h^G, --: first-order convergence. Reprinted from Goza et al. (2016) with permission

exact solutions. Convergence of F is a consequence of solving Eq. (1.36). This is in contrast to other velocity-based IB methods, which only approximately enforce the boundary constraint. Such methods introduce inaccuracies in F (Uhlmann 2005; Huang and Sung 2009; Zhang and Zheng 2007), although Yang et al. (2009) proposed improvements.

From Fig. 1.3, it is also apparent that Rf does not converge to Rf_{ex} but $EL^{-1}Rf$ converges to $EL^{-1}Rf_{\text{ex}}$. Thus using the exact force f_{ex} to enforce the boundary condition, would not produce φ_Γ exactly, but rather converge to it at first order (see also Tornberg and Engquist 2004).

In Goza et al. (2016), the convergence behavior was studied further using a singular value decomposition (SVD) of $EL^{-1} = U\Sigma V^\dagger$. Using this decomposition, Rf_{ex} may be written as a projection onto the basis of vectors formed by V:

$$Rf_{\text{ex}} = \sum_{j=1}^{n_b} \alpha_j^{\text{ex}} v_j. \tag{1.37}$$

Similarly, $EL^{-1}Rf_{\text{ex}}$ may be expressed as

$$EL^{-1}Rf_{\text{ex}} = \sum_{j=1}^{n_b} \alpha_j^{\text{ex}} \sigma_j u_j \tag{1.38}$$

where $\sigma_1, \ldots, \sigma_{n_b}$ are the singular values, and the left (right) singular vectors are denoted by u_j (v_j) corresponding to σ_j. The coefficients are defined as $\alpha_j^{\text{ex}} := (v_j^T Rf_{\text{ex}})$.

Analogous expressions can be written for Rf by replacing f_{ex} with f in (1.37) and (1.38). It was shown in Goza et al. (2016) that the sum $\sum_{j=1}^{n_b} \alpha_j$ does not converge to $\sum_{j=1}^{n_b} \alpha_j^{\text{ex}}$ under grid refinement, but converges when scaled by the σ_j. Since EL^{-1} is a discrete integral operator, the σ_j decay to small values (Hansen 1998) and the error is thus caused by high-index coefficients α_j corresponding to small σ_j.

Hence, smeared delta functions with rapidly decaying α_{ex} are favorable as spurious high-index coefficients can be filtered out effectively without loss of physical information. In contrast, delta functions with slow decay may lead to loss of physical information and thus inaccurate source terms due to inaccurate high-index coefficients.

An efficient filtering can be achieved by penalizing the spurious components of f. This can be done by pre-multiplying the source term with $\hat{E}R$ using a weighted interpolant $\hat{E} = EW$, which interpolates the smeared source term Rf onto the immersed body while preserving its integral value. The filtered source term is then $\hat{f} = \hat{E}Rf$.

In particular, W can be defined by a diagonal matrix with entries given by

$$W_{ii} = \begin{cases} 1/(R\mathbf{1})_i, & (R\mathbf{1})_i \neq 0 \\ 0, & \text{else,} \end{cases} \tag{1.39}$$

where $\mathbf{1} = [1, 1, \ldots, 1]^T \in \mathbb{R}^{n_g \times 1}$ and $(R\mathbf{1})_i$ is the ith entry in the vector $R\mathbf{1}$. Note that the weights are only nonzero within the support of the smeared delta function. The source term is redistributed by the filter $\hat{E}R$ by convolution with a kernel of smeared delta functions.

The rate of filtering of $\hat{E}R$ is proportional to the smoothness of the smeared delta function. This is a consequence of $\hat{E}R$ being an integral operator, for which the decay rate of its singular values is determined by the smoothness of its kernel (Hansen 1998). Applying this filtering technique yields more accurate source terms f as shown in Fig. 1.4. Indeed, the infinitely differentiable δ_h^G shows first-order convergence to f_{ex}, whereas the slow decay of the coefficients α_j hinders convergence for δ_h^{hat}, δ_h^3, and $\delta_h^{\cos}$. Finally, it is worth mentioning that filtering does not affect F by construction of $\hat{E}R$, and the solution φ is also unchanged since filtering is a post-processing step.

The extension of this filtering technique to the Navier–Stokes equation is straightforward. In particular, multiplication by the smeared delta and integration over the domain yields:

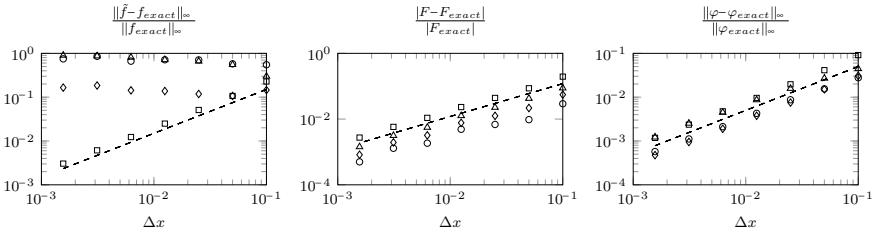


Fig. 1.4 Errors in $\hat{f}$, F, and φ versus grid spacing (h) for the Poisson problem. ○: δ_h^{hat}, ◇: δ_h^3, △: $\delta_h^{\cos}$, □: δ_h^G, --: first-order convergence. Reprinted from Goza et al. (2016) with permission

$$\begin{aligned}&\int_\Omega \int_\Gamma \mathbf{f}(\boldsymbol{X}(\boldsymbol{\xi}',t))\delta_h(\boldsymbol{x}-\boldsymbol{X}(\boldsymbol{\xi},t))\delta_h(\boldsymbol{x}-\boldsymbol{X}(\boldsymbol{\xi}',t))d\boldsymbol{\xi}'d\boldsymbol{x}\\&=\int_\Omega \left[\left(\frac{\partial}{\partial t}-\frac{1}{\text{Re}}\nabla^2\right)\mathbf{u}(\boldsymbol{x})+\mathbf{u}\cdot\nabla\mathbf{u}+\nabla p\right]\delta_h(\boldsymbol{x}-\boldsymbol{X}(\boldsymbol{\xi},t))d\boldsymbol{x}\end{aligned} \tag{1.40}$$

Analogous to the example above, the integral operator of Eq. 1.40 has an unbounded inverse because it contains a continuous kernel for any Δx. Hence, the same logic applies as above, and filtering is required to obtain accurate results. In the next section, this filtering approach is used for fully coupled fluid–structure interaction simulations.

1.2.4 Strongly Coupled Fluid–Structure Interaction

In this section, we present the extension of the IBPM to a strongly coupled fluid–structure interaction (FSI) solver for thin elastic structures (Goza and Colonius 2017).

In general, for FSI simulations, one can distinguish between monolithic and partitioned methods. In the monolithic approach, the fluid and structural equations are described with one system of equations using the same discretization scheme, which is solved by a single solver. By construction, consistent fluid–structure interface conditions are imposed in monolithic solvers. On the other hand, the partitioned approach uses individual solvers for the fluid and the structural equations, which are then coupled via appropriate boundary conditions to satisfy the solid–fluid interface conditions. This is a modular approach, which enjoys popularity for industrial as well as academic applications since separately optimized solvers for the fluid and the solid domain can be utilized. On the other hand, the main challenge of partitioned approaches is that the solid–fluid interface conditions are not implicitly satisfied. Hence, within the context of partitioned approaches, there is yet another distinction regarding the coupling methodology, namely between weakly and strongly coupled FSI schemes. While weakly coupled methods do not enforce the (nonlinear) fluid–

solid interface constraints at each time step, strongly coupled methods do converge to the monolithic equations using subiterative schemes.

On the one hand, this makes strongly coupled methods computationally more expensive. On the other hand, the staggered nature of weakly coupled schemes makes them susceptible to the so-called added-mass effect, where spurious energy is generated at the solid–fluid interface (Piperno and Farhat 2001). In particular for small solid–fluid density ratios, this can lead to fatal instabilities (Causin et al. 2005; Borazjani et al. 2008; Le Tallec 2001; Förster et al. 2007; Li and Favier 2017). For that reason we restrict ourselves to strongly coupled methods.

To impose the nonlinear interface constraint, most strongly coupled methods require the solution of a large nonlinear system. The block Gauss–Seidel method is a commonly used iterative procedure for solving the nonlinear system of equations, though it requires a relaxation parameter (often chosen heuristically) and typically converges slowly for small structural densities. Another common nonlinear solver is the Newton–Raphson method, which removes the relaxation parameter and typically converges rapidly even for small density ratios. However, this approach involves linear systems with large Jacobian matrices that cannot be solved directly, necessitating the use of large matrix-vector products in the context of some iterative solution process (Degroote et al. 2009; Mori and Peskin 2008; Hou et al. 2012). Among others, these strategies are reviewed in Sotiropoulos and Yang (2014).

In the following, we focus on thin elastic structures and solve the nonlinear algebraic system using the Newton–Raphson method. However, we avoid the need to solve a large linear system (without introducing any additional approximation into the solution process) by employing a block-LU factorization of the linearized system (Goza and Colonius 2017). The fluid part of the system is treated with the IBPM as outlined above.

For the coupled fluid–structure problem, the governing equations in Eq. (1.1) are extended by the structural equation as

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \frac{1}{\mathrm{Re}} \nabla^2 \boldsymbol{u} + \int_{\Gamma(t)} \boldsymbol{f}_\Gamma(\boldsymbol{X}(\boldsymbol{\xi}, t), t) \delta(\boldsymbol{X}(\boldsymbol{\xi}, t) - \boldsymbol{x}) \mathrm{d}\boldsymbol{\xi}, \quad (1.41a)$$

$$\nabla \cdot \boldsymbol{u} = 0, \quad (1.41b)$$

$$\frac{\rho_s}{\rho_f} \frac{\partial^2 \boldsymbol{X}(\boldsymbol{\xi}, t)}{\partial t^2} = \frac{1}{\rho_f U_\infty^2} \nabla \cdot \boldsymbol{\sigma} + \mathbf{g}(\boldsymbol{X}) - \boldsymbol{f}_\Gamma(\boldsymbol{X}), \quad (1.41c)$$

$$\int_\Omega \boldsymbol{u}(\boldsymbol{x}) \delta(\boldsymbol{x} - \boldsymbol{X}(\boldsymbol{\xi}, t)) \mathrm{d}\boldsymbol{x} = u_\Gamma(\boldsymbol{\xi}, t) = \frac{\partial \boldsymbol{X}(\boldsymbol{\xi}, t)}{\partial t}, \quad (1.41d)$$

where the solid and fluid density are denoted by ρ_s and ρ_f, respectively. The Cauchy stress is denoted by $\boldsymbol{\sigma}$, $\mathbf{g}$ is a body force, and the characteristic velocity is U_∞. The time derivative in the above is understood to be Lagrangian and the Cauchy stress is related to the second Piola–Kirchhoff stress $\boldsymbol{S}_s$ by

$$\boldsymbol{S}_s = J\boldsymbol{F}^{-1}\boldsymbol{\sigma}_s\boldsymbol{F}^{-T}, \tag{1.42}$$

where $J = \det(\boldsymbol{F})$ and $\boldsymbol{F}$ denotes the deformation gradient

$$\boldsymbol{F} = \boldsymbol{I} + \nabla\boldsymbol{u}_s, \tag{1.43}$$

and $\boldsymbol{u}_s$ is the displacement field of the solid. The second Piola–Kirchhoff stress is here defined as

$$\boldsymbol{S}_s = \mathbb{C} : \boldsymbol{E} \tag{1.44}$$

where

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I}) = \frac{1}{2}(\nabla\boldsymbol{u}_s + \nabla\boldsymbol{u}_s^T + \nabla\boldsymbol{u}_s^T\nabla\boldsymbol{u}_s) \tag{1.45}$$

is the Green–Lagrangian strain tensor and the stiffness tensor $\mathbb{C}$ is related to Young's modulus E_s, the bulk shear modulus, and Poisson's ratio ν_s.

The governing equations for the fluid region are discretized as above, and the solid equations can be discretized using a standard finite element procedure for thin elastic beams in a co-rotational formulation (see Goza and Colonius 2017; De Borst et al. 2012 for details), yielding the semi-discrete equation for the solid:

$$M_s\ddot{X} + K_s(X) = F_s(\mathbf{g} + W_s(X)f), \tag{1.46}$$

where the mass matrix M_s, the stiffness matrix K_s, and load F_s are given by

$$M_s = \frac{\rho_s}{\rho_f}\sum_j^{N_{\text{el}}}\int_{\Gamma_j^0} B^\dagger B\,\mathrm{d}\boldsymbol{X}^0, \quad K_s(X) = \frac{1}{\rho_f U_\infty^2}\sum_j^{N_{\text{el}}}\int_{\Gamma_j^0} B_E^\dagger\sigma_s\,\mathrm{d}\boldsymbol{X}^0 \tag{1.47}$$

$$F_s = \sum_j^{N_{\text{el}}}\int_{\Gamma_j^0} B^\dagger B\,\mathrm{d}\boldsymbol{X}^0 = \frac{\rho_f}{\rho_s}M_s. \tag{1.48}$$

Here, the jth element of Γ in the undeformed domain is indicated by Γ_j^0 and the shape function matrix as well as their derivatives are given by B and B_E, respectively. This formulation is expressed in the co-rotational frame and therefore accounts for geometrical nonlinearity and assumes small strains (large strains can be incorporated into K_s without affecting the algorithm). For further details, we refer to standard finite element textbooks (De Borst et al. 2012; Bathe 1996).

Using the discretizations above, the fully coupled FSI equations can be written as a first-order system of differential-algebraic equation as

$$C^\dagger C\dot{s} = -N(u) + C^\dagger LCs - C^\dagger E^\dagger \tilde{f}, \tag{1.49}$$

$$M_s \dot{u}_\Gamma = -K_s(X) + F_s(\mathrm{g} + W_f(X)\tilde{f}), \tag{1.50}$$

$$\dot{X} = u_\Gamma, \tag{1.51}$$

$$ECs - u_\gamma = 0. \tag{1.52}$$

An Adams–Bashforth and Crank–Nicolson time-marching scheme is applied for the nonlinear term and the diffusive term, respectively, for Eq. (1.49) and an implicit Newmark scheme is used for Eqs. (1.50)–(1.51). Equation (1.52) is evaluated at the current time step. This yields the following nonlinear system of algebraic equations:

$$C^\dagger AC s^{n+1} + C^\dagger E^\dagger \tilde{f}^{n+1} = r_f^n, \tag{1.53}$$

$$\frac{4}{\Delta t^2} M_s X^{n+1} + K_s(X^{n+1}) - F_s W^{n+1} \tilde{f}^{n+1} = r_{u_\Gamma}^n, \tag{1.54}$$

$$\frac{2}{\Delta t} X^{n+1} - u_\Gamma^{n+1} = r_X^n, \tag{1.55}$$

$$ECs^{n+1} - u_\Gamma^{n+1} = 0, \tag{1.56}$$

where $A = \frac{1}{\Delta t} I - \frac{1}{2} L$, $r_n^f = (\frac{1}{\Delta t} C^T C + \frac{1}{2} C^T LC)s^n + \frac{3}{2} C^T N(Cs^n) - \frac{1}{2} C^T N(Cs^{n-1})$, $r_{u_\Gamma}^n = M(\frac{4}{\Delta t^2} X^n + \frac{4}{\Delta t} u_\Gamma + \dot{u}_\Gamma^n) + Qg$, and $r_n^X = u_\Gamma + \frac{2}{\Delta t} X^n$.

To solve the nonlinear system, an iterative procedure is applied and the solution at time step n is used to initialize the iterative procedure at $k = 0$. During the kth iteration, the variables are updated as $X_{k+1}^{n+1} = X_{k+1}^{n+1} + \Delta X$ and $u_{\Gamma,k+1}^{n+1} = u_{\Gamma,k}^{n+1} + \Delta u_\Gamma$, which yields the following system:

$$\begin{bmatrix} C^T AC & 0 & 0 & C^T E^\dagger \\ 0 & 0 & \frac{4}{\Delta t^2} M + K_k & -F_s W_k^{n+1} \\ 0 & -I & \frac{2}{\Delta t} I & 0 \\ EC & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} s^{n+1} \\ \Delta u_\Gamma \\ \Delta X \\ \tilde{f}_{k+1}^{n+1} \end{bmatrix}$$

$$= \begin{bmatrix} r_n^f + O(\Delta t) \\ r_{u_\Gamma}^n - \frac{4}{\Delta t^2} M X_k^{n+1} - K_s(X_k^{n+1}) + O(\Delta t) \\ r_n^X - \frac{2}{\Delta t} X_k^{n+1} + u_{\Gamma,k}^{n+1} \\ u_{\Gamma,k}^{n+1} + O(\Delta t) \end{bmatrix} := \begin{bmatrix} r_n^f \\ r_{u_\Gamma,k} \\ r_{X,k} \\ r_{c,k} \end{bmatrix}, \tag{1.57}$$

where $K_k = dK_s/dX|_{X=X_k^{n+1}}$. For flags, the stiffness matrix has well known analytical expressions (De Borst et al. 2012; Bathe 1996). The linear system Eq.(1.57) can be factored using a block-LU decomposition, which yields

$$s^* = (C^T AC)^{-1} r_n^f \tag{1.58}$$

$$\begin{bmatrix} B_k^{n+1} & I \\ -\frac{2}{\Delta t} F_s W_k^{n+1} & \hat{K}_k \end{bmatrix} \begin{bmatrix} \tilde{f}_{k+1}^{n+1} \\ \Delta u_\Gamma \end{bmatrix} = \begin{bmatrix} ECs^* - r_{c,k} \\ \frac{2}{\Delta t} r_{u_\Gamma,k} - r_{X,k} \end{bmatrix} \tag{1.59}$$

$$\Delta X = \frac{\Delta t}{2} (\Delta u_\Gamma + r_{X,k}) \tag{1.60}$$

$$s^{n+1} = s^* - (C^T AC)^{-1} C^T E^\dagger \tilde{f}^{n+1}, \tag{1.61}$$

where $\hat{K}_k := \frac{4}{\Delta t^2} M_s + K_k$ and $B_k^{n+1} := EC(C^\dagger AC)^{-1} C^T E^\dagger$. The LU-factorized equations (1.58)–(1.61) are analogous to the previous factorizations but now include the fully coupled FSI scheme. Note that Eq. (1.58) does not depend on variables at time $n+1$ and thus must only be solved once per time step. Moreover, s^{n+1} is only updated after Eqs. (1.59)–(1.60) have converged in the iterative process. Hence, the iterations are restricted to Eqs. (1.59)–(1.60) which have dimensions of the order of number of body points rather then the entire flow domain.

The Poisson-like problems arising from Eq. (1.58), Eq. (1.61) and from each matrix-vector multiply with $B_{(k)}^{n+1}$ can be solved efficiently with fast Fourier transforms (FFT). In Goza and Colonius (2017), it was also argued that since the floating point operations of the FFT scale with the number points in the flow domain, it may be favorable to compute and store $(\hat{K}_k)^{-1}$. In that case, an analytical block Gaussian elimination of Eq. (1.59) may be performed to arrive at

$$\left(B_k^{n+1} + \frac{2}{\Delta t} (\hat{K}_k)^{-1} F_s W_k^{n+1} \right) \tilde{f}_{(k+1)}^{n+1} = ECs^* - r_{c,k} - \frac{2}{\Delta t} (\hat{K}_k)^{-1} r^{u_\Gamma,k} + r_{X,k} \tag{1.62}$$

$$\Delta u_\Gamma = \frac{2}{\Delta t} (\hat{K}_k)^{-1} (r_{u_\Gamma,k} + F_s W_k^{n+1} \tilde{f}_k^{n+1}) - r_{X,k} \tag{1.63}$$

Equation (1.62) can then be solved with a BICGSTAB scheme, which typically converges in a few iterations. In Goza and Colonius (2017), it was also shown that this iteration procedure typically converges in a few iterations and does not rely on heuristic relaxation parameters as for Gauss–Seidel-based approaches.

1.2.5 Example: The Inverted Flag Problem

Flow past a flag that is clamped at its leading edge is a canonical problem (Taneda 1968) and serves as an important benchmark problem for the development of numerical schemes (see Shelley and Zhang 2011 for a review). By contrast, when the flag is inverted, i.e., the trailing edge is clamped, only a few studies can be found in the literature (Kim et al. 2013; Gurugubelli and Jaiman 2015; Ryu et al. 2015). This configuration, however, is of interest due to its rich dynamical behavior, which includes small-deflection flapping, large-amplitude flapping, and chaotic flapping. From a numerical perspective, this setup is particular challenging as these regimes span a

large range of solid–fluid density ratios and exhibit large deformations that require a strongly coupled FSI solver. In Goza et al. (2018), this setup was investigated using the approach described in the previous sections. In addition to validating the numerics, these studies show that the IB method can be readily adapted for use in stability and bifurcation analysis. A steady-state solver employing a Newton–Raphson iteration was used to determine (potentially unstable) equilibria of the full fluid–structure system, and linearizations of the discretized equations lead to large, sparse systems of algebraic equations whose stability properties were efficiently determined using Arnoldi methods.

For the inverted flag, there are three independent non-dimensional parameters that govern the system. These are the Reynolds number $\mathrm{Re} = \frac{U_\infty L}{\nu_s}$, the mass ratio $M_\rho = \frac{\rho_s h}{\rho_f L}$, and the bending stiffness $K_B = \frac{D}{\rho_f U_\infty^2 L^3}$, where ρ_f (ρ_s) is the fluid (structure) density, U_∞ is the free-stream velocity, L is the flag length, h is the flag thickness, and the flexural rigidity is given by $D = Eh^3/(12(1-\nu^2))$ with Young's modulus E. In what follows, we consider the case of $\mathrm{Re} = 200$ and $M_\rho = 0.05$ and present the effect of decreasing the flag's stiffness K_B. More configurations and thorough analysis can be found in Goza et al. (2018).

With decreasing K_B, the flag undergoes a transition from the undeformed equilibrium (I) regime to the deformed equilibrium (II) through a divergence instability. With decreasing stiffness, the deformed equilibrium is associated with an increasingly large tip deflection and transitions from stable to unstable regimes. In particular, the small-deflection flapping regime (III) is reached by a supercritical Hopf bifurcation of the deformed equilibrium state. Decreasing the flag's stiffness further leads to large-amplitude flapping (IV, see also Fig. 1.5), which can be associated to classical vortex-induced vibration for the small density ratios as presented here. This is in contrast to heavier flags, for which large-amplitude flapping is not a classical vortex-induced vibration. For a stiffness of $K_B = 0.32$, snapshots of one flapping period are shown in Fig. 1.5 by means of vorticity contours.

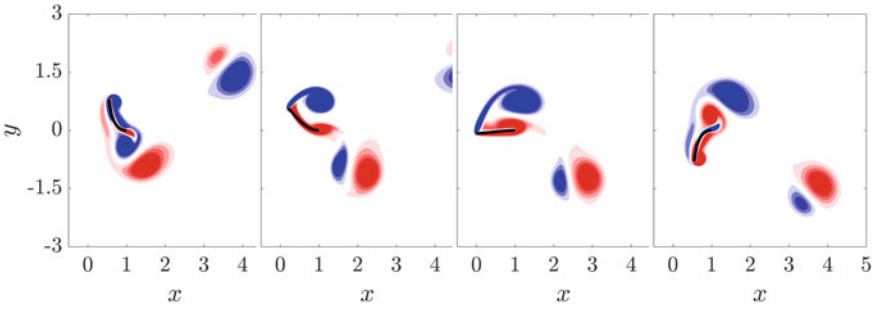


Fig. 1.5 Vorticity contours at four snapshots of a flapping period of a flag in large-amplitude flapping for $M_\rho = 0.05$. The Reynolds number and flag flexibility were chosen as $\mathrm{Re} = 200$, $K_B = 0.32$. Contours are in 18 increments from -5 to 5. Reprinted from Goza et al. (2018) with permission

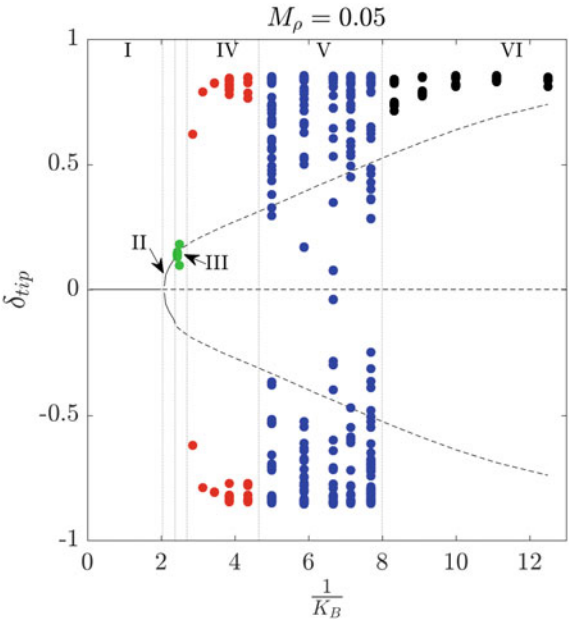


Fig. 1.6 Bifurcation diagram of inverted flag dynamics at Re = 200, showing tip deflection δ_{tip} as a function of inverse stiffness $(1/K_B)$. Regimes are denoted as I: undeformed equilibrium, II: deformed equilibrium, III: small-deflection deformed flapping, IV: large-amplitude flapping, V: chaotic flapping, VI: deflected mode. Reprinted from Goza et al. (2018) with permission

With decreasing stiffness, the system transitions to the chaotic flapping regime (V), and finally to the deflected mode regime (VI), where oscillations are primarily driven by vortex shedding. The chaotic regime is characterized by a strange attractor that alternatively samples regimes IV and VI.

These phenomena are summarized in the bifurcation diagram in Fig. 1.6 as obtained through nonlinear simulations. The simulations were started with an undeflected flag and an impulsively started flow at free-stream velocity U_∞. During the initial transient, a small body force was used to trigger any instabilities in the system. All simulations were run for at least 15 flapping cycles except for the chaotic state which requires 55 cycles. The first several cycles were neglected to avoid accounting for initial conditions. In Fig. 1.6, a set of markers at a given stiffness represents the tip deflection values δ_{tip} from a single nonlinear simulation when the flag changes direction. That is, the markers correspond to the zero-tip-velocity Poincaré sections of a velocity–displacement phase portrait of the leading edge. In addition, solid and dashed lines represent stable and unstable equilibria, respectively.

1.3 Fast Lattice Green's Function for External Flows

In this section, the immersed boundary lattice Green's function (IBLGF) method as proposed in Liska and Colonius (2017) is presented. The IBLGF is based on the unbounded domain lattice Green's function (LGF) flow solver (Liska and Colonius

2016) and the distributed Lagrange multiplier method to impose the no-slip boundary condition. The governing Navier–Stokes equations are spatially discretized on an unbounded staggered Cartesian grid, which retains crucial conservative, commutative, orthogonality, and symmetry properties of standard staggered Cartesian discretizations of infinite domains.

The advantage, however, is the use of the lattice Green's function technique, which implicitly satisfies the natural free-space boundary condition and allows block-adaptive grids to restrict the computation to a finite region (set of grid points) where the vorticity is nonzero (exceeds a small threshold). This is in contrast to common IB methods, which employ spatially truncated domains with approximate free-space boundaries. These approximations introduce blockage errors, which affect accuracy and may even change the dynamics of the flow (Tsynkov 1998; Colonius 2004; Pradeep and Hussain 2004). Thus, large computational domains in combination with stretched grids (Taira and Colonius 2007; Yun et al. 2006; Wang and Zhang 2011), local refinement (Roma et al. 1999; Griffith et al. 2007), and far-field approximations (Colonius and Taira 2008) are required to limit influence of the approximate free-space boundary condition (see also Sect. 1.2). In contrast, due to the natural free-space boundary inherent to IBLGF, an adaptive domain snugly conforming to regions of non-negligible vorticity and free of free-space boundary errors may be used for IBLGF.

By using a viscous integrating factor half-explicit Runge–Kutta scheme (IF-HERKS) in combination with an approximation-free nested projection technique and exploiting the aforementioned algebraic properties of the discrete operators, the projection steps reduce to simple discrete elliptic problems. These can in turn be solved efficiently using parallel lattice Green's function fast multipole methods (LGF-FMM) (Liska and Colonius 2014).

The operators satisfy the following topological and mimetic properties:

$$\text{Symmetry} \qquad \bar{D} = -G^\dagger, \bar{G} = -D^\dagger, \quad \bar{C} = C^\dagger \tag{1.64}$$

$$\text{Orthogonality} \qquad \text{Null}(C) = \text{Im}(G), \text{Null}(D) = \text{Im}(G) \tag{1.65}$$

$$\text{Mimetic} \qquad L_C = -G^\dagger G, L_\mathcal{F} = -GG^\dagger, L_\mathcal{E} = -D^\dagger D, L_\mathcal{V} = -DD^\dagger \tag{1.66}$$

$$\text{Commutativity} \qquad L_\mathcal{F} G = G L_C \tag{1.67}$$

1.3.1 Time Integration

In incompressible flow solvers, it is typical to use split time-stepping schemes where the viscous terms are advanced with an implicit method (alleviating any viscous time step constraint), whereas the advective terms are advanced with an explicit method which, despite introducing a CFL constraint on the time step, avoids iterative solution of nonlinear equations. A variety of specialized splitting methods have been employed, typically achieving second-order temporal accuracy. These schemes must also be cognizant of the kinematic constraints, which in general include enforcing a

divergence-free velocity field but in the case of IB methods, also include enforcing the no-slip condition. In other words, the spatially discretized incompressible Navier–Stokes equations comprise a discrete algebraic equation of index 2 (DAE-$i2$). The CN-AB2 scheme used in the previous sections is a particularly common choice for tackling such systems.

In the LGF approach, there is an opportunity to improve the time-marching scheme because the viscous terms can be integrated exactly using an integrating factor, which in turn can be done efficiently using an LGF. Once this is done, the time stepper for the remaining DAE system need no longer be split, meaning that explicit Runge–Kutta methods for DAE systems suffice, which in turn enables a wide variety of tailored schemes (low dissipation, low memory, etc.) to be employed. In particular, a family of half-explicit Runge–Kutta (HERK) methods are derived in Liska and Colonius (2016). Note that the "half-explicit" terminology refers to the solution of ODEs and algebraic constraints—not to any viscous/inviscid splitting.

The discrete integrating factor $E_Q(t)$ is the solution of the discrete heat equation $\frac{\mathrm{d}h}{\mathrm{d}t} = \kappa L_Q h$. Hence, for a given u at time τ and the integrating factor $H_Q = E_Q\left(\frac{t-\tau}{(\Delta x)^2 \mathrm{Re}}\right)$, Eq.(1.2) for $t > \tau$ is given by

$$\frac{\mathrm{d}v}{\mathrm{d}t} + [H_{\mathcal{F}}(t)]\tilde{N}([H_{\mathcal{F}}^{-1}(t)]v, t) = -Gb - [H_{\mathcal{F}}(t)][E(t)]^{\dagger}\tilde{f}, \tag{1.68a}$$

$$G^{\dagger}v = 0, \tag{1.68b}$$

$$[E][H_{\mathcal{F}}^{-1}(t)]v = u, \tag{1.68c}$$

where $v = [H_{\mathcal{F}}(t)]u$ and $b = [H_C(t)]p$. The above system constitutes a DAE-$i2$ that can be solved efficiently using an s-stage HERK scheme. The s-stages of the HERK are defined using the superscript i and time $t_k = k\Delta t$ with the time step Δt in indicated by subscript k. We now group the Lagrange multipliers, the right-hand side and the operators together such that

$$\lambda_k^i = \begin{bmatrix} p_k^i \\ \tilde{f}_k^i \end{bmatrix}, \quad \zeta_k^i = \begin{bmatrix} 0 \\ u(t_k^i) \end{bmatrix}, \quad Q^i = \left[G \; [E(t_k^i)]^{\dagger}\right], \forall i \in [1, s] \tag{1.69}$$

and we introduce the following:

$$u_k^i(\boldsymbol{n}) = \left[E_{\mathcal{F}}\left(\frac{-\bar{c}\Delta t}{(\Delta x)^2 \mathrm{Re}}\right)\right] v_k^i(\boldsymbol{n}), \quad p_k^i(\boldsymbol{n}) = \left[E_{\mathcal{F}}\left(\frac{-\bar{c}\Delta t}{(\Delta x)^2 \mathrm{Re}}\right)\right] b_k^i(\boldsymbol{n}). \tag{1.70}$$

The kth time step of the IF-HERK(u_k, t_k) scheme with the shifted coefficients $\tilde{a}_{i,j}$ and the shifted nodes $\tilde{c}_i$ (Liska and Colonius 2016; Hairer et al. 2006) can be summarized as follows:

1. Initialize: Set $u_k^0 = u_k$, $t_k^0 = t_k$
2. Multi-stage: for $i = 1, 2, \cdots, s$ solve the linear system:

$$\begin{bmatrix} (H^i_{\mathcal{F}})^{-1} & Q^{i-1}_k \\ (Q^i_k)^\dagger & 0 \end{bmatrix} \begin{bmatrix} u^i_k \\ \lambda^i_k \end{bmatrix} = \begin{bmatrix} r^i_k \\ \zeta^i_k, \end{bmatrix} \tag{1.71}$$

where

$$H^i_{\mathcal{F}} = E_{\mathcal{F}} \left(\frac{(\tilde{c}_i - \tilde{c}_{i-1})\Delta t}{(\Delta x)^2 \mathrm{Re}} \right), \quad r^i_k = h^i_k + \Delta t \sum_{j=1}^{i-1} \tilde{a}_{i,j} w^{i,j}_k + g^i_k \tag{1.72}$$

$$g^i_k = -a_{i,i} \Delta t \tilde{N}(u^{i-1}_k, t^{i-1}_k) \quad t^i_k = t_k + + \tilde{c}_i \Delta t. \tag{1.73}$$

The variables h^i_k and $w^{ki,j}_k$ are recursively computed for $i > 1$ and $j > i$ using

$$h^i_k = H^{i-1}_{\mathcal{F}} h^{i-1}_k, \quad h^1_k = 0 \tag{1.74}$$

$$w^{i,j}_k = H^{i-1}_{\mathcal{F}} w^{i-1,j}_k, \quad w^{i,i}_k = (\tilde{a}_{i,i}\Delta t)^{-1} \left(Q^{i-1}_k \hat{\lambda}^i_k \right) \tag{1.75}$$

3. Finalize: Set $u_{k+1} = u^s_k$, $\lambda_{k+1} = (\tilde{a}_{s,s}\Delta t)^{-1} \hat{\lambda^s_k}$ and $t_{k+1} = t^s_k$.

1.3.2 Linear Solver

The solution of Eq. (1.71) dominates the computational costs and is efficiently solved using the exact projection technique. Note that in contrast to 2D discrete nullspace (discrete streamfunction) methods (see, Sect. 1.2), the following formulation does not express the discrete velocity–pressure equations as discrete streamfunction–vorticity equations. In three dimensions, the resulting discrete Poisson problems are scalar problems in the case of the velocity–pressure formulation but vector problems in the case of streamfunction–vorticity formulation. Equation (1.71) can be rewritten in terms of p^i_k and $\tilde{f}^i_k$

$$M^i_k \begin{bmatrix} u^i_k \\ \hat{p}^i_k \\ \tilde{f}^i_k \end{bmatrix} = \begin{bmatrix} (H^i_{\mathcal{F}})^{-1} & G & (E^{i-1}_k)^\dagger \\ G^\dagger & 0 & 0 \\ E^i_k & 0 & 0 \end{bmatrix} \begin{bmatrix} u^i_k \\ \hat{p}^i_k \\ \tilde{f}^i_k \end{bmatrix} = \begin{bmatrix} r^i_k \\ 0 \\ u^i_k \end{bmatrix}, \tag{1.76}$$

where $\hat{p}^i_k / p^i_k = \hat{f}^i_k / \tilde{f}^i_k = \tilde{a}_{s,s}$, $u^i_k = u(t^i_k)$, and $E^i_k = E(t^i_k)$. Note that M^i_k is in general not symmetric and cannot be symmetrized as the image of the regularization operator and the interpolation operator are different. The asymmetry is inherent to HERK integration of DAE systems of index 2 with time-dependent constraint operators (Hairer et al. 2006; Brasey and Hairer 1993). Note, however, M^i_k retains symmetry for flow past non-moving rigid bodies. The nested projection technique for Eq. (1.76), obtained from an operator–block-LU decomposition of M^i_k can be written in projection-like form

$$\text{Solve for intermediate velocity:} \quad (H^i_{\mathcal{F}})^{-1} u^* = r^i_k \tag{1.77a}$$

$$\text{Solve for intermediate pressure:} \quad G^\dagger H_{\mathcal{F}} G d^* = G^\dagger u^* \tag{1.77b}$$

$$\text{Solve for intermediate IB forces:} \quad S^i_k f^* = E^i_k [u^* - H^i_{\mathcal{F}} G d^*] - u^i_k \tag{1.77c}$$

$$\text{Update forces:} \quad \hat{f}^i_k = f^* \tag{1.77d}$$

$$\text{Correct pressure:} \quad \hat{p}^i_k = p^* - (G^\dagger H^i_{\mathcal{F}} G)^{-1} G^\dagger H^i_{\mathcal{F}} (E^{i-1}_k)^\dagger \hat{f}^i_k \tag{1.77e}$$

$$\text{Correct velocity:} \quad u^i_k = u^* - H^i_{\mathcal{F}} [G \hat{p}^i_k + (E^{i-1}_k)^\dagger \hat{f}^i_k] \tag{1.77f}$$

where the force Schur complement of the LU decomposition of Eq. (1.76) S^i_k is given by

$$S^i_k = E^i_k H^i_{\mathcal{F}} [\mathrm{I} - G(G^\dagger H^i_{\mathcal{F}} G)^{-1} G^\dagger H^i_{\mathcal{F}}] (E^{i-1}_k)^\dagger, \tag{1.78}$$

where the identity operator is indicated by I. By exploiting the mimetic, orthogonality and commutativity properties, the above can be rewritten in a computationally more convenient way such as

$$L_C p^* = -G^\dagger r^i_k \tag{1.79a}$$

$$S^i_k \hat{f}^i_k = E^i_k H^i_C [r^i_k - G^\dagger p^*] - u^i_k \tag{1.79b}$$

$$\hat{p}^i_k = p^* + L_C^{-1} G^\dagger (E^{i-1}_k)^\dagger \hat{f}^i_k \tag{1.79c}$$

$$u^i_k = H^i_{\mathcal{F}} [r^i_k - G \hat{p}^i_k - (E^{i-1}_k)^\dagger \hat{f}^i_k], \tag{1.79d}$$

where the Schur complement simplifies to

$$S^i_k = E^i_k [H^i_{\mathcal{F}} + G(H^i_C)^{-1} L_C G^\dagger] (E^{i-1}_k)^\dagger. \tag{1.80}$$

Note that with the exception of $\hat{f}^i_k$, every term can efficiently be computed with either the point-operator representation of discrete operators or the Lattice Green's function fast multipole method (LGF-FMM), which will be outlined in some detail in the next section.

With regards to the solving the force Schur complement (1.79b), there exits either the possibility to use dense linear algebra techniques or iterative methods. In case of iterative methods, the conjugate gradient is a suitable candidate for a symmetric Schur complement, implying rigid and stationary immersed boundaries or more generally, for moving or deforming geometries, Krylov solvers such as GMRES or BiCGSTAB. Note that for the rigid and non-moving case, the dense linear algebra route is more convenient since the construction of S^i_k needs to be done only once and a Cholesky decomposition can be stored and used to evaluate the forces.

1.3.3 Lattice Green's Function Method

In this section, we focus on how to efficiently solve elliptic difference equations with the lattice Green's function method (Liska and Colonius 2014) in unbounded domains as they occur in the IBLGF method. The solution may be obtained by convolution of the fundamental solution of the discrete operator (lattice Green's function) with the source terms of the equation. The LGF can be derived from Fourier integrals and approximated through its through its asymptotic expansion. A significant advantage of the FLGF method is that the formally unbounded meshes may be truncated such that only regions with non-negligible source are retained in the computational domain, yielding an adaptive block-structured mesh as shown exemplary in Fig. 1.9.

In what follows, the three-dimensional Poisson equation will serve as a model equation, which is defined as:

$$[\Delta\varphi](\boldsymbol{x}) = h(\boldsymbol{x}), \quad \text{supp}(h) \subseteq \Omega, \tag{1.81}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ and Ω denotes a bounded domain in $\mathbb{R}^3$. The target field φ can be obtained by convolution of the fundamental solution of the Laplace operator $\mathrm{G}^L(\boldsymbol{x}) = -1/(4\pi|\boldsymbol{x}|)$ with the source field $h(\boldsymbol{x})$ such that

$$\varphi(\boldsymbol{x}) = [\mathrm{G}^L * h](\boldsymbol{x}) = \int_{\Omega} \mathrm{G}^L(\boldsymbol{x} - \boldsymbol{\xi}) h(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}. \tag{1.82}$$

In the discrete setting, Eq. (1.81) can be expressed as

$$[L_Q\varphi](\boldsymbol{x}_i) = h(\boldsymbol{x}_i), \quad \text{supp}(h) \subseteq \Omega_h, \tag{1.83}$$

where $\varphi, h \in \mathbb{R}^Q$, $\boldsymbol{x}_i \in \mathbb{Z}^3$, and Ω_h is a bounded domain in $\mathbb{Z}^3$. The target can then be obtained by discrete convolution such that

$$\varphi(\boldsymbol{x}_i) = [G^L * h](\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_j \in \Omega_h} G^L(\boldsymbol{x}_i - \boldsymbol{x}_j) h(\boldsymbol{x}_i), \tag{1.84}$$

where G^L indicates the LGF of the discrete 7-pt Laplacian. By diagonalizing the Laplace operator L_Q in Fourier space, an expression for $G^L(\boldsymbol{x}_i)$ can be derived (see, e.g., Delves and Joyce 2001; Glasser and Zucker 1977), yielding

$$G^L(\boldsymbol{x}_i) = \frac{1}{8\pi^3} \int_{[-\pi,\pi]^3} \frac{\exp\left(-i\boldsymbol{x}_i\boldsymbol{\xi}\right)}{2\cos(\xi_1) + 2\cos(\xi_2) + 2\cos(\xi_3) - 6} \mathrm{d}\boldsymbol{\xi}. \tag{1.85}$$

In addition, Eq.(1.85) can equivalently be written as a one dimensional, semi-infinite integral as

$$G^L(\boldsymbol{x}_i) = -\int_0^\infty \exp(-6t) I_{x_1}(2t) I_{x_2}(2t) I_{x_3}(2t) \mathrm{d}t, \tag{1.86}$$

where $I_k(t)$ is the modified Bessel function of first kind and order k. Note that Eq.(1.86) can be evaluated using an adaptive Gauss–Kronrod quadrature or alike in a straightforward manner, but it is typically more efficient to evaluate the Green's function through its asymptotic expansion in the far-field, i.e., large $|\boldsymbol{x}_i|$. In particular, the target field φ can be written as

$$\varphi(\boldsymbol{x}_i) = \varphi^{\text{near}}(\boldsymbol{x}_i) + \varphi^{\text{far}}(\boldsymbol{x}_i) + \varepsilon(\boldsymbol{x}_i), \tag{1.87}$$

where

$$\varphi^{\text{near}}(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_j \in \Omega_h^{\text{near}}(\boldsymbol{x}_i)} G^L(\boldsymbol{x}_i - \boldsymbol{x}_j) h(\boldsymbol{x}_j) \tag{1.88}$$

$$\varphi^{\text{far}}(\boldsymbol{x}_i) = \sum_{\boldsymbol{x}_j \in \Omega_h \setminus \Omega_h^{\text{near}}(\boldsymbol{x}_i)} A_G^q(\boldsymbol{x}_i - \boldsymbol{x}_j) h(\boldsymbol{x}_j), \tag{1.89}$$

and Ω_h^{near}, $\varepsilon(\boldsymbol{x}_i)$ are the near field and the error due to approximating $G^L(\boldsymbol{x}_i)$ with $A_G^q(\boldsymbol{x}_i)$ in the far-field, respectively. The q-term asymptotic expansion of $G^L(\boldsymbol{x}_i)$ is defined such that $G^L(\boldsymbol{x}_i) = A_G^q(\boldsymbol{x}_i) + O(|\boldsymbol{x}_i|^{-2q-1})$ and for $q = 2$ it reads

$$A_G^2(\boldsymbol{x}) = -\frac{1}{4\pi|\boldsymbol{x}|} - \frac{x_1^4 + x_2^4 + x_3^4 - 3x_1^2x_2^2 - 3x_1^2x_3^2 - 3x_2^2x_3^2}{16\pi|\boldsymbol{x}|^7}. \tag{1.90}$$

In practice, the results from direct integration of Eq.(1.86) are tabulated for the near-field ($|\boldsymbol{x}_i| \leq 100$), and the asymptotic expansion with $q = 3$ and $q = 2$ can be used in the far-fields for $100 < |\boldsymbol{x}_i| \leq 600$ and $|\boldsymbol{x}_i| > 600$, respectively. This ensures an error bound of the asymptotic expansion compared to the direct integration of $|\varepsilon| < 10^{-12}$.

1.3.3.1 Fast Convolutions

The direct evaluation of Eq.(1.84) requires $O(N^2)$ amount of work for N degrees of freedom and is therefore prohibitive for large computational domains. A remedy is the fast multipole method (FMM), which reduces the computational complexity from $O(N^2)$ to $O(N)$. In particular, the FLGF method (Liska and Colonius 2014) uses a kernel-independent interpolation-based fast multipole method to compute the discrete convolutions in conjunction with block-wise FFT convolution. The FMM achieves linear complexity $O(N)$ by leveraging the fact that, the solution is much smoother in the far-field than in the near-field for an elliptic kernel. Thus, a low-rank representation of the kernel is sufficient to accurately compute the contribution of far-field, while only the near-field requires full-rank representation of the kernel.

Using the generic interpolation function $\phi(\boldsymbol{x})$ and the coarse grained sampling points $\boldsymbol{x}_0, ..., \boldsymbol{x}_{n-1}$, a low-rank approximation of the kernel $K(\boldsymbol{x}, \boldsymbol{y})$ may be obtained by

$$\tilde{K}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \phi(\boldsymbol{x}) K(\boldsymbol{x}_i, \boldsymbol{y}_i) \phi(\boldsymbol{y}). \tag{1.91}$$

The discrete convolution can then be approximated by

$$\varphi(\boldsymbol{x}_i) \approx \sum_{j=0}^{M-1} \tilde{K}(\boldsymbol{x}_i, \boldsymbol{y}_j) h(\boldsymbol{y}_j) = \sum_{j=0}^{M-1} \sum_{p=0}^{n-1} \sum_{q=0}^{n-1} \phi(\boldsymbol{x}_i) K(\boldsymbol{x}_p, \boldsymbol{y}_q) \phi(\boldsymbol{y}_j) h(\boldsymbol{y}_j) \quad i = 0, ..., N-1, \tag{1.92}$$

where N is the number of target points and M the number of source points. The near- and far-field contributions are accounted for by constructing a hierarchical decomposition of the domain for which Eq.(1.92) is evaluated recursively. Typically, an octree structure $\mathcal{T}$ (quadtree in two dimensions) is used for this purpose. Let the tree have a depth L_B and the root is assumed to have level 0 and the base level $L_B - 1$ corresponds to physical domain. The tree nodes are also referred to as octants and octants without children are leaf nodes. The set of leafs on level l is indicated by B^l_{Leafs}. In the context of the FLGF, each tree node corresponds to a region, which is defined to be a Cartesian block of $N_b = n_b^3$ cells. Further, the ith octant or block at level l is denoted by $\mathcal{B}_i^l$ and the set of all octants at level l by $B^l = \bigcup_{i=0}^{N_B^l} \mathcal{B}_i^l$, where N_B^l is the number of octants on level l. The set of children and the parents are denoted by $C(\mathcal{B}_i^l)$ and $\mathcal{P}(\mathcal{B}_i^l)$, respectively.

The target field $u_i^{L_B-1}$, defined on the octant $\mathcal{B}_i^{L_B-1}$, consist of both near- and far-field contributions. The near-field contribution is given by the interaction, i.e., convolution, with region $\mathcal{N}(\mathcal{B}_i^{L_B-1})$, containing the octant itself and the nearest neighbor octants on the finest tree level $L_B - 1$. The far-field contributions are then evaluated recursively for the levels $l = L_B - 1, \ldots, 0$ and are defined as the convolution with octants in the influence region $\mathcal{I}(\mathcal{B}_i^l) = \{\hat{\mathcal{B}}_i^l \in \mathcal{F}(\mathcal{B}_i^l) \setminus \mathcal{F}(\mathcal{B}_i^{l-1})\}$, which includes the well-separated octants, i.e., $\mathcal{F}(\mathcal{B}_i^l) = \bigcup_{l=0}^{L_B-1} B^l \setminus \mathcal{N}(\mathcal{B}_i^l)$, but excludes the regions well-separated from its parents $\mathcal{F}(\mathcal{P}(\mathcal{B}_i^l))$. Schematically, the domain decomposition in near and far-field regions is depicted in Fig. 1.7.

In the FLGF method, the octants are defined to be Cartesian blocks and the convolution between each block and its influence list can be computed by block-wise FFT-based convolutions, which reduces to a complex Hadamard product in Fourier space (see Liska and Colonius 2014, for details on FFT-based convolutions). Note that compared to a direct summation as in Eq.(1.84), the computational complexity is reduced from $O(N_b^2)$ to $O(N_b \log N_b)$ for each block convolution.

It should be clear from the above that given a union of source blocks $B_s = \bigcup_{i=0}^{N_s-1} \mathcal{B}_{s,i}$ and target blocks $B_t = \bigcup_{i=0}^{N_t-1} \mathcal{B}_{t,i}$ the convolution can be evaluated as the sum of the individual convolutions as

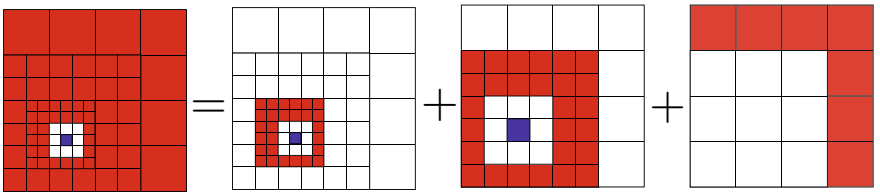


Fig. 1.7 Schematic of the hierarchical domain composition of the far-field (red, left) for an octant $\mathcal{B}_i^{L_B-1}$ (blue, left). While the near-field consist of the nearest neighbors only, the far-field is composed out of the set of influence lists for all levels. For level l the influence list contains the children of the nearest neighbors of $\mathcal{B}_i^l$'s parent, which are not contained in the near-field, i.e., are well separated. Reprinted from (Dorschner et al. 2020) with permission

$$\varphi_i = \sum_{j \in B_s} \text{conv}(G^L_{i-j}, f_j), \quad \text{for } i = 0, ..., N_t - 1, \tag{1.93}$$

where the convolution operator is denoted as conv and G^L_{i-j} is the vector containing the unique values of $G^L(\boldsymbol{x}_i - \boldsymbol{x}_j)$ evaluated on the grid points $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$ of the blocks $\mathcal{B}_{t,j}$ and $\mathcal{B}_{s,i}$, respectively.

With these definitions and the corresponding tree structure, the evaluation of Eq.(1.92) can be split into three consecutive steps. The first step is the *upward pass*, where the effective source terms are computed on each level by iterating bottom-up through the tree. Second, for each level, the convolution of each octant with its influence region is computed. This is called the *level interaction*. Finally, in the *downward pass* (iterating from the root to the leafs), all contributions are interpolated and accumulated on the next level. Schematically, the FLGF method is shown in Fig. 1.8 and can be summarized by the following algorithm:

1. *Upward pass*: Compute effective source terms at interpolation nodes
   For $l = L_B - 2, ..., 0$: For $i = 0, ...N_B^l$

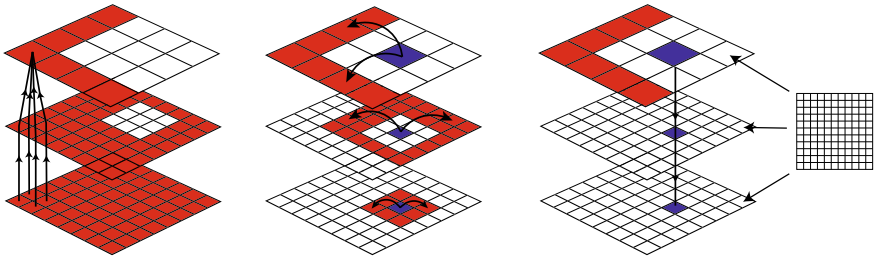


Fig. 1.8 Schematic of the fast multipole method: Left: *Upward pass*—Source regularization . Middle: *Level interaction*—Convolution of a block (blue) with its influence list (red). Right: *Downward pass*—Compute and accumulate the induced fields at the interpolation nodes. Note that each FMM cell corresponds to a Cartesian block in the FLGF. Reprinted from (Dorschner et al. 2020) with permission

$$\hat{h}_i^l = \sum_{j \in C(\mathcal{B}_i^l)} \hat{R}^{l+1} \hat{h}_j, \tag{1.94}$$

where the regularization operator $\hat{R}^{l+1}$ is the adjoint of the interpolation operator $\hat{E}^{l+1}$(see below).

2. *Level Interaction*: FFT convolution with the octant in the influence region
For $l = 0, ..., L_B - 1:$ For $i = 0, ...N_B^l$

$$\hat{v}_i^l = \sum_{j \in \mathcal{I}(\mathcal{B}_i^l)} \text{conv}(G_{i-j}^L, f_j), \tag{1.95}$$

where $\text{conv}(\cdot)$ is the FFT convolution operator.

3. *Downward pass*: Compute and accumulate induced field at interpolation nodes
For $l = 0, ..., L_B - 1:$ For $i = 0, ...N_B^l$

$$\hat{\varphi}_i^l = \hat{v}_i^l + \hat{E}_i^{l-1} \hat{\varphi}_i^{l-1}, \tag{1.96}$$

where the interpolation operator $\hat{E}^l$ interpolates from the parent onto the child block.

Owing to the regularity of the Cartesian block mesh, the interpolation operators are implemented using Lagrangian polynomials, and $n_I \leq 10$ interpolation nodes are used to yield a relative interpolation error of $\epsilon \approx 10^{-12}$ for an analytic function approximation. The regularization operator is given by the adjoint of the interpolation operator and often called anterpolation. In summary, the FLGF method combines the fastest methods for regular meshes, while retaining the geometrical flexibility and overall linear complexity inherent to FMM. Excellent computational rates and parallel performance have been reported in Liska and Colonius (2014) and (Dorschner et al. 2020).

1.3.3.2 Domain Adaptivity

As mentioned previously, the LGF method may truncate the formally unbounded computationally domain to regions with non-negligible source. When the LGF is employed in the context of the IBLGF this translates to solving the NSE only in regions that are dictated by the flow evolution and cells which, up to a prescribed threshold, do not affect the flow evolution may be removed in the course of a simulation. This is a direct consequence of the vorticity, $\omega = \nabla \times \boldsymbol{u}$, decaying exponentially at large distances form the immersed body. For instance, considering the solution of Eq. (1.79a):

$$p^*(\boldsymbol{x}_i) = [G_C^L * y](\boldsymbol{x}_i), \quad y(\boldsymbol{x}_i) = [-G^\dagger r_k^i](\boldsymbol{x}_i), \tag{1.97}$$

where $G^{\dagger} r_k^i$ is a discrete approximation of the divergence of the Lamb vector $\nabla \cdot \boldsymbol{l}$, where $\boldsymbol{l} = \boldsymbol{\omega} \times \boldsymbol{u}$, while $\boldsymbol{x}_i$ denotes a discrete location. Since $\omega \to 0$ at large distances from the body, it follows that also $\nabla \cdot \boldsymbol{l}$ and its discrete counterpart become exponentially small. Thus, the domain on which the field is induced is finite and is truncated when $G^{\dagger} r_k^i$ is smaller than a given threshold ϵ. When using this procedure, the discrete velocity needs to be refreshed using the discrete velocity $w = Cu$ in order to obtain a consistent velocity field. The domain adaptivity is implemented using the block-structured mesh as outlined above. For further implementation details on the domain adaptivity, the reader is referred to Liska and Colonius (2016).

1.3.4 Examples

1.3.4.1 Flow Past a Sphere

In this section, the benchmark simulation of the flow past an impulsively started sphere at Reynolds number Re = 3700 (Liska and Colonius 2017) is summarized. Additional benchmarks and validation studies can be found in Liska and Colonius (2017), Mengaldo et al. (2017). The turbulent flow past a sphere at Re = 3700 is a challenging, canonical benchmark problem and has thus been investigated by many researchers both experimentally (Kim and Durbin 1988) and numerically (Yun et al. 2006; Rodriguez et al. 2011; Dorschner et al. 2016). The nominal velocity of the sphere is $(U, 0, 0)$ and a small, initial perturbation is introduced in the flow field in order to break any symmetries. The flow is computed for $0 \leq t^*/U \leq 60$ using 81,920 Lagrangian markers, where t^* indicates initialization from a large-time solution of a sphere at Re = 1000. The sphere has unit radius and the grid spacing was chosen conservatively as $\Delta x \simeq 4.3 \times 10^{-3}$ to resolve the thin boundary layer on the surface of the sphere. The time step size is chosen such that the CFL number based on the maximum point-wise velocity remains below 0.9. The time-averaged results were obtained over the last five large-scale vortex shedding cycles (St = 0.215). A snapshot of the vortical structures using the Q-criterion is shown in Fig. 1.9 and is in line with what is reported in the literature. The adaptive nature of the grid is shown in Fig. 1.9, where the computational domain is comprised of only regions, where the vorticity is non-negligible. In this setup, the threshold for the grid adaptivity was chosen as $\epsilon = 5 \times 10^{-4}$ and verified to accurately capture the unbounded domain flow. The flow is further characterized by the mean surfaces stresses and the net body forces. First, we consider the mean skin friction coefficient $C_{\mathrm{f}} = \tau_w/(\frac{1}{2}\rho U^2)$, where τ_{w} is the local wall shear stress and the pressure coefficient $C_{\mathrm{p}} = (p - p_\infty)/(\frac{1}{2}\rho U)$, where p and p_∞ are the local and the free-stream pressure, respectively. Note that the raw point-wise values of the surface stress tensor are partially filtered using the boundary force post-processing technique (Goza et al. 2016), to construct a smoothed boundary forces. The time-averaged results of C_{p} and C_{f} along the polar angle θ are shown in Fig. 1.10. Finally, the mean values for drag coefficient $\overline{C}_{\mathrm{d}}$, base pressure

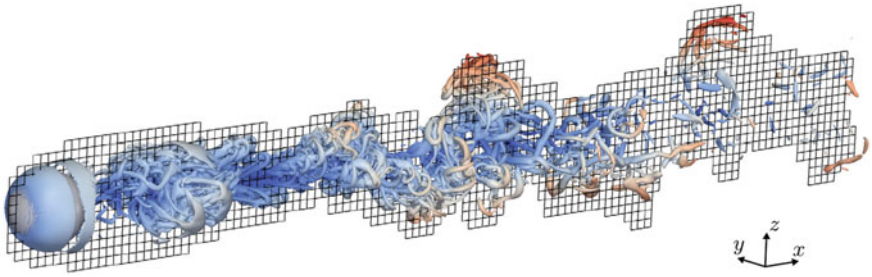


Fig. 1.9 Vortex cores in the wake of a sphere at Re = 3700 are illustrated by isosurfaces of constant Q-value. Depicted are isosurface of $QD^2/U^2 = 2$ colored by radial distance from the centerline of the sphere in the streamwise direction. The mesh is a cross-sectional cut of the block-wise adaptive computational domain that have been coarsened by a factor of two in each direction for visualization purposes. Reprinted from Liska and Colonius (2017) with permission

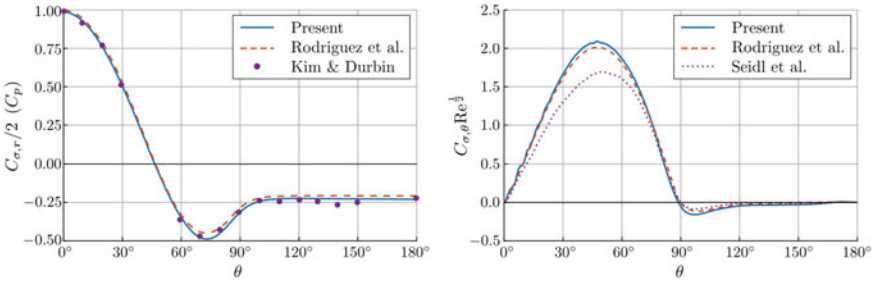


Fig. 1.10 Comparison of the time-averaged pressure (left) and skin friction (right) coefficients as functions of the polar angle, θ, for a sphere at Re = 3700. Results are compared to values reported by Rodriguez et al. (2011) (DNS at Re = 3700), Kim and Durbin (1988) (exp. at Re = 4200), and Seidl et al. (1997) (DNS at Re = 5000). Reprinted from Liska and Colonius (2017) with permission

coefficient $\overline{C}_{\mathrm{pb}}$, separation angle $\overline{\theta}_{\mathrm{s}}$, polar locations of the minimum surface pressure $\overline{\theta}_{\mathrm{p,min}}$ and of the maximum skin friction $\overline{\theta}_{\tau,\mathrm{max}}$ are reported in Table 1.1 and shows good agreement with the literature values.

1.3.4.2 Flow Past an Inclined, Rotating, Circular Disk

As an outlook, this setup can easily be extended to rotating disks by prescribing appropriate boundary velocities for the Lagrangian IB points. Such studies are of interest for developing novel designs of unmanned air vehicles or alike. While spheres and disks have both been studied extensively in literature, significant discrepancies between numerical and experimental measurements for disks and flat cylinders of large aspect ratio have fueled recent interest in further numerical investigations and thus makes it a good benchmark for the IBLGF (Gao et al. 2018; Tian et al. 2017;

Table 1.1 Turbulent flow past sphere at Re = 3700 and the comparison of the mean values for drag coefficient $\overline{C}_{\mathrm{d}}$, base pressure coefficient $\overline{C}_{\mathrm{pb}}$, separation angle $\overline{\theta}_{\mathrm{s}}$, polar locations of the minimum surface pressure $\overline{\theta}_{p,\mathrm{min}}$ and of the maximum skin friction $\overline{\theta}_{\tau,\mathrm{max}}$ with literature values

Contribution		Re	$\overline{C}_d$	$\overline{C}_{\mathrm{pb}}$	$\overline{\theta}_{\mathrm{s}}$	$\overline{\theta}_{\mathrm{p,min}}$	$\overline{\theta}_{\tau,\mathrm{max}}$
IBLGF	DNS	3700	0.389	−0.230	88.9	73	47
Yun et al. (2006)	LES	3700	0.355	−0.194	90	–	–
Rodriguez et al. (2011)	DNS	3700	0.394	−0.207	89.4	72	48
Dorschner et al. (2016)	DNS	3700	0.383	−0.220	89.993	–	–
Kim and Durbin (1988)	exp.	4200	–	−0.224	–	–	–
Seidl et al. (1997)	DNS	5000	0.38	–	89.5	71	50

Chrust et al. 2015). In addition, an extension to rotating disks (e.g., Frisbees) will be presented. The rotating disk has been suggested as a configuration for a micro-air vehicle based on the stabilizing influence of rotation on the flight dynamics (Potts and Crowther 2001, 2002; Lorenz 2007).

The Reynolds number is defined as $\mathrm{Re} = \frac{U_\infty D \sin(\alpha)}{\nu}$, where U_∞ is the free-stream velocity, α is the angle of attack, and D is the disk diameter, which is set to unity. The Strouhal number is defined by $\mathrm{St} = \frac{f D \sin(\alpha)}{U_\infty}$, where f is the frequency of the primary vortex shedding.

Mean drag and lift coefficients for various angles of attack and a fixed Reynolds number of Re = 500 are compared to the numerical simulations of Tian et al. (2017) in Fig. 1.11. The grid spacing was chosen to be $\Delta x = 0.012$ and the time step is set to $\Delta t = 0.004$. The adaptivity threshold for the grid was set to $\epsilon = 5 \times 10^{-4}$ unless stated otherwise. For simulations at lower Reynolds numbers the grid spacing is chosen based on the estimated scaling of the boundary layer. With an expected $\mathrm{Re}^{-\frac{1}{2}}$ scaling of the laminar boundary layer thickness, the value of $(\Delta x)\mathrm{Re}^{\frac{1}{2}}$ is kept approximately constant as in Liska and Colonius (2017). The CFL number based on the maximum point-wise velocity is kept below 0.5 by setting $\Delta t/\Delta x = 1/3$. While in Tian et al. (2017) the disk was modeled to have a thickness of $0.002D$, the

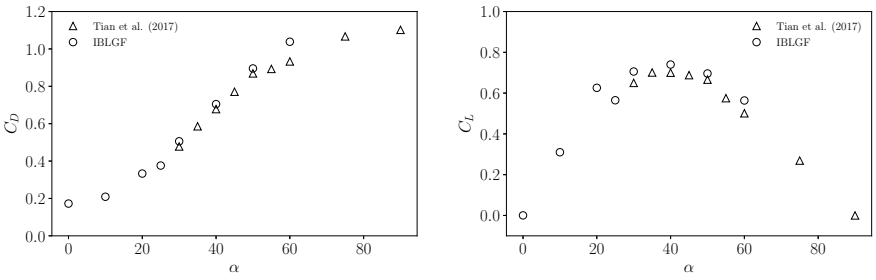


Fig. 1.11 Drag and lift coefficient for various angles of attack α at Re = 500

IBLGF simulations assume an infinitely thin disk, which is in turn regularized on the grid scale via the discrete delta function. Mean quantities are averaged over at least 170 convective time units for unsteady solutions ($\alpha \geq 25°$) and 130 units for steady solutions ($\alpha \leq 20°$). Good agreement between both numerical simulations validates the IBLGF. Exemplary, in Fig. 1.13 the vorticity isosurfaces for ω_x along with a slice of the computational mesh are shown for angles of attack of $\alpha = 20°$ and $\alpha = 70°$, respectively. Note the disjoint computational mesh for $\alpha = 70°$, which reduces the computational cost significantly compared to a uniform solver (Fig. 1.12).

In addition, the simulations are validated by comparison of the critical Reynolds number and Strouhal number as the flow transitions from a steady state regime to a periodic state through a supercritical Hopf bifurcation with increasing Reynolds number and fixed angle of attack. In Fig. 1.12, the threshold of the Hopf instability as a function of angle of attack is plotted in terms of critical Reynolds and Strouhal number. The critical values for both Reynolds and Strouhal number are obtained by extrapolating from unsteady cases run near the critical point as done in Ghaddar et al. (1986), Pereira and Sousa (1993). As apparent from the plot, the critical values

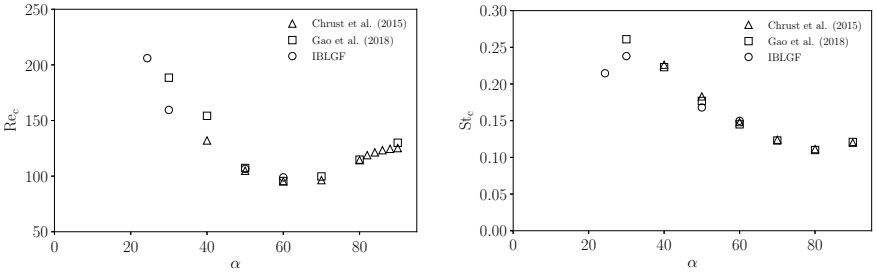


Fig. 1.12 Critical Reynolds number Re_c and critical Strouhal number St_c for various angles of attack α

(a) $\alpha = 25°$. Red, blue and grey isosurfaces are shown for $\omega_x = +2$, $\omega_x = -2$ and $|\omega| = 1.5$, respectively.

(b) $\alpha = 70°$. Red, blue and grey isosurfaces are shown for $\omega_x = +0.5$,$\omega_x = -0.5$ and $|\omega| = 0.25$, respectively. The adaptive threshold was set to $\epsilon = 5 \cdot 10^{-5}$ for this case.

Fig. 1.13 Isosurfaces of streamwise vorticity ω_x for flow over an infinitely thin disk at $\mathrm{Re} = 500$, $\lambda = 0$. The grid of blocks is shown on the spanwise xy-plane through the centerline of the disk. Each block consists of $10 \times 10 \times 10$ cells. Adaptive threshold is set to $\epsilon = 5 \times 10^{-5}$

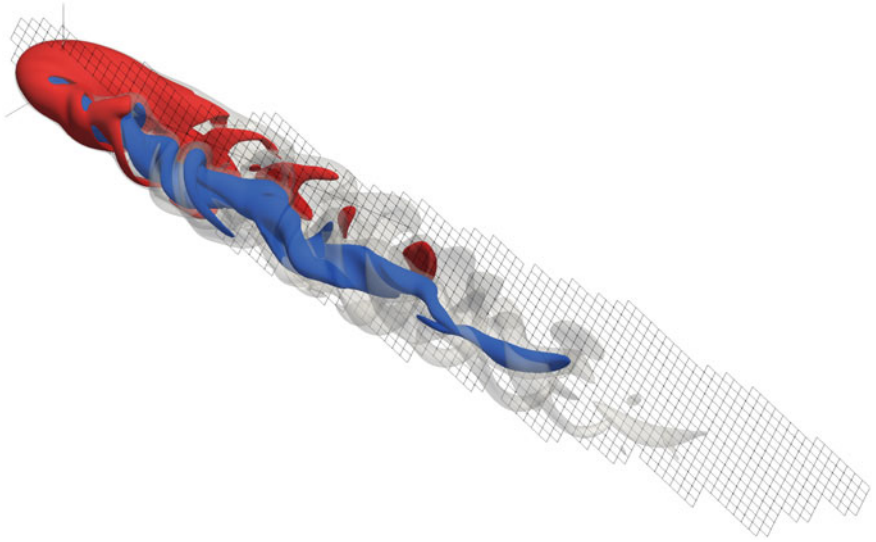


Fig. 1.14 Isosurfaces of vorticity for flow over an infinitely thin disk at Re = 500, an angle of attack $\alpha = 0$ and rotating at a tip-speed ratio of $\lambda = 3$. See also caption of Fig. 1.13

agree well with the numerical simulations of Gao et al. (2018), Chrust et al. (2015). Note that while (Chrust et al. 2015) modeled an infinitely thin disk, the simulations reported in Gao et al. (2018) modeled the disk to have a thickness of $0.02D$. A snapshot of a rotating disk with a tip-speed ratio of $\lambda = 3$ is shown in Fig. 1.14 as an outlook of future studies.

1.3.5 Toward AMR

Note that while in the IBLGF the grid is adaptive, the resolution is uniform, which can become limiting in many applications. In fact, while uniform Cartesian meshes can significantly decrease the cost per degree of freedom (DoF), the total number of DoF can be prohibitive for strongly anisotropic or inhomogeneous problems with localized source regions. This issue is particularly prominent for, e.g., high Reynolds number flows or problems where the range of scales is large. A particularly acute challenge is for bluff bodies where one must resolve thin attached laminar or turbulent boundary layers at the same time as a broad wake. Here, adaptive mesh refinement methods (AMR), pioneered by Berger and Colella (1989), Berger and Oliger (1984) can be crucial. These methods adapt the local density of the computational elements to the local spatial resolution requirements. In the context of the IB method, AMR techniques have been embedded (Roma et al. 1999; Griffith et al. 2007; Vanella et al. 2014) and have shown to greatly reduce the computational costs of simulations. In

addition, block-wise AMR has become popular compared to cell-wise refinement (see Dubey et al. 2014 for a survey of popular block-structured mesh refinement packages), since the overhead of the underlying data structure and load balancing has shown to be significantly more efficient (Nissen et al. 2013; Dreher and Grauer 2005).

While the FLGF method discussed above is adaptive and utilizes a hierarchical block-structured grid for the FMM solution, it is not an AMR technique. A natural extension of the FLGF is thus to a block-structured AMR solver without additional overhead of the underlying data structure. Note, however, that when refining the physical domain by embedding locally refined grid patches within the computational domain, the free-space boundary conditions implied by the lattice Green's functions become problematic since the refinement patches itself do have a well defined boundary condition, which is imposed by the surrounding domain and is not the free space. A remedy was found in Dorschner et al. (2020), by projecting the source field onto each level within its support by appropriate coarsening and interpolation operators and subsequent application of the FLGF method on each level independently. The resulting multi-resolution scheme retains linear computational complexity and second-order accuracy and additional allows for arbitrary block-wise refinement by factors of two. An exemplary mesh topology of the multi-resolution scheme is shown in Fig. 1.15 for a thin vortex ring.

While in Dorschner et al. (2020) the groundwork toward a multi-resolution IBLGF solver was laid, its incorporation into the full IBLGF solver is ongoing work.

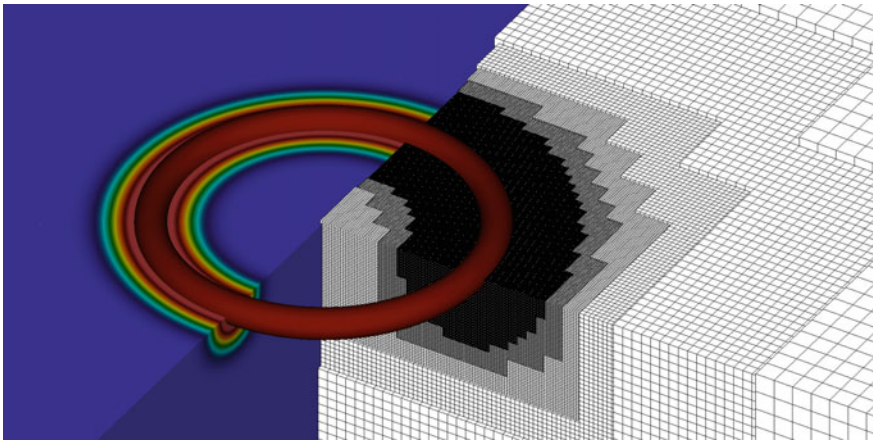


Fig. 1.15 Mesh topology and vorticity field for a thin vortex ring and the numerical solution of the streamfunction for five levels of refinement

1.4 Summary and Future Directions

In this chapter, we have summarized a projection approach to the IB method that exploits features of the underlying, uniform Cartesian mesh to achieve robust, efficient solution of incompressible external flow problems. Extensions for fully coupled FSI for thin elastic structures, and a novel fast lattice Green's function for 3D problems were highlighted. Examples problems such as inverted flag flutter, and flow past a sphere and rotating disks were presented to demonstrate the capabilities of the numerics for complex flow problems.

The IBPM takes advantages of formulating the continuity and no-slip boundary conditions as Lagrange multipliers to enforce these constraints to prescribed precision without the need of an artificial constitute relation. This formulation removes any additional constraints on the CFL number for rigid bodies, which is then only limited by the choice of the time-stepping algorithm. Through carefully chosen discrete operators, a modified Poisson equation arises, which is symmetric and positive definite and thus amenable to efficient numerical solvers such as conjugate-gradient method. In addition, we reviewed appropriate filtering techniques to accurately compute surfaces stresses and forces, which allowed the extension to strongly coupled fluid interaction for thin elastic structures. The simplicity of the IBPM formulation proves useful in finding (potentially unstable) steady equilibrium solutions of coupled FSI problems, as well as solving associated global linear stability problems. This capability was demonstrated on an inverted elastic flag setup.

When combined with a lattice Green's function approach, the IB method can be used for three-dimensional external flows while, at the same time, minimizing the computational expense by restricting the computational domain to a snug region near the immersed surface in which the vorticity is nonzero. The method naturally incorporates exact free-space boundary conditions. By using a viscous integrating factor half-explicit Runge–Kutta scheme in combination with the approximation-free nested projection technique and exploiting mimetic properties of the discrete operators, the projection steps reduce to simple discrete elliptic problems, which can be solved efficiently with the FLGF method. The FLGF method is based on a kernel-independent interpolation-based fast multipole method and therefore exhibits linear computational complexity, while taking advantage of a block-structured Cartesian mesh by using standard FFT routines for fast convolutions.

While the combined IBLGF approach has been demonstrated in benchmarks such as the turbulent flow past a sphere or flow past an inclined rotating disk, its extension to higher Reynolds numbers requires further developments. While it is true that uniform grids in general allow us to use efficient numerical solvers, the number of degrees of freedom becomes prohibitive for the large range of scales in high Reynolds number flows. Thus, future efforts are directed toward a multi-resolution IBLGF scheme. A block-structured multi-resolution algorithm based on LGFs for elliptic difference equations, which retains favorable computational complexity and necessary regularity has been detailed in Dorschner et al. (2020). Incorporation in the IBLGF will be reported in future contributions. Such multi-resolution scheme will

not only open the door for high Reynolds number direct numerical simulations but also inclusion of explicit turbulence models such as wall-resolved or wall-modeled large-eddy simulations (LES). In the context of IB, this has been successfully implemented in Piomelli and Balaras (2002), Iaccarino and Verzicco (2003), Yang and Balaras (2006), You et al. (2007), Catchirayer et al. (2018) (and references therein) and therefore provides a promising path toward high Reynolds numbers and applications with engineering relevant complexity. In addition, by using Krylov solvers such as GMRES, BIGSTAB, and their flexible extensions (Saad 1993; Chen et al. 2016) to solve the force Schur complement, the IBLGF scheme can be extended to a fully coupled three-dimensional fluid–structure solver. This would retain the benefits of IBLGF in terms of accuracy, scalability, and efficiency due to truncated computational domain, while increasing the range of applications significantly.

Acknowledgements BD gratefully acknowledges support from the Swiss National Science Foundation (SNF Grant No. P2EZP2_178436), TC was supported by ONR grant N00014-16-1-2734. We are likewise grateful to numerous students and collaborators, particularly Sam Taira, Sebastian Liska, and Andres Goza, who contributed to the development of IB projection techniques described herein. We thank Marcus Lee and Ke Yu for recent contributions to the AMR code development; Marcus Lee also contributed his results for flow past a spinning disk to this article.

References

Ahn HT, Kallinderis Y (2006) Strongly coupled flow/structure interactions with a geometrically conservative ALE scheme on general hybrid meshes. J Comput Phys 219(2):671–696

Bathe KJ (1996) Finite element procedures. Prentice-Hall, Englewood Cliffs, NJ

Berger MJ, Colella P (1989) Local adaptive mesh refinement for shock hydrodynamics. J Comput Phys 82(1):64–84

Berger MJ, Oliger J (1984) Adaptive mesh refinement for hyperbolic partial differential equations. J Comput Phys 53(3):484–512

Beyer RP, LeVeque RJ (1992) Analysis of a one-dimensional model for the immersed boundary method. SIAM J Numer Anal 29(2):332–364

Borazjani I, Ge L, Sotiropoulos F (2008) Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies. J Comput Phys 227(16):7587–7620. https://doi.org/10.1016/j.jcp.2008.04.028

Brasey V, Hairer E (1993) Half-explicit Runge-Kutta methods for differential-algebraic systems of index 2. SIAM J Numer Anal 30(2):538–552

Catchirayer M, Boussuge JF, Sagaut P, Montagnac M, Papadogiannis D, Garnaud X (2018) Extended integral wall-model for large-eddy simulations of compressible wall-bounded turbulent flows. Phys Fluids 30(6):065106

Causin P, Gerbeau JF, Nobile F (2005) Added-mass effect in the design of partitioned algorithms for fluid-structure problems. Comput Methods Appl Mech Eng 194(42–44):4506–4527. https://doi.org/10.1016/j.cma.2004.12.005

Chang W, Giraldo F, Perot B (2002) Analysis of an exact fractional step method. J Comput Phys 180(1):183–199

Chen J, McInnes LC, Zhang H (2016) Analysis and practical use of flexible BiCGStab. J Sci Comput 68(2):803–825

Chrust M, Dauteuille C, Bobinski T, Rokicki J, Goujon-Durand S, Wesfreid J, Bouchet G, Dušek J (2015) Effect of inclination on the transition scenario in the wake of fixed disks and flat cylinders. J Fluid Mech 770:189–209

Colella P, Graves DT, Keen BJ, Modiano D (2006) A Cartesian grid embedded boundary method for hyperbolic conservation laws. J Comput Phys 211(1):347–366

Colonius T (2004) Modeling artificial boundary conditions for compressible Flow. Ann Rev Fluid Mech 36(1):315–345

Colonius T, Taira K (2008) A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. Comput Methods Appl Mech Eng 197(25–28):2131–2146. https://doi.org/10.1016/j.cma.2007.08.014

De Borst R, Crisfield MA, Remmers JJ, Verhoosel CV (2012) Nonlinear finite element analysis of solids and structures. Wiley

Degroote J, Bathe KJ, Vierendeels J (2009) Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction. Comput Struct 87(11–12):793–801

Delves RT, Joyce GS (2001) On the Green function for the anisotropic simple cubic lattice. Ann Phys 291(1):71–133

Dorschner B, Frapolli N, Chikatamarla SS, Karlin IV (2016) Grid refinement for entropic lattice Boltzmann models. Phys Rev E 94(5):053311

Dorschner B, Yu K, Mengaldo G, Colonius T (2020) A fast multi-resolution lattice green's function method for elliptic difference equations. J Comput Phys 109270

Dreher J, Grauer R (2005) Racoon: A parallel mesh-adaptive framework for hyperbolic conservation laws. Parallel Comput 31(8–9):913–932

Dubey A, Almgren A, Bell J, Berzins M, Brandt S, Bryan G, Colella P, Graves D, Lijewski M, Löffler F et al (2014) A survey of high level frameworks in block-structured adaptive mesh refinement packages. J Parallel Distrib Comput 74(12):3217–3227

Fadlun E, Verzicco R, Orlandi P, Mohd-Yusof J (2000) Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161(1):35–60

Förster C, Wall WA, Ramm E (2007) Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. Computer Methods Appl Mech Eng 196(7):1278–1293. https://doi.org/10.1016/j.cma.2006.09.002

Gao S, Tao L, Tian X, Yang J (2018) Flow around an inclined circular disk. J Fluid Mech 851:687–714

Ghaddar N, Korczak K, Mikic B, Patera A (1986) Numerical investigation of incompressible flow in grooved channels. Part 1. Stability and self-sustained oscillations. J Fluid Mech 163:99–127

Glasser ML, Zucker IJ (1977) Extended Watson integrals for the cubic lattices. Proc Natl Acad Sci 74(5):1800–1801

Glowinski R, Pan TW, Periaux J (1998) Distributed Lagrange multiplier methods for incompressible viscous flow around moving rigid bodies. Comput Methods Appl Mech Eng 151(1–2):181–194

Goldstein D, Handler R, Sirovich L (1993) Modeling a no-slip flow boundary with an external force field. J Comput Phys 105(2):354–366

Goza A, Colonius T (2017) A strongly-coupled immersed-boundary formulation for thin elastic structures. J Comput Phys 336:401–411

Goza A, Liska S, Morley B, Colonius T (2016) Accurate computation of surface stresses and forces with immersed boundary methods. J Comput Phys 321:860–873

Goza A, Colonius T, Sader JE (2018) Global modes and nonlinear analysis of inverted-flag flapping. J Fluid Mech 857:312–344

Griffith BE, Hornung RD, McQueen DM, Peskin CS (2007) An adaptive, formally second order accurate version of the ibm-Peskin.pdf. J Comput Phys 223(1):10–49

Gurugubelli P, Jaiman R (2015) Self-induced flapping dynamics of a flexible inverted foil in a uniform flow. J Fluid Mech 781:657–694

Hairer E, Lubich C, Roche M (2006) The numerical solution of differential-algebraic systems by Runge-Kutta methods, vol 1409. Springer, Berlin

Hall CA (1985) Numerical solution of Navier-Stokes problems by the dual variable method. SIAM J Algebraic Discrete Methods 6(2):220–236
Hansen PC (1998) Rank-deficient and discrete illposed problems: numerical aspects of linear inversion, vol 4. SIAM
Hermansson J, Hansbo P (2003) A variable diffusion method for mesh smoothing. Commun Numer Methods Eng 19(11):897–908. https://doi.org/10.1002/cnm.639
Hirt CW, Amsden AA, Cook JL (1974) An arbitrary Lagrangian-Eulerian computing method for all flow speeds. J Comput Phys 14(3):227–253
Hou G, Wang J, Layton A (2012) Numerical methods for fluidstructure interaction—a review. Commun Comput Phys 12(2):337–377
Hu XY, Khoo B, Adams NA, Huang F (2006) A conservative interface method for compressible flows. J Comput Phys 219(2):553–578
Huang WX, Sung HJ (2009) An immersed boundary method for fluid-flexible structure interaction. Comput Methods Appl Mech Eng 198(33):2650–2661
Iaccarino G, Verzicco R (2003) Immersed boundary technique for turbulent flow simulations. Appl Mech Rev 56(3):331–347
Kallemov B, Bhalla A, Griffith B, Donev A (2016) An immersed boundary method for rigid bodies. Commun Appl Math Comput Sci 11(1):79–141
Kim H, Durbin P (1988) Observations of the frequencies in a sphere wake and of drag increase by acoustic excitation. Phys Fluids 31(11):3260–3265
Kim D, Cossé J, Cerdeira CH, Gharib M (2013) Flapping dynamics of an inverted flag. J Fluid Mech 736
Kirkpatrick M, Armfield S, Kent J (2003) A representation of curved boundaries for the solution of the Navier-Stokes equations on a staggered three-dimensional Cartesian grid. J Comput Phys 184(1):1–36
Kress R (2014) Linear integral equations, 3rd edn, vol 82. Springer, Berlin
Lai MC, Peskin CS (2000) An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. J Comput Phys 160(2):705–719
Le Tallec P (2001) Fluid structure interaction with large structural displacements. Computer Methods Appl Mech Eng 190(24–25):3039–3067. https://doi.org/10.1016/S0045-7825(00)00381-9
Li Z, Favier J (2017) A non-staggered coupling of finite element and lattice Boltzmann methods via an immersed boundary scheme for fluid structure interaction. Comput Fluids 143:90–102. https://doi.org/10.1016/j.compfluid.2016.11.008
Lilly DK (1965) On the computational stability of numerical solutions of time-dependent non-linear geophysical fluid dynamics problems. Mon Weather Rev 93(1):11–26
Liska S, Colonius T (2014) A parallel fast multipole method for elliptic difference equations. J Comput Phys 278:76–91
Liska S, Colonius T (2016) A fast lattice Green's function method for solving viscous incompressible flows on unbounded domains. J Comput Phys 316:360–384
Liska S, Colonius T (2017) A fast immersed boundary method for external incompressible viscous flows using lattice Green's functions. J Comput Phys 331:257–279
Lorenz RD (2007) Spinning flight: dynamics of frisbees, boomerangs, samaras, and skipping stones. Springer Science & Business Media
Mengaldo G, Liska S, Colonius KYT, Jardin T (2017) Immersed boundary lattice Green function methods for external aerodynamics. In: 23rd AIAA computational fluid dynamics conference, p 3621
Mittal R, Iaccarino G (2005) Immersed boundary methods. Ann Rev Fluid Mech 37:239–261
Mohd-Yusof J (1997) For simulations of flow in complex geometries. Ann Res Briefs 317–327
Mori Y, Peskin CS (2008) Implicit second-order immersed boundary methods with boundary mass. Comput Methods Appl Mech Eng 197(25–28):2049–2067
Morinishi Y, Lund TS, Vasilyev OV, Moin P (1998) Fully conservative higher order finite difference schemes for incompressible flow. J Comput Phys 143(1):90–124

Nakata T, Liu H (2012) A fluid-structure interaction model of insect flight with flexible wings. J Comput Phys 231(4):1822–1847
Nicolaides RA (1992) Direct discretization of planar div-curl problems. SIAM J Numer Anal 29(1):32–56
Nicolaides RA, Wu X (1997) Covolume solutions of three-dimensional div-curl equations. SIAM J Numer Anal 34(6):2195–2203
Nissen A, Kreiss G, Gerritsen M (2013) High order stable finite difference methods for the Schrodinger equation. J Sci Comput 55(1):173–199
Pereira J, Sousa J (1993) Finite volume calculations of self-sustained oscillations in a grooved channel. J Comput Phys 106(1):19–29
Perot B (2000) Conservation properties of unstructured staggered mesh schemes. J Comput Phys 159(1):58–89
Peskin CS (1972) Flow patterns around heart valves: A numerical method. J Comput Phys 10(2):252–271. https://doi.org/10.1016/0021-9991(72)90065-4
Piomelli U, Balaras E (2002) Wall-layer models for large-eddy simulations. Ann Rev Fluid Mech 34(1):349–374
Piperno S, Farhat C (2001) Partitioned procedures for the transient solution of coupled aeroelastic problems - part II: Energy transfer analysis and three-dimensional applications. Comput Methods Appl Mech Eng 190(24–25):3147–3170
Potts J, Crowther W (2001) Flight control of a spin stabilised axisymmetric disc-wing. In 39th aerospace sciences meeting and exhibit, p 253
Potts J, Crowther W (2002) Frisbee (TM) aerodynamics. In 20th AIAA applied aerodynamics conference, p 3150
Pradeep DS, Hussain F (2004) Effects of boundary condition in numerical simulations of vortex dynamics. J Fluid Mech 516:115–124
Rodriguez I, Borell R, Lehmkuhl O, Segarra CDP, Oliva A (2011) Direct numerical simulation of the flow over a sphere at Re= 3700. J Fluid Mech 679:263–287
Roma AM, Peskin CS, Berger MJ (1999) An Adaptive Version of the Immersed Boundary Method. J Comput Phys 153(2):509–534. https://doi.org/10.1006/jcph.1999.6293
Ryu J, Park SG, Kim B, Sung HJ (2015) Flapping dynamics of an inverted flag in a uniform flow. J Fluids Struct 57
Saad Y (1993) A flexible inner-outer preconditioned GMRES algorithm. SIAM J Sci Comput 14(2):461–469
Seidl V, Muzaferija S, Perić M (1997) Parallel DNS with local grid refinement. Appl Sci Res 59(4):379–394
Seo JH, Mittal R (2011) A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. J Comput Phys 230(19):7347–7363
Shelley MJ, Zhang J (2011) Flapping and bending bodies interacting with fluid flows. Ann Rev Fluid Mech 43:449–465
Sotiropoulos F, Yang X (2014) Immersed boundary methods for simulating fluid-structure interaction. Prog Aerosp Sci 65:1–21. https://doi.org/10.1016/j.paerosci.2013.09.003
Stein DB, Guy RD, Thomases B (2017) Immersed Boundary Smooth Extension (IBSE): A high-order method for solving incompressible flows in arbitrary smooth domains. J Comput Phys 335:155–178
Taira K, Colonius T (2007) The immersed boundary method: A projection approach. J Comput Phys 225(2):2118–2137. https://doi.org/10.1016/j.jcp.2007.03.005
Taneda S (1968) Waving motions of flags. J Phys Soc Jpn 24(2):392–401
Tezduyar TE, Behr M, Mittal S, Liou J (1992) New strategy for finite element computations involving moving boundaries and interfaces. The deforming-spatial domain/space-time procedure. II. Computation of free surface flows, two-liquid flows, and flows with drifting cylinders. Comput Methods Appl Mech Eng 94(3):353–371
Tezduyar TE, Sathe S, Keedy R, Stein K (2006) Space-time finite element techniques for computation of fluid-structure interactions. Comput Methods Appl Mech Eng 195(17–18):2002–2027

Thompson JF, Soni BK, Weatherill NP (1998) Handbook of grid generation. CRC Press. p 1136 https://doi.org/10.1201/9781420050349
Tian X, Xiao L, Zhang X, Yang J, Tao L, Yang D (2017) Flow around an oscillating circular disk at low to moderate Reynolds numbers. J Fluid Mech 812:1119–1145
Tornberg AK, Engquist B (2004) Numerical approximations of singular source terms in differential equations. J Comput Phys 200(2):462–488
Tsynkov SV (1998) Numerical solution of problems on unbounded domains. A review. Appl Numer Mathematics 27(4):465–532. https://doi.org/10.1016/S0168-9274(98)00025-7
Uhlmann M (2005) An immersed boundary method with direct forcing for the simulation of particulate flows. J Comput Phys 209(2):448–476
Vanella M, Posa A, Balaras E (2014) Adaptive mesh refinement for immersed boundary methods. J Fluids Eng 136(4):040909
Wang S, Zhang X (2011) An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows. J Comput Phys 230(9):3479–3499
Yang J, Balaras E (2006) An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. J Comput Phys 215(1):12–40
Yang X, Zhang X, Li Z, He GW (2009) A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations. J Comput Phys 228(20):7821–7836
Ye T, Mittal R, Udaykumar H, Shyy W (1999) An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. J Comput Phys 156(2):209–240
You D, Wang M, Moin P, Mittal R (2007) Large-eddy simulation analysis of mechanisms for viscous losses in a turbomachinery tip-clearance flow. J Fluid Mech 586:177–204
Yun G, Kim D, Choi H (2006) Vortical structures behind a sphere at subcritical Reynolds numbers. Phys Fluids 18(1):015102. https://doi.org/10.1063/1.2166454
Zahedi S, Tornberg AK (2010) Delta function approximations in level set methods by distance function extension. J Comput Phys 229:2199–2219
Zhang N, Zheng ZC (2007) An improved direct-forcing immersed-boundary method for finite difference applications. J Comput Phys 221(1):250–268
Zhang X, Schmidt D, Perot B (2002) Accuracy and conservation properties of a three-dimensional unstructured staggered mesh scheme for fluid dynamics. J Comput Phys 175(2):764–791

Chapter 2
Direct Lagrangian Forcing Methods Based on Moving Least Squares

Marcos Vanella and Elias Balaras

2.1 Introduction

Immersed boundary methods have emerged as an option to represent a stationary or moving body in a fluid flow simulation, where the primitive variables of the flow (pressure and velocity in case of incompressible flow) are evolved using fast solvers on logically Cartesian grids. A common feature of all immersed boundary methods is that grid lines need not be aligned with body surfaces. The effect of the boundaries on the fluid is approximated by a forcing function or local velocity reconstructions. Their efficiency has been particularly exploited over the years in cases of moving bodies and fluid–structure interactions (FSI). The representation of an immersed body is usually done using unstructured surface meshes. The objective of all immersed boundary variants is to approximate the no-slip boundary condition on the immersed surface. They either assume elastic or viscoelastic properties on incompressible solids, and derive their equations of motion in a continuum setting (Peskin 1972, 1977; Goldstein et al. 1993; Peskin 2003; Griffith et al. 2007), or impose kinematic constraints on the surrounding Eulerian velocity collocation points (direct Eulerian forcing; Mohd-Yusof 1997; Fadlun et al. 2000; Balaras 2004; Mittal and Iaccarino 2005), or the surface control points themselves (direct Lagrangian forcing; Uhlmann 2005; Vanella and Balaras 2009; Pinelli et al. 2010; Kempe and Frohlich 2012; Krishnan et al. 2017).

The specified forces to impose boundary conditions are computed directly from the momentum equations or can be obtained using Lagrange multipliers on the set of equations governing the FSI problem (Glowinski et al. 1999, 2000; Colonius and

M. Vanella
National Institute for Standards and Technology, Gaithersburg, MD, USA
e-mail: marcos.vanella@nist.gov

E. Balaras (✉)
George Washington University, Washington, DC, USA
e-mail: balaras@gwu.edu

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_2

Taira 2008). In general, a surface control (marker) point, l, is related to a set Eulerian grid points, e_{IJ}, as shown in Fig. 2.1. The body surface Γ_s divides the solid region Ω_s from the fluid region Ω_f. Depending on the implementation, marker velocity and location are compared to their Eulerian counterparts and used to define a suitable force density or velocity to be imposed on the stencil points. The immersed boundary formulation proposed by Peskin (1977, 2003) considers the principle of least action (virtual work for conservative systems) for a continuous elastic material subject to the incompressibility constraint. The material is defined in curvilinear coordinates (ξ, η, ψ) and assumed elastic in a subset of these. The equations of motion are derived in Eulerian coordinates considering constrained virtual velocities. It is important to note that the elastic body motion is described in terms of an Eulerian velocity field:

$$\rho_s(\mathbf{x}, t)\frac{D\mathbf{u}(\mathbf{x}, t)}{Dt} = \mathbf{f}(\mathbf{x}, t) \tag{2.1}$$

where ρ_s is the solid's density, $D\mathbf{u}/Dt$ is the material time derivative of the solid's spatial velocity field and $\mathbf{f}$ is the elastic force density, which includes the pressure gradient term and tangential stresses along the curvilinear directions; the latter are functions of shearing strains, dependent on the displacement field. The incompressible viscous flow equations have analogous form to Eq. (2.1), with the addition of a viscous term, $\mu\Delta\mathbf{u}$, where μ is the fluid viscosity. In these equations, the Eulerian quantities ρ_s and $\mathbf{f}$ are related to their material counterpart employing the definition of the Dirac delta function:

$$\rho_s(\mathbf{x}, t) = \int M(\xi, \eta, \psi)\delta(\mathbf{x} - \mathbf{X}(\xi, \eta, \psi, t))\mathrm{d}\xi\,\mathrm{d}\eta\,\mathrm{d}\psi \tag{2.2}$$

$$\mathbf{f}(\mathbf{x}, t) = \int \mathbf{F}(\xi, \eta, \psi, t)\delta(\mathbf{x} - \mathbf{X}(\xi, \eta, \psi, t))\mathrm{d}\xi\,\mathrm{d}\eta\,\mathrm{d}\psi \tag{2.3}$$

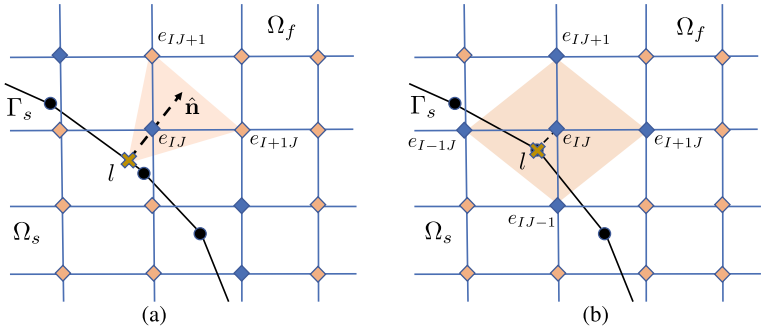


Fig. 2.1 Lagrangian and Eulerian meshes: fluid Ω_f and solid Ω_s regions divided by discretized body surface Γ_s. **a** Example support domain in direct Eulerian forcing. Forcing point e_{IJ} lays in normal direction to surface point l. **b** Example support domain for direct Lagrangian forcing, a force density is computed in marker l and redistributed to the associated Eulerian stencil

where M and $\mathbf{F}$ are the Lagrangian counterparts to ρ_s and $\mathbf{f}$. In addition,

$$\frac{\partial \mathbf{X}}{\partial t} = \mathbf{U}(\mathbf{X}, t) = \int \mathbf{u}(\mathbf{x}, t)\delta(\mathbf{x} - \mathbf{X}(\xi, \eta, \psi, t))\mathrm{d}\mathbf{x}. \tag{2.4}$$

Equations (2.2)–(2.4) provide strict mathematical means to transfer the density, force and velocity fields between Eulerian and material descriptions. All integrals make use of the same kernel function, $\delta(\mathbf{x} - \mathbf{X})$. This formulation assumes that the incompressible solid material is confined by a closed manifold surface within the three-dimensional space, which depends on a set of surface coordinates (ξ, η). Then, a set of sufficiently dense markers is used together with this discrete kernel to *interpolate* the velocities defined by Eq. (2.4) from the Eulerian stencil to each marker, or to *spread* back to the Eulerian stencil the resulting mass and force densities in Eqs. (2.2) and (2.3). In numerical simulations, the continuum equations are discretized, and interaction among Lagrangian and Eulerian grids has to be defined by the use of a smooth approximation to the Dirac delta function δ_h. The choice of this approximation is constrained by the need to preserve mass, force and moments among the two sets of grids, while $\delta_h(\mathbf{x}) \to \delta(\mathbf{x})$ as $h \to 0$. A common definition of δ_h is:

$$\delta_h(\mathbf{x}) = p_c\ \Phi\left(\frac{x_1}{h_x}, \frac{x_2}{h_y}, \frac{x_3}{h_z}\right), \tag{2.5}$$

where p_c is a normalization factor and Φ is a function of the coordinates x_1, x_2, x_3 of point $\mathbf{x}$ and Eulerian cell sizes h_x, h_y, h_z. In several implementations, the expression of Φ has been further simplified:

$$\Phi\left(\frac{x_1}{h_x}, \frac{x_2}{h_y}, \frac{x_3}{h_z}\right) = \phi\left(\frac{x_1}{h_x}\right)\phi\left(\frac{x_2}{h_y}\right)\phi\left(\frac{x_3}{h_z}\right), \tag{2.6}$$

effectively reducing the interpolation to the dimension by dimension application of a scalar function $\phi(\varsigma)$. The properties of ϕ are found from the constraints previously named for δ_h. The interpolation function $\phi(\varsigma)$ is assumed continuous, and must have partition of unity and zero first moment properties to conserve momentum and torque transfer (Peskin 2003). The method described above requires the definition of elastic properties for an immersed solid, which can be problematic when used in rigid body dynamical systems (Balaras 2004; Mittal and Iaccarino 2005; Lai and Peskin 2000).

Alternatively one can consider directly the *discretized* momentum equations and introduce forcing such that the immersed boundary no-slip and no-penetration condition is approximated (Mohd-Yusof 1997; Fadlun et al. 2000). Consider, for example, the two-dimensional setting of Fig. 2.1a. A forward Euler time discretization from time step n to $n + 1$ gives:

$$\frac{\mathbf{u}_{IJ}^{d} - \mathbf{u}_{IJ}^{n}}{\Delta t} = \text{rhs}\left(\mathbf{u}^{n}, p^{n}\right) + \mathbf{f}_{IJ} \tag{2.7}$$

$$\mathbf{f}_{IJ} = \frac{\mathbf{u}_{IJ}^{d} - \widetilde{\mathbf{u}}_{IJ}}{\Delta t}, \quad \widetilde{\mathbf{u}}_{IJ} = \mathbf{u}_{IJ}^{n} + \Delta t \ \text{rhs}\left(\mathbf{u}^{n}, p^{n}\right) \tag{2.8}$$

where u_{IJ}^{d} is the velocity prescribed on an Eulerian point e_{IJ} at the time level $n+1$. The explicit rhs contains viscous, advective and pressure gradient terms, and $\mathbf{f}_{IJ}$ is the force density required to approximate $\mathbf{u}_{IJ}^{n+1}$ with $\mathbf{u}_{IJ}^{d}$. Also, $\widetilde{\mathbf{u}}_{IJ}$ is the intermediate velocity field used in fractional step or pressure-correction methods (Perot 1993; Armfield and Street 2002). Note that $\mathbf{f}_{IJ}$ can be computed directly from Eq. (2.7), only when all terms are treated explicitly. When no wall-modeling is employed, the velocity $\mathbf{u}_{IJ}^{d}$ is generally interpolated from a stencil utilizing points on the immersed body surface, $\mathbf{u}_{l}$, and external Eulerian points (*fluid* points):

$$\mathbf{u}_{IJ}^{d} = \phi_{l}^{d}\,\mathbf{u}_{l} + \sum_{\alpha,\beta} \phi_{I+\alpha,J+\beta}\,\mathbf{u}_{I+\alpha,J+\beta}^{n+1} \tag{2.9}$$

where the subscript l and indexes $\alpha, \beta \subset \{0, \pm 1, \pm 2, \ldots\}$ s.t. $|\alpha| + |\beta| > 0$ identify the Lagrangian marker and external fluid points that compose the stencil. ϕ_{l}^{d}, $\phi_{I+\alpha,J+\beta}$ are interpolation functions. Note that in fractional step methods, the velocities $\mathbf{u}_{I+\alpha,J+\beta}^{n+1}$ on the fluid stencil are not known a priori when defining the force field $\mathbf{f}_{IJ}$, so in general intermediate velocities are used. In FSI problems, $\mathbf{u}_{l}$ is computed from the equations governing the dynamics of the body (i.e., Hou et al. 2012; Yang et al. 2008; Vanella et al. 2010). The Eulerian grid points where the velocity (and in some cases pressure) field is reconstructed can be defined immediately outside the object as shown in Fig. 2.1a (Fadlun et al. 2000; Balaras 2004; Yang and Balaras 2006), inside the object (Mittal et al. 2008), or both (Yang and Balaras 2006; Luo et al. 2012). Reviews on Eulerian forcing methods can be found in Mittal and Iaccarino (2005) and Hou et al. (2012).

Lagrangian direct-forcing schemes have also been proposed (Uhlmann 2005; Vanella and Balaras 2009; Pinelli et al. 2010). Here, the direct-forcing function is computed on Lagrangian markers, rather than on the Eulerian grid nodes. Equation (2.8) is now defined in material coordinates:

$$\mathbf{F}_{l} = \frac{\mathbf{U}_{l}^{d} - \widetilde{\mathbf{U}}_{l}}{\Delta t}, \quad \widetilde{\mathbf{U}}_{l} = \mathbf{U}_{l}^{n} + \Delta t \ \text{RHS}(\mathbf{u}^{n}, p^{n}), \tag{2.10}$$

where the upper case symbols denote the same variables as in Eq. (2.8), but at the surface marker points. $\widetilde{\mathbf{U}}_{l}$ is the Lagrangian counterpart of the intermediate velocity $\widetilde{\mathbf{u}}_{IJ}$. The force density $\mathbf{F}_{l}$ plays the role of a penalization force similar to the elastic force density (Eq. 2.3) in Peskin's formulation (Peskin 1977, 2003). However, its derivation is based on a kinematic constraint given by the no-slip condition on the solid boundary. Discrete transfer operators I_{h}, T_{h} for interpolation and spreading are also required:

$$\widetilde{\mathbf{U}}_l = I_h(\widetilde{\mathbf{u}}_{I+\alpha,J+\beta}) \tag{2.11}$$

$$\mathbf{f}_{I+\alpha,J+\beta} = T_h(\mathbf{F}_l), \tag{2.12}$$

where index pairs α and β refer to the Eulerian interpolation stencil related to each marker l. In Fig. 2.1b, a Lagrangian marker l associated with a sample five-point kernel stencil is shown. The closest point e_{IJ} to l is used to define the stencil. The requirements on these operators are similar to the continuum case: I_h should be mean preserving and accurate to the order of the spatial discretization, and ideally volume-conserving; T_h should be smooth, force and torque preserving, have compact support and low transpiration across the body surface. Assuming the interpolation stencil is composed of *ne* Eulerian grid points, then the interpolation operator, I_h, for the scalar field, v, can be defined by the sums:

$$V_l = \sum_{k=1}^{\text{ne}} \phi_k^l(\mathbf{x}_k, \mathbf{X}_l) v_k, \tag{2.13}$$

where l refers to a Lagrangian surface marker, v^k are the values of v on the stencil points ($k \rightarrow IJ$ in Fig. 2.1) and the ϕ_k^l are a set of interpolation functions. The spreading operator T_h for a field f from a set of Lagrangian markers to an Eulerian point k can also be defined as:

$$f_k = \sum_{l=1}^{\text{nl}} \varphi_k^l(\mathbf{x}_k, \mathbf{X}_l) F_l, \tag{2.14}$$

where the $\varphi_k^l(\mathbf{x}_k, \mathbf{X}_l)$ are a set of spreading coefficients, and nl is the number of Lagrangian markers related to the grid point k. The discrete transfer operations given by Eqs. (2.13) and (2.14) should be constructed in a way that the requirements for I_h, T_h discussed above are satisfied. To this end, several schemes have been proposed on the literature. Uhlmann (2005) used the regularized delta functions δ_h introduced in Peskin (1972) as kernels for interpolation and spreading of variables between the Eulerian and Lagrangian grids. The overall implementation was targeted to rigid spherical particle suspensions in a laminar and turbulent flows. Other options for building interpolation and transfer functions are based on meshless techniques and scattered data approximation (Li and Liu 2004; Liu and Gu 2005). These include the use of the reproducing kernel particle method (RKPM) (Pinelli et al. 2010), moving least squares (MLS) (Vanella and Balaras 2009; de Tullio and Pascazio 2016; Le and Khoo 2017; Wang et al. 2019) and inverse distance interpolation (ID) (Krishnan et al. 2017). In next section, we describe a methodology to build the transfer functions using moving least squares interpolation in Cartesian grids.

2.2 Defining Lagrangian Forcing Transfer Functions Using Moving Least Squares

2.2.1 Building the Transfer Kernels

We construct our transfer operators using MLS shape functions with compact support (Lancaster and Salkauskas 1981; Liu and Gu 2005). For each Lagrangian marker, we: (i) Identify the closest Eulerian grid node. Referring to Fig. 2.2a, a marker l_a is associated with the grid node x_a, which is in the center of a cell with dimensions h_x and h_y in the x- and y-directions, respectively. Marker l_b is associated with the grid node x_b and so on. Note that more than one Lagrangian markers from the same, or different immersed bodies, can be associated with the same Eulerian grid point. (ii) Define a support domain around each Lagrangian marker, in which the shape functions will be constructed. In our case, the support domain is a rectangular box of size $2H_x \times 2H_y \times 2H_z$ centered at the location of the marker. H_x, H_y and H_z are different for each marker and are proportional to the local Eulerian grid. We use $H_x = 1.2h_x$, $H_y = 1.2h_y$ and $H_z = 1.2h_z$ found to be a good compromise on structured grids (Vanella and Balaras 2009). (iii) Associate a volume, $\Delta V^l = A^l h^l$ (A^l is the area of the body surface associated with marker l, and h^l is a local thickness to be defined) to each marker point.

In Fig. 2.2a, the associated volumes ΔV^{l_a} and ΔV^{l_b} for markers l_a and l_b are shown. There is no overlapping between successive volumes, ΔV^l, and the sum of all local A^l is equal to the total area of the immersed object surface. We define the

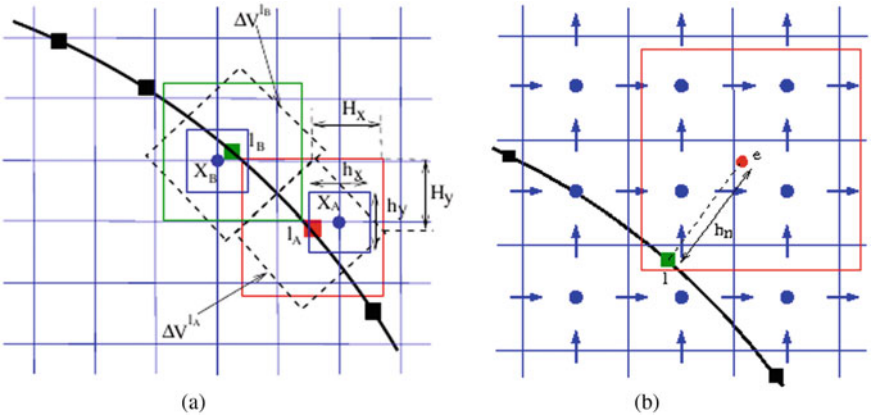


Fig. 2.2 **a** Definition of the support domain for two neighboring Lagrangian markers, l_A and l_B, which are color coded for clarity. X_A and X_B denote the closest Eulerian nodes to l_A and l_B, respectively. The corresponding volumes ΔV are also shown (dashed line). **b** The normal probe defined by the Lagrangian marker l and point, e is shown together with the support domain used in the MLS approximation. Figure reprinted with permission from Vanella and Balaras (2009)

transfer operator to interpolate the component $\widetilde{U}_i^l$ $(i = 1, 2, 3)$ of velocity $\widetilde{\mathbf{U}}_l$, from corresponding velocity components $\tilde{u}_i^k$ of the stencil $\widetilde{\mathbf{u}}_{I+\alpha,J+\beta}$, given by Eq. (2.11). Using the MLS method, $\widetilde{U}_i^l$ for each Lagrangian marker, l, can be approximated in its support domain as follows:

$$\widetilde{U}_i^l(\mathbf{x}) = \sum_{j=1}^{m} p_j(\mathbf{x}) a_j(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{a}(\mathbf{x}), \tag{2.15}$$

where $\mathbf{p}^T(\mathbf{x})$ is the basis functions vector of length m, $\mathbf{a}(\mathbf{x})$ is a vector of coefficients and $\mathbf{x} = (x, y, z)$ is a position within the interpolation stencil region (we are interested in the position of the Lagrangian marker $\mathbf{X}^l$). We use a linear basis, $\mathbf{p}^T(\mathbf{x}) = [\,1\ x\ y\ z\,]$, which is a cost-efficient choice and represents the field variation at minimum linearly. To obtain the coefficient vector, $\mathbf{a}(\mathbf{x})$, the following weighted $L2$-norm is defined:

$$J = \sum_{k=1}^{\text{ne}} W\left(\mathbf{x} - \mathbf{x}^k\right)\left[\mathbf{p}^T(\mathbf{x}^k)\mathbf{a}(\mathbf{x}) - \tilde{u}_i^k\right]^2, \tag{2.16}$$

where $\mathbf{x}^k$ is the position vector of the Eulerian point k in the stencil, $\tilde{u}_i^k$ is the intermediate velocity in direction i for grid point k and $W(\mathbf{x} - \mathbf{x}^k)$ is a weight function defined below. ne is the total number of grid points in the interpolation stencil, which for the linear basis, involves five and seven points in two- and three-dimensions, respectively. We set the closest point to the Lagrangian marker to be the center point in the stencil. Minimizing J with respect to $\mathbf{a}(\mathbf{x})$ leads to:

$$\begin{aligned}
\mathbf{A}(\mathbf{x})\ \mathbf{a}(\mathbf{x}) &= \mathbf{B}(\mathbf{x})\ \tilde{\mathbf{u}}_i^k \quad \text{with,} \\
\mathbf{A}(\mathbf{x}) &= \sum_{k=1}^{\text{ne}} W(\mathbf{x} - \mathbf{x}^k)\mathbf{p}(\mathbf{x}^k)\mathbf{p}^T(\mathbf{x}^k), \\
\mathbf{B}(\mathbf{x}) &= [W(\mathbf{x} - \mathbf{x}^1)\mathbf{p}(\mathbf{x}^1) \quad \cdots \quad W(\mathbf{x} - \mathbf{x}^{\text{ne}})\mathbf{p}(\mathbf{x}^{\text{ne}})], \quad \text{and} \\
\tilde{\mathbf{u}}_i^k &= [\tilde{u}_i^1 \quad \cdots \quad \tilde{u}_i^{\text{ne}}]^T.
\end{aligned} \tag{2.17}$$

The size of matrix $\mathbf{A}(\mathbf{x})$ depends on the size of the basis vector, $\mathbf{p}(\mathbf{x})$, and it is 3×3 in two-dimensions and 4×4 in three-dimensions, while $\mathbf{B}(\mathbf{x})$ is of size $3 \times$ ne in two-dimensions or $4 \times$ ne in three-dimensions. Combining Eqs. (2.15) and (2.17) one can write $\widetilde{U}_i^l$ as follows:

$$\widetilde{U}_i^l(\mathbf{x}) = \sum_{k=1}^{\text{ne}} \phi_k^l(\mathbf{x})\tilde{u}_i^k = \mathbf{\Phi}^T(\mathbf{x})\tilde{\mathbf{u}}_i^k \tag{2.18}$$

where $\mathbf{\Phi}(\mathbf{x}) = \mathbf{p}(\mathbf{x})\ \mathbf{A}(\mathbf{x})^{-1}\ \mathbf{B}(\mathbf{x})$ is a column vector with length ne, containing the shape function values for marker point l. Cubic splines are used for the weight functions, $W(\mathbf{x} - \mathbf{x}^k)$, above, which can be written as:

$$W(\mathbf{x}-\mathbf{x}^k) = W_s(x-x^k)W_s(y-y^k)W_s(z-z^k) \tag{2.19}$$

$$W_s(x_i-x_i^k) = \begin{cases} 2/3 - 4\overline{r_k}^2 + 4\overline{r_k}^3 & \text{for} \quad \overline{r_k} \leq 0.5 \\ 4/3 - 4\overline{r_k} + 4\overline{r_k}^2 - 4/3\overline{r_k}^3 & \text{for} \quad 0.5 \leq \overline{r_k} \leq 1.0 \\ 0 & \text{for} \quad \overline{r_k} > 1.0 \end{cases} \tag{2.20}$$

where $\overline{r_k} = |x_i - x_i^k|/H_i$, and $x_i = x, y, z$. These functions are monotonically decreasing and are sufficiently smooth in the support domain. The resulting shape functions reproduce exactly the linear polynomial contained in their basis and possess the partition of unity property $\sum_{i=1}^{\text{ne}} \phi_i(\mathbf{x}) = 1$ (Liu and Gu 2005). Also, the field approximation is continuous on the global domain as the *MLS* shape functions are compatible. Note that in practice, to avoid ill conditioning of the matrix $\mathbf{A}(\mathbf{x})$, the origin of the coordinate system is shifted to the location of the Lagrangian marker $\mathbf{X}^l$ (Liu and Gu 2005). Equation (2.18) gives $\widetilde{U}_i^l$, which can be substituted into the left of Eq. (2.10) to obtain the volume force F_i on Lagrangian markers. To transfer F_i to the Eulerian points associated with each marker l, the same shape functions ϕ_k^l are used properly scaled by a factor c_l, determined later. The forces on the Eulerian grid nodes are:

$$f_i^k = \sum_{l=1}^{\text{nl}} c_l \phi_k^l F_i^l, \tag{2.21}$$

where f_i^k is the volume force in the Eulerian point k, direction i, ϕ_k^l is the shape function relating variables between grid point k and marker l, and F_i^l is the force in marker l. To properly rescale the shape functions, we require that the total force acting on the fluid is not changed by the transfer:

$$\sum_{k=1}^{\text{nte}} f_i^k \Delta V^k = \sum_{l=1}^{\text{ntl}} F_i^l \Delta V^l \tag{2.22}$$

where $\Delta V^k = (h_x \times h_y \times h_z)$ is the volume associated with the Eulerian grid point k, and $\Delta V^l = A^l h^l$ is the volume associated with the marker l, with $h^l = 1/3 \sum_{k=1}^{\text{ne}} \phi_k^l (h_x + h_y + h_z)$. nte and ntl are the total number of forced grid points and total number of Lagrangian markers, respectively. Also, the area A^l for marker l is obtained by a simple angle averaging process. Substituting Eq. (2.21) into Eq. (2.22) and rearranging the sums in the left hand side in terms of the total number of markers we get:

$$\sum_{l=1}^{\text{ntl}} \sum_{k=1}^{\text{ne}} \phi_k^l \Delta V^k c_l F_i^l = \sum_{l=1}^{\text{ntl}} \Delta V^{\text{El}} c_l F_i^l = \sum_{l=1}^{\text{ntl}} F_i^l \Delta V^l \tag{2.23}$$

where ΔV^{El} is the averaged Eulerian grid volume associated with the Lagrangian marker l. For Eq. (2.23), to hold the scaling factor c_l needs to be set to:

$$c_l = \frac{\Delta V^l}{\Delta V^{\mathrm{EI}}} = \frac{A^l h^l}{\Delta V^{\mathrm{EI}}}, \tag{2.24}$$

One can also show that the above scheme guarantees the equivalence of total torque between the Eulerian and Lagrangian meshes (Vanella and Balaras 2009).

The computational cost of building the transfer functions ϕ_k^l is mainly related to the inversion of matrix $\mathbf{A}(\mathbf{x})$. Strategies to speedup this step have been recently presented in the literature (Li et al. 2015; Spandan et al. 2018). It should be noted that the selected stencil size will define the forcing band size and the *sharpness* of the boundary representation. Lagrangian forcing methods have in general first-order overall asymptotic accuracy (Li and Ito 2006), although numerical evidence of higher accuracy in selected tests problems has been reported in the literature for MLS transfer kernels (Vanella and Balaras 2009; Li et al. 2015; de Tullio and Pascazio 2016; Krishnan et al. 2017), as well as the ID interpolation scheme (Krishnan et al. 2017). It is important to note that the immersed boundary force constructed does not span the interior of solids. This is computationally efficient (no need to know which cells are inside and which are outside of a complex volume, and force interior cells), specially in moving object problems and FSI. This may result, however, in higher flow penetration into the object than seen in Eulerian forcing methods, but strategies to minimize it can be found in the literature (Kempe and Frohlich 2012). Another important consideration is the marker density on the solid surface, which has to be such that continuity of the forcing band is assured. This is not trivial to achieve, in particular, in cases where the body is allowed to translate across different Eulerian grid resolutions. In Sect. 2.3, we will discuss this issue in detail and present a method to perform Lagrangian mesh adaptivity.

2.2.2 *Estimating Surface Forces from Linearized Fields*

The local hydrodynamic force per unit area on a surface element,

$$f_i^H = \tau_{ji} n_j = \left[-p\delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] n_j, \tag{2.25}$$

is computed directly from the flow field around the body (where f_i^H is the hydrodynamic surface force in the x_i direction, τ_{ji} is stress tensor, and n_j is the direction cosine of the normal unit vector in x_j direction). The use of Eq. (2.25) requires estimating p and $\partial u_i/\partial x_j$ on the body surface. For each Lagrangian marker, l, on the body we create a normal probe by locating an external point, e, at distance h_n from the surface (see Fig. 2.2b). The distance h_n can be set to: $h_n = \left(h_x + h_y + h_z\right)/3$. To compute the surface pressure at marker, l, we first compute the pressure, p^e, at point e, using the MLS interpolation described previously. The support domain in this case is centered around point e as shown in Fig. 2.2b. Next, the value of $\partial p/\partial n$ is obtained from the momentum equation normal to the boundary (Yang and Balaras 2006):

$$\frac{\partial p}{\partial n} = -\frac{D\mathbf{u}}{Dt} \cdot \mathbf{n}, \tag{2.26}$$

where $\mathbf{n}$ is the normal unit vector passing through the marker l, and $\frac{D\mathbf{u}}{Dt}$ is the acceleration of the marker. The value of the pressure at the surface is then obtained from:

$$p^l = p^e - \frac{\partial p}{\partial n} h_n \tag{2.27}$$

The velocity derivatives, $\partial U_i/\partial x_j$, at the location e for each Lagrangian marker, l, are computed by differentiating Eq. (2.18):

$$\frac{\partial U_i}{\partial x_j} = \sum_{k=1}^{\text{ne}} \frac{\partial \phi_k}{\partial x_j} u_i, \tag{2.28}$$

where $\partial \phi_k/\partial x_j$ comes from the solution of an additional system of equations similar to (2.17) (see Liu and Gu 2005). As h_n is of the order of the local grid size, and assuming a linear velocity variation near the body, the derivatives, $\partial U_i/\partial x_j$, defined in Eq. (2.28) are good approximations for the derivatives $\partial u_i/\partial x_j$, at the surface. As we will demonstrate in the example below, the *normal probe* approach utilizing Eq. (2.25) results in accurate prediction of the surface forces when the grid resolution is sufficiently high to resolve the local velocity gradients near the immersed boundary. For high Reynolds numbers, where this condition cannot be satisfied due to cost considerations, a model-based enhancement strategy is discussed in Sect. 2.4.

2.2.3 Example: Oscillating Cylinder in a Cross-Flow

In this example, we consider the case of a transversely oscillating cylinder in a cross-flow. The dominant parameters are the Reynolds number $\text{Re} = U_\infty D/\nu$ (U_∞ is the inflow velocity), forcing frequency, f_e and amplitude, a_0 of the oscillation. When f_e varies around the natural shedding frequency, f_0, interesting phenomena occur due to the complex energy transfer between the fluid and the body (Gu et al. 1994; Guilmineau and Queutey 2002). Capturing the detailed flow physics for this problem requires an accurate reproduction of the vorticity dynamics on the cylinder surface and is a stringent test for non-boundary-conforming schemes. We use the parametric space from experiments (Gu et al. 1994), boundary-conforming simulations (Guilmineau and Queutey 2002) and computations using an Eulerian direct-forcing scheme (Yang and Balaras 2006). The motion of the cylinder is given by $y(t) = a_0 \sin(2\pi f_e t)$. We consider three cases with $\text{Re} = 185$, $a_0 = 0.2D$ and $f_e/f_0 = 1.0, 1.1, 1.2$, respectively. For all cases the computational domain was set to $50D \times 30D$ in the streamwise and cross-stream directions, respectively; the cylinder located at $10D$ from the inflow. Free-slip conditions are used at the freestream boundaries and a convective condition at outflow boundary (Orlanski 1976). We used

two stretched grids with resulting cell sizes around the cylinder $\Delta x = \Delta y = 0.008D$ and $\Delta x = \Delta y = 0.004D$. A series of tests for flow over a stationary cylinder was first conducted to examine the sensitivity of the results to the grid resolution. Mean and root-mean-square (rms) values of the drag and lift coefficients on the fine grid were $\overline{C}_{\mathrm{D}} = 1.377$, $C_{\mathrm{D}}^{\mathrm{rms}} = 0.0296$ and $C_{\mathrm{L}}^{\mathrm{rms}} = 0.461$, and the corresponding values on the coarser grid are within 1.5% of the above, showing grid independence of the results.

The temporal evolution of the lift and drag coefficients for the case of the oscillating cylinder are shown in Fig. 2.3a, b. Note the scheme gives a smooth variation of the force coefficients. In Fig. 2.3c, a comparison of $\overline{C}_{\mathrm{D}}$, $C_{\mathrm{D}}^{\mathrm{rms}}$ and $C_{\mathrm{L}}^{\mathrm{rms}}$ for the different excitation frequencies is shown with the corresponding results from the named references. A comparison for the phase angle between the lift and transverse displacement is shown in Fig. 2.3d. The largest discrepancy appears in $\overline{C}_{\mathrm{D}}$ and is of the order of 3.5%. We note that the grid resolution around the cylinder used here is comparable to the one in the reference computations, where $\Delta x \sim 0.005D$.

Next, we assess local forces. In Fig. 2.4, the distributions of surface pressure and skin friction coefficients, C_{p} and C_{f}, are shown for a time level corresponding to the extreme upper position. Results for both grids are included from our computations and are compared with corresponding results from Guilmineau and Queutey (2002)

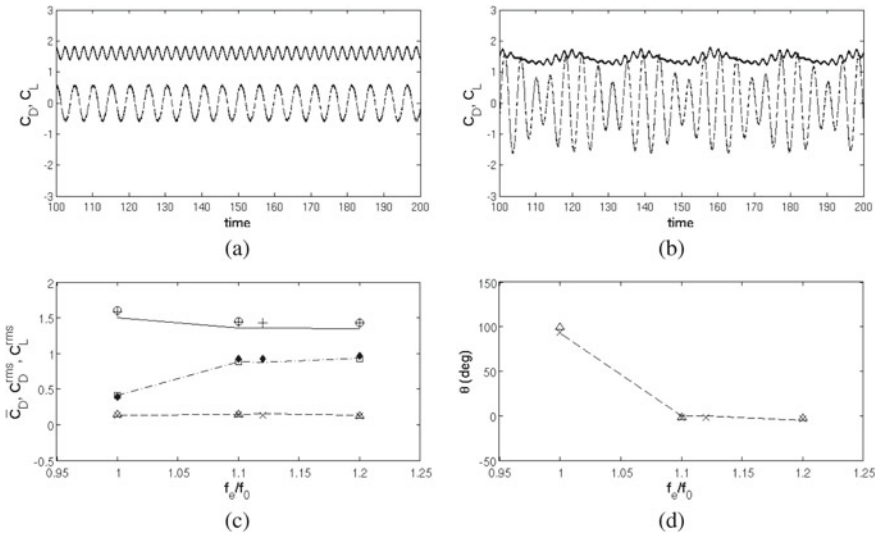


Fig. 2.3 Drag and lift coefficients as a function of time for the case of cylinder oscillating in a cross-flow for **a** $f_e/f_0 = 1.0$, and **b** $f_e/f_0 = 1.2$ (— C_{D} and -- C_{L}). **c** Comparison of force coefficients. ∘ $\overline{C}_{\mathrm{D}}$, △ $C_{\mathrm{D}}^{\mathrm{rms}}$, □ $C_{\mathrm{L}}^{\mathrm{rms}}$ are the present results for the fine grid; — $\overline{C}_{\mathrm{D}}$, -- $C_{\mathrm{D}}^{\mathrm{rms}}$, -·- $C_{\mathrm{L}}^{\mathrm{rms}}$ from Guilmineau and Queutey (2002), and + $\overline{C}_{\mathrm{D}}$, × $C_{\mathrm{D}}^{\mathrm{rms}}$, ♦ $C_{\mathrm{L}}^{\mathrm{rms}}$ from Yang and Balaras (2006). **d** Phase angle between lift force and vertical displacement. △ are the present results on the fine grid; -- Guilmineau and Queutey (2002) and × Yang and Balaras (2006). Figure reprinted with permission from Vanella and Balaras (2009)

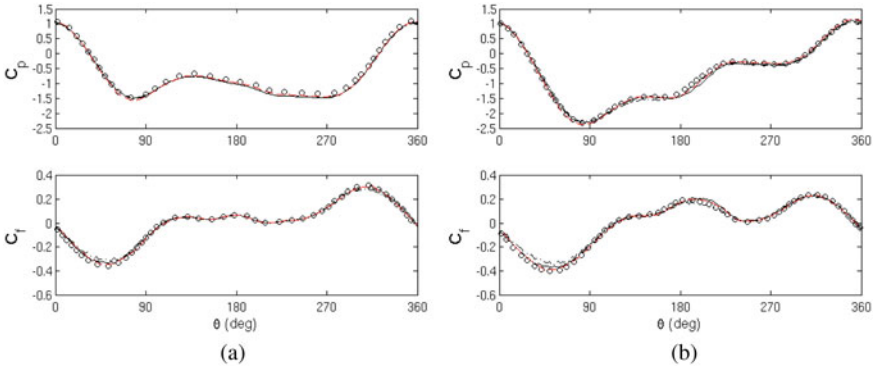


Fig. 2.4 Distribution of the pressure and skin friction coefficients C_p and C_f for the case of a cylinder oscillating in a cross-flow. The cylinder is located at the extreme upper position. -.- present results for $\Delta x = 0.008D$, — present results for $\Delta x = 0.004D$, ∘ body-fitted computations in Guilmineau and Queutey (2002), - - non-boundary-conforming computations in Yang and Balaras (2006). **a** $f_e/f_0 = 1.0$; **b** $f_e/f_0 = 1.2$. Figure reprinted with permission from Vanella and Balaras (2009)

and Yang and Balaras (2006). The higher sensitivity of C_f to grid resolution results in slightly lower peak values on our coarse grid computations. A method for improving shear stresses at the wall will be presented in Sect. 2.4. The finer grid results agree well with the reference data. We note in this and other following examples that C_p is less sensitive to the grid resolution.

2.3 Introducing Lagrangian Mesh Adaptivity

2.3.1 Overview

In the class of methods described above the equations of fluid motion are solved on a fixed Eulerian grid, and the immersed body is represented by a set of surface nodes and elements, defining a Lagrangian grid. The relation between these two grids is important to the accuracy, robustness and efficiency of the method. The criteria, however, for designing either one are usually very different. In the case of the Lagrangian grid, for example, the number and size of surface elements are guided by the complexity and local curvature of the surface, while for the Eulerian grid is guided by velocity and pressure gradients. The sensitivity to the relation between the two grids also depends on the specifics of the forcing scheme. For example, direct-forcing schemes, where the solution is reconstructed around an Eulerian grid node in the neighborhood of the body (i.e., Balaras 2004), are less sensitive compared to schemes where a forcing function is computed at the Lagrangian markers, such as the ones considered here.

In particular, one of the main requirements in all Lagrangian forcing schemes is that the Lagrangian and Eulerian grids need to be of comparable resolution to ensure proper transfer of forces from one grid to the other. An example where the Lagrangian grid is fine enough to maintain the continuity of the forcing band across the Eulerian grid is sketched in Fig. 2.5a, while a case where a breakdown of continuity of the forcing field happens is shown in Fig. 2.5b. The latter scenario is common in practical applications as the Eulerian grid is refined in areas of high-velocity gradients, and Lagrangian grid is refined in areas with high curvature. Even in simple geometrical configurations, it is difficult to enforce both constraints simultaneously. Consider, for example, high Reynolds number flow around a sphere. The boundary layers originating from the stagnation point in the front are very thin, requiring a fine Eulerian grid to resolve them. As the Reynolds number increases, the Eulerian grid needs to be refined accordingly. To ensure that the forcing is properly transferred (no-slip condition correctly approximated), similar refinement needs to be done to the Lagrangian grid, although the geometry is already represented by a sufficient number of elements.

It is, therefore, conceivable that some form of local adaptive refinement/de-refinement of the Lagrangian grid, to satisfy the constraints coming from neighboring Eulerian nodes, will improve the accuracy and robustness of most immersed boundary approaches. An h-refinement strategy, however, where local refinement is achieved by splitting existing surface elements into several smaller ones, or by locally introducing additional nodes, will significantly increase the cost as well as the implementation complexity. A logical option is distributing Lagrangian markers on each triangle surfaces such that their local density is commensurate with the Eulerian grid size. For example, in Spandan et al. (2018), where markers are defined in triangles centroids, local Lagrangian refinement is achieved by subdividing the triangles by splitting their edges. The procedure is dynamic without information storage on the

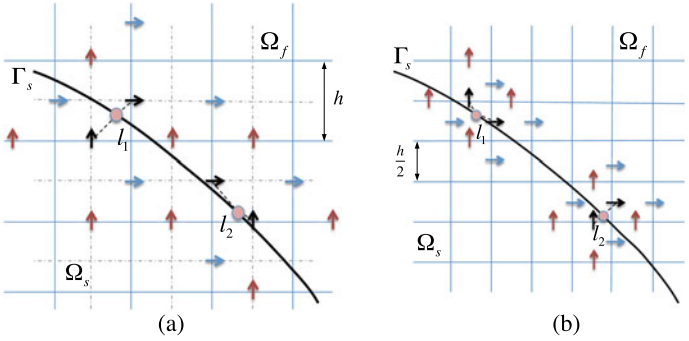


Fig. 2.5 Immersed boundary forcing stencils for single Lagrangian grid and two different staggered Eulerian grids: Ω_f and Ω_s are the fluid and solid regions, respectively, separated by a surface Γ_s. **a** Two marker points l_1, l_2 are sufficient to marginally cover the forcing band on an Eulerian grid of cell size h. **b** Regions with no force imposition (holes) are left for the same Lagrangian markers on a finer Eulerian grid of cell size $h/2$. Figure reprinted with permission from Posa et al. (2017)

refined elements. Likewise, below we will discuss a strategy based on isoparametric surface element mapping, which locally refines the Lagrangian grid by surface element without altering the existing triangulation data structure or increasing storage requirements.

2.3.2 *Dynamic Lagrangian Mapping*

Figure 2.6a shows the surface of a solid body, which is represented by a tessellation, $\Im$, composed of the union of triangular subdomains $\{\varepsilon\} \subset \Im$. The vertex points for this mesh are the set of nodes, $\aleph_\iota$, and the information for the local nodes, n_ι, $\iota = 1, 2, 3$, of every element, ε, is readily known, using a connectivity list typical of finite element data-structures (Hughes 2000; Bathe 2007). Kinematical information of these nodes, $\mathbf{X}^{n_\iota}$, $\mathbf{U}^{n_\iota}$, $\dot{\mathbf{U}}^{n_\iota}$, may be prescribed in time, or driven dynamically by equations of motion of the solid. Linear interpolation functions are used to define the location of marker points $\mathbf{X}^{\varepsilon,l}$ within the triangle ε. A master element of unitary side length in the direction of the natural coordinates, ξ, η (see Fig. 2.6b) can be considered. The interpolation functions for this linear triangle are:

$$N_1 = 1 - \xi - \eta; \quad N_2 = \xi; \quad N_3 = \eta \tag{2.29}$$

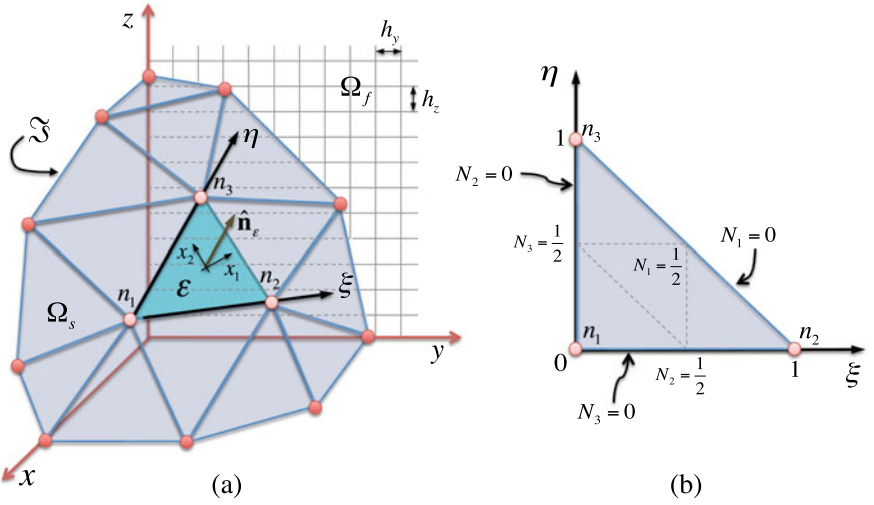


Fig. 2.6 Triangulation $\Im$ representing the IB surface inside a Cartesian Eulerian grid: **a** a particular element ε has vertices n_1, n_2, n_3 in local numbering, defining the triangle geometric quantities (area A_ε, normal $\hat{\mathbf{n}}_\varepsilon$). **b** The natural coordinate system is used to represent the master triangle and map points within it to physical coordinates. The linear shape functions are N_1, N_2, N_3. Figure reprinted with permission from Posa et al. (2017)

The physical location of a particular point l within the triangle, defined by natural coordinates ξ, η is given by,

$$\mathbf{X}^{\varepsilon,l} = \sum_{\iota=1}^{3} N_{\iota}^{\varepsilon,l} \mathbf{X}^{n_\iota} \tag{2.30}$$

The transformation from natural to planar coordinates x_1, x_2 for this linear triangle is characterized by a Jacobian matrix of the form:

$$[\mathbf{J}] = \begin{bmatrix} \frac{\partial x_1}{\partial \xi} & \frac{\partial x_2}{\partial \xi} \\ \frac{\partial x_1}{\partial \eta} & \frac{\partial x_2}{\partial \eta} \end{bmatrix} \tag{2.31}$$

The Jacobian $J = \det(\mathbf{J})$ defines the ratio of physical to natural areas $\mathrm{d}x_1\,\mathrm{d}x_2 = J\,\mathrm{d}\xi\,\mathrm{d}\eta$ and can be used for the area integration on isoparametric elements. Several different mapping methods can be devised to distribute Lagrangian markers within the triangle. A possible choice is illustrated in Fig. 2.7a in natural coordinates. In order to locate the set of points within the triangle, a variable φ defining the distance to corners in ξ and η is used. Here, $\varphi = a\Delta_\xi$, where $\Delta_\xi = 1/m_\xi$. Note that

$$m_\xi = \frac{h_\varepsilon}{c\Delta} + 1. \tag{2.32}$$

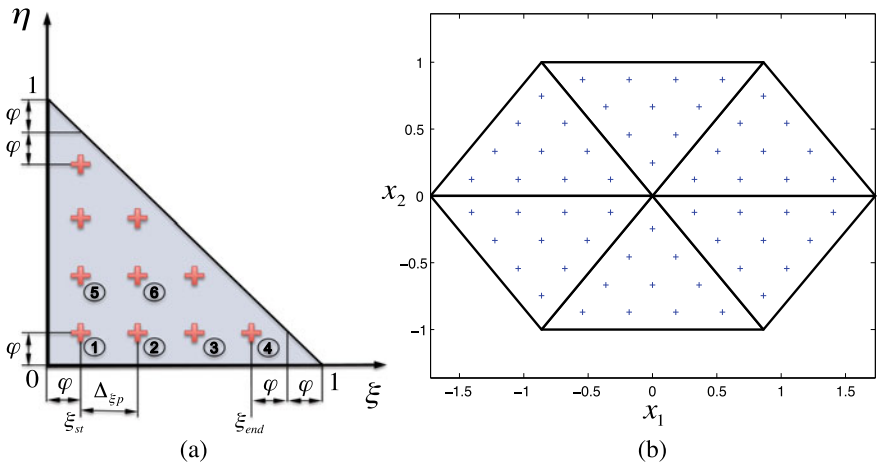


Fig. 2.7 Mapping A, evenly distributed points across the element: **a** distribution of marker points on the master element, the number of markers in the natural η direction N_η is always equal to the N_ξ points on ξ. Positioning and distance variables are the same in both directions. Here, $N_\xi = 4$ for illustration. **b** A regular set of triangles in physical coordinates x_1, x_2 with their corresponding mapped Lagrangian points, again $N_\xi = 4$. Figure reprinted with permission from Posa et al. (2017)

where h_ε is a characteristic length for triangle, ε and Δ are the local characteristic length of the Eulerian grid. The constant $c < 1$ defines the ratio of the distance between markers within the element to the local cell size. The factor a determines the marker location with respect to the triangle corners. A value $a = 1/2$ produces a fairly uniform marker paving on a regular patch of triangles in physical space (see Fig. 2.7b for a two-dimensional case). The method for point distribution is exactly the same for ξ or η. The first line of markers is positioned at a distance $\xi_{\text{st}} = \varphi$ from the origin, and the last marker at a location $\xi_{\text{end}} = 1 - 2\varphi$. The distance between points is obtained as:

$$\Delta_{\xi p} = \frac{\xi_{\text{end}} - \xi_{\text{st}}}{(m_\xi - 1)} = \frac{1 - 3\varphi}{(m_\xi - 1)}. \tag{2.33}$$

As expected $\Delta_{\eta p} = \Delta_{\xi p}$. The paving algorithm consists of a double loop, first in index j (from 1 to m_η) and then in index i (1 to $m_\xi + 1 - j$), using lower triangular limits. The index of a particular point (ε, l) can be obtained by the formula,

$$l = i + (j - 1)m_\xi - \frac{(j - 1)(j - 2)}{2}, \tag{2.34}$$

and the position of the marker in physical coordinates is given by the corresponding interpolation:

$$\xi_i = \varphi + (i - 1)\Delta_{\xi p} \tag{2.35}$$

$$\eta_j = \varphi + (j - 1)\Delta_{\eta p} \tag{2.36}$$

$$\mathbf{X}^{\varepsilon,l} = N_1^{\varepsilon,l}\mathbf{X}^{n_1} + N_2^{\varepsilon,l}\mathbf{X}^{n_2} + N_3^{\varepsilon,l}\mathbf{X}^{n_3}, \tag{2.37}$$

where the shape functions associated with the mapped point are $N_1^{\varepsilon,l} = 1 - \xi_i - \eta_j$, $N_2^{\varepsilon,l} = \xi_i$ and $N_3^{\varepsilon,l} = \eta_j$. Velocities $\mathbf{U}^{\varepsilon,l}$ and accelerations $\dot{\mathbf{U}}^{\varepsilon,l}$ are computed in the same manner. In cases of general rigid body motion, both velocity and acceleration of the markers can also be defined analytically, using for example, the markers location respect to the body center of mass and the rigid kinematics equations.

This mapping scheme facilitates the decoupling of the resolution requirements between the Lagrangian and Eulerian grids. The immersed body is always described by a set of triangles that properly captures the local curvature, but when the size of an element is larger than the local Eulerian grid, then the proper number of Lagrangian markers is introduced to preserve the continuity and smoothness of the forcing field. Most importantly, all the information related to position and kinematics of marker points is not stored in memory, as it can be dynamically recomputed using the mapping processes with a small computational overhead.

2.3.3 Adaptive Reconstruction

The adaptive MLS scheme simply adds an extra step in the process outlined in Sect. 2.2, where interpolation is now modified looping over the ε triangles. The mapping scheme described previously is applied on each triangle to generate data of Lagrangian marker (ε, l) ($\mathbf{X}^{\varepsilon,l}$, $\mathbf{U}^{\varepsilon,l}$, A_ε^l at time level $n+1$). For surface marker (ε, l), interpolate $\tilde{U}_i^{\varepsilon,l}$ from surrounding Eulerian stencils using Eq. (2.11), and one of the methods to construct I_h (Uhlmann 2005; Vanella and Balaras 2009; Pinelli et al. 2010).

Once the position of a given marker (ε, l) is computed and the associated Eulerian point stencil defined, the intermediate velocities can be interpolated to the marker using the following operator:

$$\tilde{U}_i^{\varepsilon,l} = \sum_{k=1}^{ne} \phi_k^{\varepsilon,l} \left(\mathbf{x}^k - \mathbf{X}^{\varepsilon,l}\right) \tilde{u}_i^k \tag{2.38}$$

where $\phi_k^{\varepsilon,l}$ is a set of interpolation functions defined using MLS. Then, Eq. (2.10) is used to compute the forcing field, $F_i^{\varepsilon,l}$, with the interpolated intermediate velocities, $\tilde{U}_i^{\varepsilon,l}$, and the markers velocity components $U_i^{\varepsilon,l}$, $i = 1, 2, 3$, as input. Finally, this forcing field is transferred back to the Eulerian grid:

$$f_i^k = \sum_{(\varepsilon,l)\,\subset\,\mathrm{nl}} c_{\varepsilon,l}\, \phi_k^{\varepsilon,l} \left(\mathbf{x}^k - \mathbf{X}^{\varepsilon,l}\right) F_i^{\varepsilon,l} \tag{2.39}$$

where $c_{\varepsilon,l}$ are scaling factors defined to ensure conservation of total force and torque during transfer, and nl refers to the set of mapped markers related to the Eulerian point k. The coefficient $c_{\varepsilon,l}$ is computed by requiring that:

$$\sum_{k=1}^{\mathrm{nte}} f_i^k \Delta V^k = \sum_{\varepsilon\subset\Im} \sum_{l=1}^{m_\varepsilon} F_i^{\varepsilon,l} \Delta V_\varepsilon^l \tag{2.40}$$

where nte is the total number of Eulerian nodes, m_ε the total number of mapped markers in element ε, $\Delta V^k = (h_x h_y h_z)_k$ the volume of the cell from Eulerian point k and $\Delta V_\varepsilon^l = A_\varepsilon^l h_\varepsilon^l$. As we discussed in Sect. 2.2, the local thickness, h_ε^l, can be defined as follows:

$$h_\varepsilon^l = \sum_{k=1}^{\mathrm{ne}} \phi_k^{\varepsilon,l} \frac{(h_x + h_y + h_z)_k}{3}. \tag{2.41}$$

We should note that the above definition for h_ε^l works well for isotropic grids. In the case of highly anisotropic, stretched grids, however, it can lead to forcing that lacks smoothness. To alleviate this issue, one can define h_ε^l as follows:

$$h_{\varepsilon}^{l} = \frac{\Delta V^{\mathrm{El}}}{\sum_{(\varepsilon,l) \subset \mathrm{nl}} A_{\varepsilon}^{l}}, \tag{2.42}$$

where the denominator is the sum of all areas of the Lagrangian markers associated with the Eulerian node in consideration. Finally, the transfer coefficient, $c_{\varepsilon,l}$ that satisfies Eq. (2.40) can be defined as:

$$c_{\varepsilon,l} = \frac{\Delta V_{\varepsilon}^{l}}{\Delta V^{\mathrm{El}}} = \frac{A_{\varepsilon}^{l} h_{\varepsilon}^{l}}{\Delta V^{\mathrm{El}}}, \tag{2.43}$$

where $\Delta V^{\mathrm{El}} = \sum_{k=1}^{\mathrm{ne}} \phi_{k}^{\varepsilon,l} \Delta V^{k}$.

2.3.4 *Example: Flow Around an Oscillating Sphere*

We will use the oscillating sphere problem to illustrate the use of the adaptive MLS scheme and to stress its importance in maintaining proper marker particle density for accurate IB reconstructions. The sphere has a diameter, D, oscillates in an otherwise quiescent fluid. The vertical motion of the sphere is given by the harmonic function $U(t) = U_o \cos(\omega t)$, where U_o is a reference velocity, and ω is the oscillation angular frequency. The non-dimensional parameters important in this flow case are the Reynolds, $\mathrm{Re} = U_o D/\nu$, and Stokes, $\varepsilon = (\omega R^2/2\nu)^{1/2}$ numbers, where $R = D/2$, and ν is the fluids kinematic viscosity. We consider a case with $\mathrm{Re} = 100$ and $\varepsilon = 5$ (period of oscillation, $T = \pi$). Computations are conducted on uniform grids with 32, 48, 64, 96, 128 cubed cells in the diameter, D, of the sphere. The number of marker particles introduced by the adaptive scheme varied between 9 and 80 K. In Fig. 2.8a, a force comparison between the present computation and the reference results reported in Yang and Balaras (2006) is shown. The agreement is excellent for both the pressure and viscous components for the same grid resolution (64 cells along D). In Fig. 2.8b, the predicted drag force is compared for different resolutions. As expected the viscous component converges slower when compared to the pressure contribution.

2.4 Enhanced Surface Force Estimation

2.4.1 *Overview*

The computation of the hydrodynamic forces on an immersed body is a great challenge, particularly for Lagrangian forcing schemes. In early implementations of this approach for spherical particles, the momentum balance approach was utilized (see Uhlmann 2005; Kempe and Frohlich 2012), where the total force was estimated as

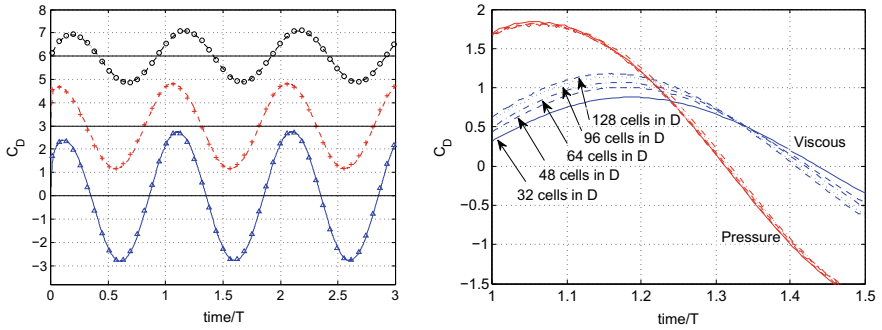


Fig. 2.8 Transversely oscillating sphere at Re = 100, $\varepsilon = 5$. **a** – – – pressure drag; —·— viscous drag; ——— total drag; Symbols are corresponding reference results for the same configuration and grid resolution reported in Yang and Balaras (2006). Note that force coefficient curves are offset by a factor of 3 for clarity. **b** Grid convergence for the adaptive scheme. Figure reprinted with permission from Posa et al. (2017)

the sum of the immersed boundary forcing term and the fluid acceleration inside the solid body:

$$\mathbf{F}_{N,h} = -\int_V \mathbf{f} \mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{u} \mathrm{d}V, \tag{2.44a}$$

$$\mathbf{M}_{N,h} = -\int_V \mathbf{r} \times \mathbf{f} \mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{r} \times \mathbf{u} \mathrm{d}V, \tag{2.44b}$$

where $\mathbf{F}_{N,h}$ and $\mathbf{M}_{N,h}$ are the hydrodynamic force and moment, respectively. V is the volume occupied by the solid body on the Eulerian grid. $\mathbf{f}$ is the immersed boundary forcing term on the Eulerian grid, and $\mathbf{r}$ is the position vector with respect to the object's center of mass. The first term on the right side of Eqs. (2.44a) and (2.44b) is computed by adding the immersed boundary forcing term and its moment on the Lagrangian points, as the transfer functions between Eulerian and Lagrangian grids are designed to preserve the total force and moment. In spherical particle cases, the second term on the RHS of Eq. (2.44a) can be approximated using the acceleration of the center of mass, $\dot{\mathbf{u}}_c$, and particle volume V_c (Uhlmann 2005), $\frac{\mathrm{d}}{\mathrm{d}t}\int_V \mathbf{u} \mathrm{d}V \approx V_c \dot{\mathbf{u}}_c$, or evaluated numerically using the diameter of the sphere to identify the particle volume (Uhlmann 2005; Kempe and Frohlich 2012). For a particle of arbitrary shape, this method requires the identification the Eulerian cells within the domain occupied by the particle and the volume fraction of the boundary cells to correctly compute the integral. Moreover, the use of Eqs. (2.44a) and (2.44b) does not provide the distribution of hydrodynamic forces on the surface, and does not provide sufficient hydrodynamic force information in cases of deformable particles, or when statistics of the hydrodynamic surface stress are needed.

A more general approach to compute the hydrodynamic forces is by direct numerical integration of stresses along the body surface:

$$\mathbf{F}_{N,h} = \int_{\partial V} \tau \cdot \mathbf{n} \mathrm{d}S, \tag{2.45a}$$

$$\mathbf{M}_{N,h} = \int_{\partial V} \mathbf{r} \times (\tau \cdot \mathbf{n}) \mathrm{d}S, \tag{2.45b}$$

where ∂V is the particle surface, $\mathbf{n}$ is the unit vector in the normal direction and $\tau = -p\mathbf{I} + (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/\mathrm{Re}$ is the hydrodynamic stress tensor. This approach can provide both the total hydrodynamic forces and the distribution of the hydrodynamic stress acting on the particle surface. In Sect. 2.2.2, the surface hydrodynamic stresses are computed using the *normal probe* approach, where a linear reconstruction for velocities and pressure along the normal direction is done. This approach requires very fine grids to accurately resolve the velocity gradients within the boundary layer rendering it very expensive when applied to turbulent flows with finite-size particles, for example. Both the above methods should yield similar hydrodynamic forces if the grid is fine enough to resolve the boundary layers, which in most cases comes with a high computational cost. In the following section, we will discuss a strategy to compute the hydrodynamic forces on coarse grids, where the boundary layer approximation is utilized to compensate for the lack of grid resolution.

2.4.2 *Model-Based Enhancement of Surface Force Prediction*

The viscous layer close to the body surface can be approximated by the boundary layer equations in a local orthogonal curvilinear coordinate system, $\xi - \eta$ (see Fig. 2.9):

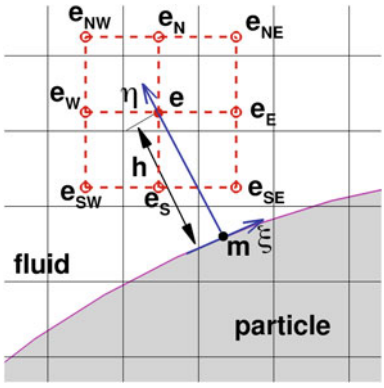


Fig. 2.9 Schematic of the local curvilinear coordinate system and the external point array. Figure reprinted with permission from Wang et al. (2019)

$$\frac{\partial u_\xi}{\partial t} + \frac{u_\xi}{h_\xi}\frac{\partial u_\xi}{\partial \xi} + u_\eta \frac{\partial u_\xi}{\partial \eta} = -\frac{1}{h_\xi}\frac{\partial p}{\partial \xi} + \frac{1}{\mathrm{Re}}\frac{\partial^2 u_\xi}{\partial \eta^2}, \tag{2.46a}$$

$$\frac{\partial h_\xi}{\partial \eta} u_\xi^2 = \frac{\partial p}{\partial \eta}, \tag{2.46b}$$

where p is the pressure, and u_ξ and u_η are velocity components in the streamwise (ξ) and normal (η) directions, respectively. Here, the rectilinear axis η is defined in the normal direction to the solid surface (into the fluid region). The curvilinear ξ is defined by the intersection between the plane spanned by η and the relative velocity of the fluid at the external point e with respect to point m, and the surface of the body. According to the axis definition, the scale factor h_η is 1 and h_ξ depends on the surface curvature. In contrast to flat-plate boundary layer flows, the momentum balance expressed by Eq. (2.46b), indicates that surface curvature generates a pressure gradient in the normal direction. Ideally, the local discretization and solution of Eqs. (2.46a) and (2.46b) can be employed to obtain the surface stresses (Posa and Balaras 2014; Balaras et al. 1996). However, for bodies of arbitrary shape and orientation with respect to the fluid grid, the implementation of such a scheme can be very complex leading to costly computations. To avoid direct numerical solution of the above equations we utilize Eq. (2.46b), which governs the pressure variation along the normal direction for the flow over a curvilinear boundary. The pressure gradient along a line $\xi = \text{const.}$ can be expressed in a general form:

$$\left.\frac{\partial p}{\partial \eta}\right|_{\xi=\text{const.}} = g(\eta), \tag{2.47}$$

where $g(\eta)$ is a function of normal coordinate η. The function $g(\eta)$ depends on the coupled effect of the surface curvature and the near flow field and does not admit a universal expression. Assuming as a first-order approximation that $g(\eta)$ varies linearly within the boundary layer, the pressure gradient and pressure near the particle surface are given by:

$$\frac{\partial p}{\partial \eta} = b_\mathrm{p} + a_\mathrm{p}\eta, \tag{2.48a}$$

$$p = c_\mathrm{p} + b_\mathrm{p}\eta + \frac{1}{2}a_\mathrm{p}\eta^2, \tag{2.48b}$$

where a_p, b_p and c_p are coefficients to be determined by the local pressure information $p|_e$, $\partial p/\partial \eta|_e$, and $\partial p/\partial \eta|_m$. Here, $p|_e$ is the pressure at an external point e along the normal direction (as shown in Fig. 2.9). $\partial p/\partial \eta|_m$ and $\partial p/\partial \eta|_e$ are the pressure gradient in the normal direction at the point m on the particle surface and the external point e, respectively. The pressure at the external point e can be interpolated from the Eulerian grid:

$$p|_e = \sum_{i=1}^{\mathrm{ne}} \phi_i^e p_i, \tag{2.49}$$

where ne is the number of Eulerian points in the support domain of external point e. Note that ϕ_i^e are the shape functions relating point e to its interpolation stencil. The pressure gradient in the normal direction at point m can be estimated from the momentum equation:

$$\left.\frac{\partial p}{\partial \eta}\right|_m = \left.\left(-\frac{D\mathbf{u}}{Dt} + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{u}\right)\cdot\mathbf{n}\right|_m \approx \left.-\frac{D\mathbf{u}}{Dt}\cdot\mathbf{n}\right|_m, \tag{2.50}$$

where $D\mathbf{u}/Dt$ is the material derivative of velocity and $\mathbf{n}$ is the unit normal vector at point m on the surface of the body. The viscous force term $(1/\mathrm{Re})\nabla^2\mathbf{u}$ at the surface can be ignored, considering that it mainly contributes to the forces in the tangential direction. The computation of the pressure gradient $\partial p/\partial\eta$ at point e, defined in the curvilinear system $\xi - \eta$, is discussed below. Finally, the pressure on the surface of the particle can be estimated by:

$$p|_m = p|_e - \frac{1}{2}\left(\left.\frac{\partial p}{\partial \eta}\right|_m + \left.\frac{\partial p}{\partial \eta}\right|_e\right)h, \tag{2.51}$$

where h is the distance between points m and e.

The viscous stress on the surface of the body can be evaluated using Eq. (2.46a), which can be simplified by ignoring the effect of inertia and convective terms as suggested in Posa and Balaras (2014). Equation (2.46a) reduces to,

$$\frac{1}{\mathrm{Re}}\frac{\partial^2 u_\xi}{\partial \eta^2} = \frac{1}{h_\xi}\frac{\partial p}{\partial \xi}. \tag{2.52}$$

The effect of curvature on the variation of $\partial p/\partial\xi$ in the normal direction can be investigated by taking the derivative $\partial/\partial\xi$ on Eq. (2.46b),

$$\frac{\partial}{\partial \eta}\left(\frac{\partial p}{\partial \xi}\right) = \frac{\partial}{\partial \xi}\left(\frac{\partial h_\xi}{\partial \eta}u_\xi^2\right), \tag{2.53}$$

where the relation $\frac{\partial}{\partial\xi}\left(\frac{\partial p}{\partial\eta}\right) = \frac{\partial}{\partial\eta}\left(\frac{\partial p}{\partial\xi}\right)$ is used. It is seen from the last equation that the surface curvature causes a variation of $\partial p/\partial\xi$ along the normal direction. Similar to the pressure, the variation of pressure gradient $\partial p/\partial\xi$ depends on the coupled effect of curvature and the flow conditions within the boundary layer, not admitting a universal exact expression. The linear function, $\partial p/\partial\xi = b + a\eta$, can serve as a first-order approximation for the variation of $\partial p/\partial\xi$ in the normal direction. In this scenario, Eq. (2.52) reduces to,

$$\frac{1}{\mathrm{Re}}\frac{\partial^2 u_\xi}{\partial \eta^2} = b + a\eta \tag{2.54}$$

where a and b are two coefficients to be determined by the local flow information. Integrating Eq. (2.54), an analytical expression for the velocity profile at $\xi = \text{const.}$ can be obtained:

$$u_\xi(\eta) = d + c\eta + \frac{b}{2}\eta^2 + \frac{a}{6}\eta^3, \tag{2.55}$$

where again, the c, d coefficients must be determined from the local flow state (i.e., non-slip condition at the boundary point m and the flow information at the external local point e). The details can be found in Wang et al. (2019). The local pressure and velocity distribution resulting from Eqs. (2.51) and (2.55), respectively, can now be used to compute the local forces on the body. In three-dimensional boundary layers, the curvilinear term of the momentum balance equation in the normal direction, Eq. (2.46), will be given by a more complex form involving the curvature in the direction normal to the $\xi - \eta$ plane. However, the pressure gradient within the boundary layer can still be modeled by the linear functions in Eqs. (2.48a) and (2.54). Thus, the proposed hydrodynamic stress model is applicable to both two- and three-dimensional boundary layers.

2.4.3 Example: Flow Over a Circular Cylinder

To demonstrate the features of the force enhancement discussed above, we will consider a circular cylinder of diameter, D, fixed in a uniform cross-flow with velocity U, at Reynolds number $\text{Re} = UD/\nu = 40$. The flow is steady and two-dimensional. The uniform upstream flow is specified at the inlet, and a convective boundary condition is used at the outlet. The free-slip boundary condition is set at the bottom and top boundaries. The non-slip boundary condition is enforced at the cylinder surface. The flow around a cylinder at this Reynolds separates and forms a steady 'dead water' region, as shown in Fig. 2.10. The separation point θ, geometrical measurements of the 'dead water' region (L, a and b as shown in Fig. 2.10) and the drag coefficient, C_D, will be compared to reference data in the literature. To quantify the sensitivity of the hydrodynamic forces to grid resolution when computed using the *normal probe* approach for the MLS direct-forcing scheme discussed above, we conducted a series of computations with increasing resolution. In each case, the grid around the cylinder was approximately uniform and in the range of, $D/384 < \text{dh} < D/24$. The resulting distribution of pressure coefficient, C_p, and the tangential velocity gradient, $\partial u_\xi/\partial\eta$, on the cylinder's surface are shown in Fig. 2.11. The pressure coefficient is fairly insensitive to the grid resolution, and all grids are within 2% of the reference solution. The tangential velocity gradient along the wall-normal direction, one the other hand, is clearly under-predicted on the coarser grids. To quantify the difference with the reference solution, we list the computed $\left|\partial u_\xi/\partial\eta\right|$ at $\theta = 130°$ on different grids on Table 2.1. Approximately 192 grid points across the diameter of the cylinder are needed for the error to be less than 5%, while on the coarsest grid, $\text{dh} = D/24$, the error is 33.7%.

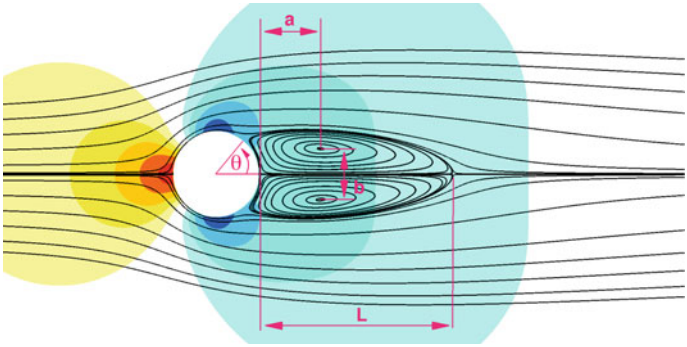


Fig. 2.10 Pressure and streamlines around a stationary circular cylinder at Re = 40. The contours for pressure range from −0.5 (blue) to 0.5 (red) with 10 equal intervals. Figure reprinted with permission from Wang et al. (2019)

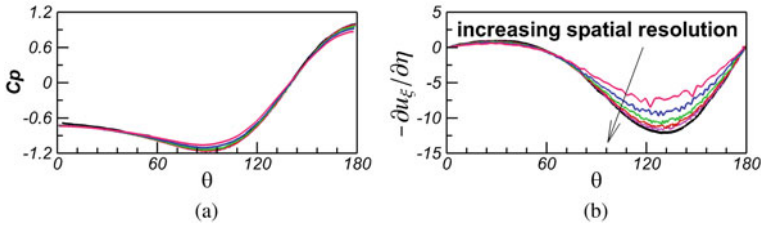


Fig. 2.11 **a** Pressure coefficient, C_p; **b** velocity gradient, $\partial u_\xi/\partial\eta$, on the cylinder's surface at Re = 40. The color lines shows the results computed by the *normal probe* approach based on different grid sizes. The grid sizes are dh = $D/24$, dh = $D/48$, dh = $D/96$, dh = $D/192$ and dh = $D/384$, respectively. The pressure coefficients are shifted by different constants for the simulations with different grids. The black line ——— shows the reference results by Braza et al. (1986). Figure reprinted with permission from Wang et al. (2019)

The prediction of the hydrodynamic force follows a similar trend. Table 2.1 also lists the drag force estimated by the *normal probe* approach for all grids. As expected the error in the viscous drag is higher than that on the pressure drag. For the coarsest grid, (dh = $D/24$) these errors are 14.8% and 2%, respectively. The grid resolution has to be increased at least 8 times (to dh = $D/192$) in each direction, to keep the errors within 5% when computing the viscous drag and within 2% in computing the total drag. These results are consistent with the findings in Tenneti et al. (2011) on the convergence of hydrodynamic forces in IB methods.

We should also note that the errors in the computation of the hydrodynamics forces utilizing a *normal probe* approach depend on the details of the IB formulation as well as the position of the probe. In the present Lagrangian, direct forcing, MLS-based IB approach the kernel width of 2.4dh and the probe extends 2.0dh from the wall. For implementations with wider or narrower support domains and/or the use of higher-order interpolations, the force convergence may be different. The proposed model, which is practically equivalent to a physics-based correction on the forces

Table 2.1 Sensitivity of the tangential velocity gradient and drag force estimated with the *normal probe* approach for the flow around a circular at Re = 40. DT: total drag; DP: pressure drag; DV: viscous drag

Grid length	$\lvert\partial u_\xi/\partial\eta\rvert$	Diff %	DT	Diff %	DP	Diff %	DV	Diff %
$D/24$	7.98	33.7	0.73	−6.4	0.50	−2.0	0.23	−14.8
$D/48$	9.57	20.6	0.75	−3.8	0.50	−2.0	0.25	−7.4
$D/96$	10.59	12.1	0.76	−2.6	0.50	−2.0	0.25	−7.4
$D/192$	11.45	4.9	0.77	−1.3	0.51	0.0	0.26	−3.7
$D/384$	11.82	1.9	0.78	0.0	0.51	0.0	0.27	0.0
Braza et al. (1986)	12.05	–	0.78	–	0.51	–	0.27	–

The reference results are computed from Braza et al. (1986) and the *difference* for a variable, ϕ, is defined as $(\phi_{\text{comp}} - \phi_{\text{ref}})/\phi_{\text{ref}} \times 100\%$

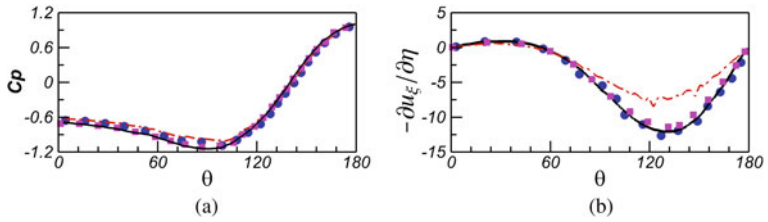


Fig. 2.12 **a** Pressure coefficient, C_p; **b** velocity gradient, $-\partial u_\xi/\partial\eta$, on the cylinder's surface at Re = 40. ——— Reference results by Braza et al. (1986); ● Proposed model on $h = D/24$; the *normal probe* approach on different grids: —·— dh $= D/24$; ■ dh $= D/192$. Figure reprinted with permission from Wang et al. (2019)

predicted by the *normal probe* approach, should be applicable to most direct forcing, immersed boundary methods in a straightforward manner. Depending on the support domains for the transfer functions of the specific formulation, however, the location of the external point, e and grid resolution may have to be adjusted.

To investigate the level of improvement when the model above is utilized, computations on the coarsest of the grids considered above (dh $= D/24$) with and without the model were conducted. Figure 2.12 shows the predicted pressure and shear stress distribution on the cylinder's surface with and without utilizing the proposed model. The reference results in Braza et al. (1986), where a boundary-conforming solver is used, are also included for comparison. It can be seen that the pressure distribution is predicted fairly accurately by both schemes and is less sensitive to grid resolution. The wall stress is captured accurately by the proposed method and is under-predicted when the model is switched off. The predicted separation point, geometrical measurements of the 'dead water' region, L, a and b (as defined in Fig. 2.10) and the drag coefficient, C_D, are excellent agreement with the literature (see Wang et al. 2019 for details) despite the coarse grid resolution.

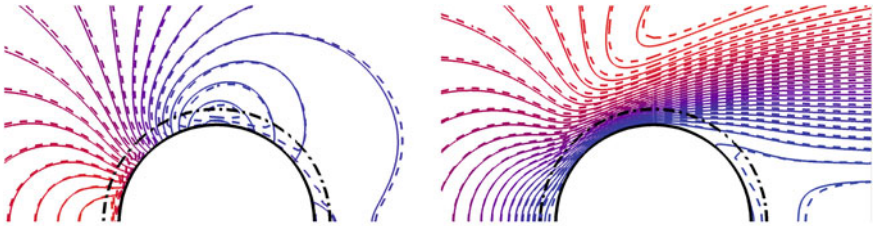


Fig. 2.13 Flow around a circular cylinder at Re $= 40$. **a** Iso-contours of the pressure (range: 0.5 red to -0.5 blue, with an equal interval of 0.05); **b** iso-contours of the streamwise velocity (range 1.15 red to -0.1 blue with an equal interval of 0.05). $-\cdot-$ coarse grid computation (dh $= D/24$) with the proposed model; ——— fine grid (dh $= D/384$) computation. The dash-dotted circle indicates the position of the external point e, at $D/12$ from the wall. Figure reprinted with permission from Wang et al. (2019)

To better understand the near-wall behavior of the model, the pressure and velocity fields obtained with the proposed model on dh $= D/24$ are compared to a high-resolution computation on very fine grid with dh $= D/384$. Isolines of the pressure and velocity distribution in the vicinity of the cylinder are shown in Fig. 2.13. The coarse grid solution consists of two parts: the outer flow computed on the Eulerian grid with dh $= D/24$, and the near-wall flow predicted by the model. The *boundary* between the two is indicated by the dashed line in the figure. The agreement is very good.

2.5 Examples

2.5.1 Free Longitudinal Flight of a Fly Model

In this section, we will present results of free longitudinal flight for a fly model created from digital images of a *Musca Domestica*. The model is composed of one pair of rigid wings, RW and LW, hinged to a rigid body, $RB1$, which represents the insect's head and thorax, and another body, $RB2$, representing the insect's abdomen. The latter is also articulated to $RB1$ (see Fig. 2.14). The coordinate transformations among the different body reference frames with respect to the inertial frame, N, are defined in terms of Euler angle sequences. For the case of longitudinal flight considered here, the lateral motion and rotations around axes in the insects symmetry plane are neglected. The wing kinematics and inertia properties are symmetric.

The head-abdomen length of the model is $L_{TA} = 1.03b$, where b is the one wingspan (hinge to tip). The wings have a thickness of $0.025b$. A simple set of wing kinematics, representative of *Diptera* wing motion is prescribed in all simulations. For the right wing, for example, we set:

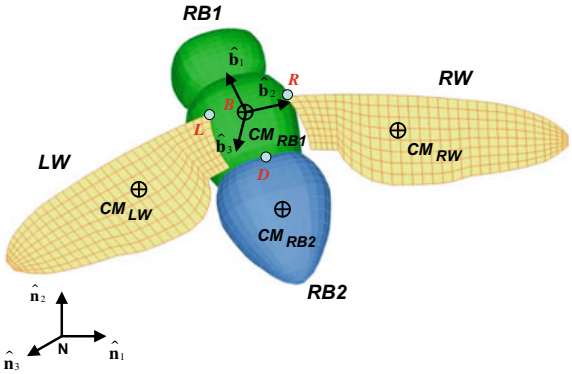


Fig. 2.14 Four-rigid body model for the full insect. Rigid body *RB*1, head-thorax, is represented by the tracking frame *B* attached to its center of mass. Rigid body *RB*2, the abdomen, is hinged to *RB*1 at point *D* where its respective body frame is defined. The wing bodies *RW* and *LW* are articulated to *RB*1 at points *R* and *L*, respectively, where their tracking frames are defined. Figure reprinted with permission from Vanella et al. (2018)

$$\phi_3(t) = A_\phi \sin\left(\omega_f t + \alpha\right), \quad \theta_3(t) = 0, \quad \psi_3(t) = \psi_m + A_\psi \cos\left(\omega_f t\right), \qquad (2.56)$$

where $A_\phi = \pi/4$ is the amplitude of the wing-thorax relative angle of attack, and $\alpha = \pi/6$ (advanced rotation) is the phase. The mean stroke angle is $\psi_m = -\pi/36$, and the stroke amplitude $A_\psi = 55\pi/180$. The reference velocity U_R used is the mean wing tip velocity given by $U_\mathrm{R} = \dot{\psi}_{3\mathrm{mean}} b$, where $\dot{\psi}_{3\mathrm{mean}} = 2A_\psi \omega_f/\pi$. The Reynolds number is $\mathrm{Re} = U_\mathrm{R} b/\nu = 500$. The left wing angles were defined such that symmetric flapping occurs. The *RB*1 orientation angle on an initial prescribed kinematics simulation was set to $\theta = \pi/3$, and the stroke plane was aligned to the horizontal plane. The details on the inertia and material properties can be found in Vanella et al. (2018).

A unitary dimensionless torsion stiffness K_T was employed at the abdomen hinge, keeping the relative angle between *RB*1 and *RB*2 within a 5° amplitude. Also, an advanced mean stroke angle of, $\psi_m = 5.5\pi/180$ was used. A slight pitch up motion was found along the simulation. It is important to note that, in several flapping wing systems, both hovering and forward flight have been found to be dynamically unstable through linear stability analysis (i.e., Sun and Xiong 2005; Taylor and Thomas 2003). Therefore, it is expected that the solution will eventually diverge as the integration progresses.

In Fig. 2.15a, the values of the state variables $x(t), y(t), \theta(t)$ and $\theta_2(t)$ as a function of integration time are shown. The vertical position $z(t)$ is seen to increase throughout the calculation consistent with the fact that the mean resulting vertical force is directed upwards. The horizontal coordinate of the center of mass of *RB*1 takes oscillating positive values, which diminish as the calculation progresses. This is due to the fact that the orientation angle $\theta(t)$ (and stroke plane angle) of *RB*1 increases, introducing a horizontal force component along the $-x$ direction. The oscillatory component of

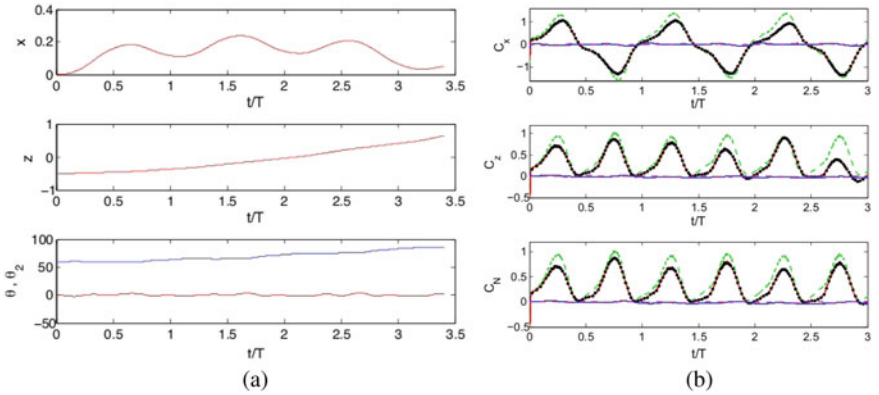


Fig. 2.15 **a** Variation of positions $x(t)$, $z(t)$ and angular coordinates $\theta(t)$ (blue), $\theta_2(t)$ (red) with time for the free longitudinal flight; **b** variation of force coefficients $C_x(t)$, $C_z(t)$ and $C_N(t)$ (force normal to stroke plane) with time for the free longitudinal flight: (blue) $RB1$, (dashed magenta) $RB2$, (red) RW, and (•) LW. The green curves correspond to $C_x(t)$, $C_z(t)$ and $C_L(t)$ from the prescribed kinematics simulation. Figure reprinted with permission from Vanella et al. (2018)

$x(t)$ is about $0.1b$, which is consistent with the data reported in Wu et al. (2009) for a *Manduca Sexta* hawkmoth model with similar wing to body mass ratio. The orientation angle $\theta(t)$ increases steadily, due to moment imbalance. A lower value of ψ_m is required to reduce this effect. Also, $\theta(t)$ oscillates with a peak to peak amplitude of 4° similarly to what is reported in Wu et al. (2009).

In Fig. 2.15b, the time variation of force coefficients in the horizontal, x, and vertical, z, directions, and also in a direction normal to the stroke plane (lift direction for the prescribed kinematics simulation) are shown. It is seen that due to the stroke plane rotation, C_x and C_z vary significantly as time increases. The coefficients resulting from the prescribed kinematics calculation are plotted on the same figures for comparison. All force coefficients take lower values than the fixed body calculation. This body motion effect results in lower ability of the wings to transfer momentum to the fluid. As a consequence, the flight performance of the flapping system is reduced.

The instantaneous flow fields at different simulation times are shown in Fig. 2.16. Here, an isocontour of the second invariant of the velocity gradient tensor, Q, colored by the vorticity, ω_y, shows the evolution of the large coherent structures, namely the leading-edge and tip vortices. The leading-edge vortices attached to each wing detach in the vicinity of the tips (see Fig. 2.16a, b), where the vorticity is reoriented forming the wingtip vortices. Vorticity is also shed from the regions of the wing proximal to the bodies. As a result, two vortical structures are being generated on each wing as shown in Fig. 2.16c, d. The secondary vortices are dependent on the planform geometry of the wings used. The sequence visualized in Fig. 2.16 shows the vertical displacement of the model due to lift force.

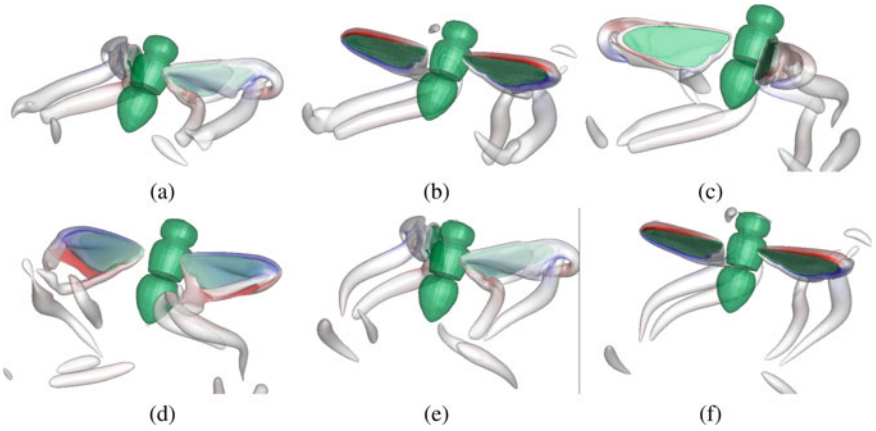


Fig. 2.16 Q isocontour colored by vorticity on the y direction. 40 contours of ω_y from -20 to 20 are used. **a–f** $t/T = 0.75, 1, 1.25, 1.5, 1.75, 2$. Figure reprinted with permission from Vanella et al. (2018)

2.5.2 *Turbulence Interacting with Finite-Size Particles*

The interaction of forced isotropic turbulence with spheres and ellipsoids is presented. Turbulence is generated in a domain with dimensions, $[-\pi, \pi] \times [-\pi, \pi] \times [-\pi, \pi]$, using a linear forcing method (Carroll and Blanquart 2013). The spherical particles have a diameter of $D = \pi/8$, while ellipsoidal particles have a major axis of $a = \pi/4$, and two minor axes of $b = c = \pi/8$. For both cases, the density ratio between the particles and the fluid is 1.02 and the volume fraction 0.2%. The contact model by Wan and Turek (2007) is used to deal with particle–particle collisions. The simulations are conducted on a 256^3 grid at Reynolds number of $\mathrm{Re}_\lambda = 116$.

Typical flow structures are shown in Fig. 2.17, where the particles are larger than the Kolmogorov scale. The shear flows generated near the surface of the particles are visible. These shear flows may reduce velocity fluctuations by increasing the energy dissipation. At the same time, the wakes produced by the particles enhance the production of the turbulent kinematic energy and increase the velocity fluctuations (Bellani et al. 2012). The effects of the particle type on turbulence can be quantified by computing the probability density function (PDF) of the velocity fluctuations (see Fig. 2.18a). We sampled 100 realizations over a time period of $240 \leqslant t < 390$, resulting in $N_p = 100 \times 256 \times 256 \times 256$. The extreme velocity events are damped by the particles, especially the spherical particles. The damping of the extreme velocity events can be clearly seen in Fig. 2.18a where the turbulence fluctuations with spherical particles decay faster when $\xi > 3$. The ellipsoidal particles generate more energetic wakes than the spherical particles and as a result the velocity fluctuations decrease with a lower rate.

The PDFs of the angular velocity of the particles is an important quantity in characterizing this complex interaction and are shown in Fig. 2.18b. To compute

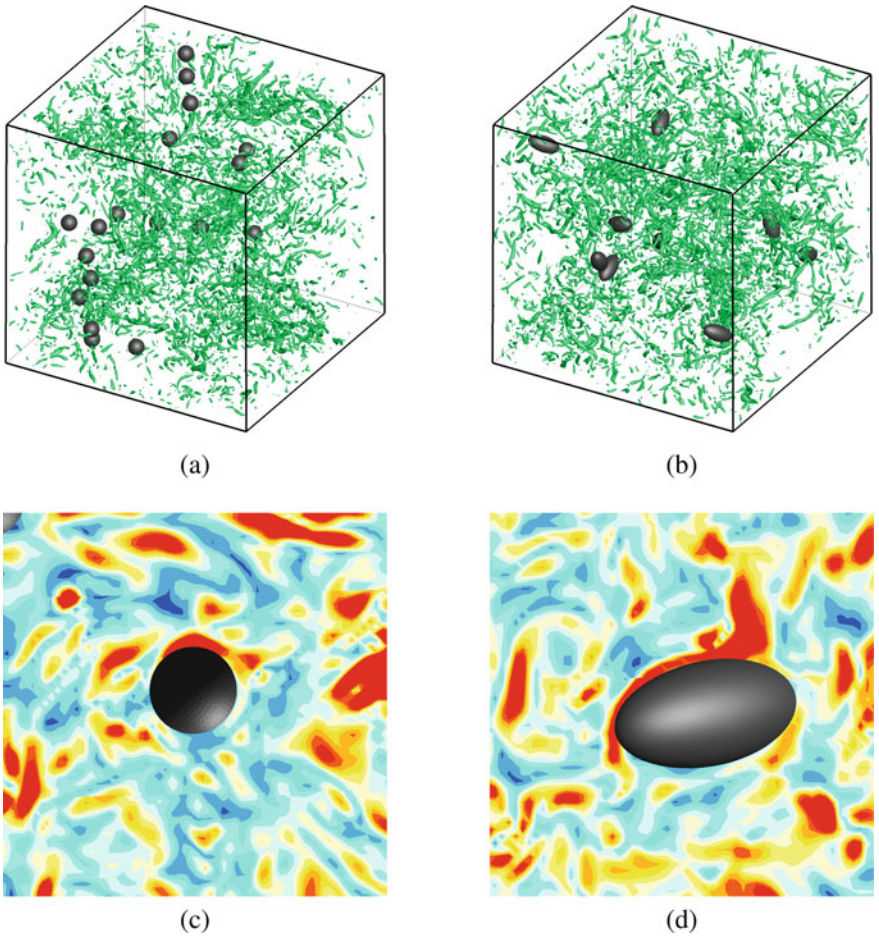


Fig. 2.17 Particle–turbulence interaction, **a** vortex structures around spherical particles, **b** vortex structures around ellipsoidal particles, **c** vorticity magnitude around one spherical particle, from 0 (blue) to 100 (red), 10 intervals, **d** vorticity magnitude around one ellipsoidal particle, from 0 (blue) to 100 (red), 10 intervals. Figure reprinted with permission from Wang et al. (2019)

this quantity, the rotational velocity of each particle is saved at each time step in the simulation. In this case, the number of sample points, N_p, used to compute the PDF of rotational velocity is, $N_p = N_t \cdot n_p$, where n_p is the number of the particles ($n_p = 9$ for ellipsoids, and $n_p = 18$ for spheres), and N_t is the number of discrete time levels used in the sampling. The resulting PDF shown in Fig. 2.18b uses $N_t \times N_P = (8.33 \times 10^5) \times 18$ points for spherical particles and $N_t \times N_P = (8.33 \times 10^5) \times 9$ points for ellipsoidal particles, where $N_t = 8.33 \times 10^5$ is within the simulation time interval of $250 < t < 500$ with $\mathrm{d}t = 3.0 \times 10^{-4}$.

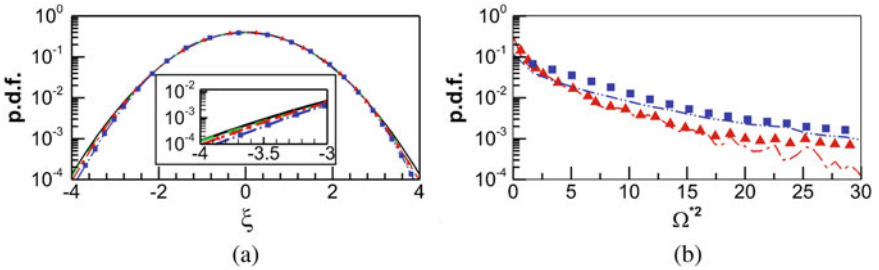


Fig. 2.18 **a** Normalized probability density function of the turbulence with different particles. **b** The probability density function of the particle's rotating speed. ——— normal distribution, – – single phase, – · – ellipsoidal particles with model, – · · – spherical particles with model, ▲ ellipsoidal particles, *normal probe* approach, ■ spherical particles with model, *normal probe* approach. Figure reprinted with permission from Wang et al. (2019)

Both spherical and ellipsoidal particles show similar behavior when the angular velocity is low ($\Omega^{*2} < 5$). At higher rotation speeds ($\Omega^{*2} > 5$), the PDF of ellipsoidal particles takes lower values when compared to that of the spherical particles, likely due to the fact that ellipsoidal particles are less prone to rotation at the speeds defined in this particular problem setup. This is intuitively expected, given the same level of turbulent forcing on both sets of simulations, but higher rotational inertia of the ellipsoids. Another reason might be that the ellipsoidal particles tend to align preferentially with the principal axis of the fluid strain. Overall the effects of the particle types on turbulence, as demonstrated in by the statistics above, is consistent with the experimental observations by Bellani et al. (2012), even though the current simulations are conducted at lower Reynolds numbers.

We have also conducted the simulations without the proposed model. There is no apparent difference on the PDF of the fluid phase, as shown in Fig. 2.18a. However, the rotation of the particles will be over-predicted because of the under-prediction of the shear stress if the proposed model is not used, as shown in Fig. 2.18b. The differences between the model and normal probe approach in the PDF of the velocity fluctuations are small, probably caused by the relatively low Reynolds number in the simulation. However, the PDFs of the particle rotation speed are different and reflect a higher rate for the case of the normal probe approach. This is consistent with the trend by this approach to under-predict the shear stress on the particle surface.

2.6 Outlook and Perspective

CFD simulations in complex flows of practical interest have been traditionally carried out utilizing low-order, finite-volume solvers based on unstructured grids. More recently, the development of high-order techniques, such as discontinuous Galerkin and spectral-element methods, that arguably combine high accuracy with flexibility

in dealing with complex geometries, is constantly gaining ground. In addition, the whole spectrum of methodologies is nowadays within reach of the community by the availability of open-source software (e.g., OpenFOAM, Gerris, Nek5000, PyFR, etc.). As a result, there is a growing interest in exploring the cost/efficiency of various approaches and solvers in different application regimes. Within this framework, immersed boundary methods can be viewed as means of extending the reach of efficient/conservative structured solvers to complex geometrical configurations. They have a particular advantage in problems involving large boundary motion and/or deformation, because addressing the latter within classical boundary-conforming formulations usually comes with increased complexity and cost. We should note, however, that direct comparisons of immersed boundary and boundary-conforming methods in terms of cost for fixed accuracy are sparse and would be an interesting topic for future research.

Improving the efficiency of immersed boundary methods, especially for high Reynolds numbers flows, hinders upon the availability of wall-modeling strategies for this class of methods. Turbulent, high Reynolds number flows pose stringent resolution requirements to all methods, but are particularly restrictive to immersed boundary formulations on structured grids due to the lack of flexibility to selectively distribute points in areas of high-velocity gradients near solid boundaries. Increasing the wall-normal resolution, for example, to resolve the thin boundary layers at high Reynolds numbers usually requires simultaneous refinement in all coordinate directions—rather than only the wall-normal in boundary fitted grids. Direct extension of classical wall-modeling strategies for boundary-conforming solvers is not trivial and as of today, few examples can be found in the literature. It is worth mentioning the recent work by Sih et al. (2019), where a wall-modeling strategy for the class of methods discussed above is proposed. They reported promising results for the case of the flow around an axisymmetric body of revolution. The development of robust wall-modeling strategies for immersed boundary methods is a critical step in expanding in a whole new area of applications.

Acknowledgements We are grateful to Dr. Posa and Prof. Wang for their contributions in developing the enhanced surface force prediction model and for conducting some of the computations reported in this work. The work has been supported by grants from the National Science Foundation, and the Office of Naval Research.

References

Akselvoll K, Moin P (1996) An efficient method for temporal integration of the Navier-Stokes equations in confined axisymmetric geometries. J Comput Phys 125(2):454–463

Armfield S, Street R (2002) An analysis and comparison of the time accuracy of fractional-step methods for the Navier-Stokes equations on staggered grids. Int J Numer Methods Fluids 38(3):255–282

Balaras E (2004) Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations. Comput Fluids 33(3):375–404

Balaras E, Benocci C, Piomelli U (1996) Two-layer approximate boundary conditions for large-eddy simulations. AIAA J 34(6):1111–1119
Bathe KJ (2007) Finite element procedures. Klaus-Jurgen Bathe, Englewood Cliffs
Bellani G, Byron ML, Collignon AG, Meyer CR, Variano EA (2012) Shape effects on turbulent modulation by large nearly neutrally buoyant particles. J Fluid Mech 712:41–60
Braza M, Chassaing P, Minh HH (1986) Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. J Fluid Mech 165:79–130
Carroll PL, Blanquart G (2013) A proposed modification to Lundgren's physical space velocity forcing method for isotropic turbulence. Phys Fluids 25(10):105114
Colonius T, Taira K (2008) A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions. Comput Methods Appl Mech Eng 197(25–28):2131–2146
Constantinescu G, Squires K (2003) LES and DES investigations of turbulent flow over a sphere at Re = 10,000. Flow Turbul Combust 70(1–4):267–298
de Tullio MD, Pascazio G (2016) A moving-least-squares immersed boundary method for simulating the fluid-structure interaction of elastic bodies with arbitrary thickness. J Comput Phys 325:201–225
Fadlun E, Verzicco R, Orlandi P, Mohd-Yusof J (2000) Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161(1):35–60
Glowinski R, Pan TW, Hesla TI, Joseph DD (1999) A distributed Lagrange multiplier/fictitious domain method for particulate flows. Int J Multiph Flow 25:755–794
Glowinski R, Pan TW, Hesla TI, Joseph DD, Periaux J (2000) A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. J Comput Phys 169:363–426
Goldstein D, Handler R, Sirovich L (1993) Modeling a no-slip flow boundary with an external force field. J Comput Phys 105:354–366
Griffith BE, Hornung RD, McQueen DM, Peskin CS (2007) An adaptive, formally second order accurate version of the immersed boundary method. J Comput Phys 223(1):10–49
Gu W, Chyu C, Rockwell D (1994) Timing of vortex formation from an oscillating cylinder. Phys Fluids 6(11):3677–3682
Guilmineau E, Queutey P (2002) A numerical simulation of vortex shedding from an oscillating circular cylinder. J Fluid Struct 16(6):773–794
Hou G, Wang J, Layton A (2012) Numerical methods for fluid-structure interaction—a review. Commun Comput Phys 12(2):337–377
Hughes TJR (2000) The finite element method: linear static and dynamic finite element analysis. Dover Publications, Mineola, NY
Johnson T, Patel V (1999) Flow past a sphere up to a Reynolds number of 300. J Fluid Mech 378:19–70
Kempe T, Frohlich J (2012) An improved immersed boundary method with direct forcing for the simulation of particle laden flows. J Comput Phys 231(9):3663–3684
Kim J, Kim D, Choi H (2001) An immersed-boundary finite-volume method for simulations of flow in complex geometries. J Comput Phys 171(1):132–150
Krishnan S, Shaqfeh E, Iaccarino G (2017) Fully resolved viscoelastic particulate simulations using unstructured grids. J Comput Phys 338(1):313–338
Lai MC, Peskin CS (2000) An immersed boundary method with formal second order accuracy and reduced numerical viscosity. J Comput Phys 160:705–719
Lancaster P, Salkauskas K (1981) Surfaces generated by moving least squares methods. Math Comput 37(155):141–158
Le D, Khoo B (2017) A moving-least-square immersed boundary method for rigid and deformable boundaries in viscous flow. Commun Comput Phys 22(4):913–934
Lee S (2000) A numerical study of the unsteady wake behind a sphere in a uniform flow at moderate Reynolds numbers. Comput Fluids 29(6):639–667
Lee J, Kim J, Choi H, Yang K-S (2011) Sources of spurious force oscillations from an immersed boundary method for moving-body problems. J Comput Phys 230(7):2677–2695

Li D, Wei A, Luo K, Fan J (2015) An improved moving least squares reconstruction for immersed boundary method. Int J Numer Methods Eng 104(8):789–804
Li Z, Ito K (2006) The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains. Society for Industrial and Applied Mathematics
Li S, Liu WK (2004) Meshfree particle methods. Springer, Berlin, Heidelberg
Liu GR, Gu YT (2005) An introduction to meshfree methods and their programming. Springer, Netherlands
Liu WK, Jun S, Zhang YF (1995) Reproducing kernel particle methods. Int J Numer Methods Fluids 20(8–9):1081–1106
Luo H, Dai H, de Sousa PJF, Yin B (2012) On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries. Comput Fluids 56:61–76
Mei R (1994) Flow due to an oscillating sphere and an expression for unsteady drag on the sphere at finite Reynolds number. J Fluid Mech 270:133–174
Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261
Mittal R, Dong H, Bozkurttas M, Najjar F, Vargas A, von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227(10):4825–4852
Mohd-Yusof J (1997) Combined immersed boundaries/B-splines methods for simulations of flows in complex geometries. CTR annual research briefs. NASA Ames/Stanford University, Stanford, CA
Orlanski I (1976) A simple boundary condition for unbounded hyperbolic flows. J Comput Phys 21(3):251–269
Perot JB (1993) An analysis of the fractional step method. J Comput Phys 108(1):51–58
Peskin CS (2003) The immersed boundary method. Acta Numer 11:479–517
Peskin CS (1972) Flow patterns around heart valves—numerical method. J Comput Phys 10(2):252–271
Peskin CS (1977) Numerical analysis of blood flow in the heart. J Comput Phys 25:220–252
Pinelli A, Naqavi IZ, Piomelli U, Favier J (2010) Immersed-boundary methods for general finite-difference and finite-volume Navier-Stokes solvers. J Comput Phys 229(24):9073–9091
Posa A, Balaras E (2014) Model-based near-wall reconstructions for immersed-boundary methods. Theor Comput Fluid Dyn 28(4):473–483
Posa A, Vanella M, Balaras E (2017) An adaptive reconstruction for Lagrangian, direct-forcing, immersed-boundary methods. J Comput Phys 351:22–436
Roma AM, Peskin CS, Berger MJ (1999) An adaptive version of the immersed boundary method. J Comput Phys 153(2):509–534
Seo JH, Mittal R (2011) A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. J Comput Phys 230(19):7347–7363
Sih B, Yang X, Gin G, He G, Wang S (2019) Wall-modeling for large-eddy simulation of flows around an axisymmetric body using the diffuse-interface immersed boundary method. Appl Math Mech Engl Ed 40:305–320
Spandan V, Lohse D, de Tullio M, Verzicco R (2018) A fast moving least squares approximation with adaptive Lagrangian mesh refinement for large scale immersed boundary simulations. J Comput Phys 375:228–239
Spedding G, McArthur J (2010) Span efficiencies of wings at low Reynolds numbers. J Aircraft 47(1):120–128
Sun M, Xiong Y (2005) Dynamic flight stability of a hovering bumblebee. J Exp Biol 208:447–459
Taylor GK, Thomas ALR (2003) Dynamic flight stability in the desert locust *Schistocerca gregaria*. J Exp Biol 206:2803–2829
Tenneti S, Garg R, Subramaniam S (2011) Drag law for monodisperse gas-solid systems using particle-resolved direct numerical simulation of flow past fixed assemblies of spheres. Int J Multiph Flow 37(9):1072–1092
Tomboulides AG (1993) Direct and large-eddy simulation of wake flows: flow past a sphere. Ph.D. thesis, Princeton University, Princeton

Uhlmann M (2005) An immersed boundary method with direct forcing for the simulation of particulate flows. J Comput Phys 209(2):448–476
Van Kan J (1986) A second-order accurate pressure-correction scheme for viscous incompressible flow. SIAM J Sci Stat Comput 7(3):870–891
Vanella M (2010) A fluid structure interaction strategy with application to low Reynolds number flapping flight. Ph.D. thesis, Department of Mechanical Engineering, University of Maryland
Vanella M, Balaras E (2009) A moving-least-squares reconstruction for embedded-boundary formulations. J Comput Phys 228(18):6617–6628
Vanella M, Rabenold P, Balaras E (2010) A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid-structure interaction problems. J Comput Phys 229(18):6427–6449
Vanella M, Posa A, Balaras E (2014) Adaptive mesh refinement for immersed boundary methods. ASME J Fluids Eng 136(4):040909
Vanella M, Wang S, Balaras E (2018) Direct and large-eddy simulations of biological flows. In: Grigoriadis D, Geurts B, Kuerten H, Fröhlich J, Armenio V (eds) Direct and large-eddy simulation X. ERCOFTAC series, vol 24. Springer, Cham
Wan D, Turek S (2007) An efficient multigrid-FEM method for the simulation of solid-liquid two phase flows. J Comput Appl Math 203(2):561–580
Wang S, Vanella M, Balaras E (2019) A hydrodynamic stress model for simulating turbulence/particle interactions with immersed boundary methods. J Comput Phys 382:240–263
Wu JH, Zhang YL, Sun M (2009) Hovering of model insects: simulation by coupling equations of motion with Navier-Stokes equations. J Exp Biol 212:3313–3329
Yang J, Balaras E (2006) An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. J Comput Phys 215(1):12–40
Yang J, Stern F (2015) A non-iterative direct forcing immersed boundary method for strongly-coupled fluid-solid interactions. J Comput Phys 295:779–804
Yang J, Preidikman S, Balaras E (2008) A strongly-coupled, embedded-boundary method for fluid-structure interactions of elastically mounted rigid bodies. J Fluids Struct 24:167–182

Chapter 3
Mass Conservation in Sharp Interface Immersed Boundary Method—A GPGPU Accelerated Implementation

Manish Kumar, Apurva Raj, and Somnath Roy

3.1 Introduction

Immersed boundary method (IBM) is an attractive option for simulation of viscous flow over complex and moving boundaries as it allows the use of a regular structured mesh, and furthermore, the method is inherently very simple to implement. Peskin first proposed this method back in 1972 to study biological flows (Peskin 1972), and in the past few decades IBM has been extensively used in studying wide ranges of applications involving flows observed over complex, moving and deforming bodies (Mittal and Iaccarino 2005; Mittal et al. 2008; Picano et al. 2015). The statistical comparison and computational efficiency of IBM schemes when compared to the conventional body-fitted numerical approaches are also well reported (Verzicco et al. 2000; The Immersed Boundary Approach to Fluid Flow Simulation 2020). Although the sharp interface variant of the IBM is efficient as well as simple to implement, it is posed with difficulties in satisfying the mass conservation in the vicinity of the immersed surface. Unphysical pressure oscillations are also generally observed near the surface boundary, especially in the cases involving moving boundaries in incompressible flows. The mass loss is attributed to the violation of geometric conservation law at the intercepted/immersed cells (Kamakoti and Shyy 2004). Hou and Shi observe that the area loss can be as large as 23% in the immersed cells, and the finite difference discretization needs a very small time step to avoid the significant loss of mass (Hou and Shi 2008). Additionally, in most of the numerical techniques dealing with incompressible flow simulations, pressure correction is computed to ensure a divergence-free velocity field. Therefore, the violation of mass conservation in the intercepted cells leads to spurious pressure fluctuations. The spurious

M. Kumar
Malaviya National Institute of Technology, Jaipur, Rajasthan, India
e-mail: manish.aspiring@gmail.com

A. Raj · S. Roy (✉)
Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India
e-mail: somnath.roy@mech.iitkgp.ac.in

A. Raj
e-mail: raj.apurva05@gmail.com

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_3

pressure fluctuations perceived near the interface in IBM further intensify in case of handling moving or deforming bodies because mass conservation is hampered by the continuously changing fluid volume and continuous changes in the status of the nodes (from solid to fluid and vice versa) in the intercepted cell near the boundary.

The difficulty in ensuring mass conservation and the associated pressure fluctuations near the immersed surface hinder the application of IBM in accurately computing the heat transfer, mixing parameters and dynamic forces over the immersed surface. In case of heat transfer and mixing calculations, the improper mass conservation results in an erroneous prediction of energy and/or species (Hartmann et al. 2006; Roy and Acharya 2012; Lee et al. 2010); whereas, the incorrect dynamic forces lead to imprecise quantification of drag, lift and other forces over the moving surface. Most crucially, the unphysical and inaccurate forces over the surface of the immersed body translate into an incorrect prediction of the trajectories or shape of the body in the next time step, and as the solution marches in time, the achieved simulation results diverge from the realistic behavior.

In context to the abovementioned key challenges in IBM, various researchers have proposed different approaches to overcome them (Ye et al. 1999; Udaykumar et al. 2001; Seo and Mittal 2011; Liu and Hu 2014). These approaches can be classified largely as (i) cut cell approach (Udaykumar et al. 2001; Ye et al. 1999) and (ii) using high grid density (Liu and Hu 2014). A strict adherence to the geometric conservation is attempted to evade the improper mass conservation and associated spurious pressure fluctuations in both the approaches. However, it is reported that the cut cell method results in matrix stiffness issue and involves numerical complexities while handling geometric irregularities (especially for 3D bodies) (Ye et al. 1999; Udaykumar et al. 2001; Seo and Mittal 2011). In case of the implementations using higher grid density, the computational load is significantly high, and these schemes are generally inefficient in solving complex moving body problems. Based on a cut cell method, Seo and Mittal developed a new variant of sharp interface IBM (Seo and Mittal 2011), which accounts for the mass in/out of the cut cell adjacent to the moving and thereby strongly imposes the mass conservation. The smaller cut cells are merged with the regular cells to avoid ill-conditioning. This scheme further suppresses the pressure fluctuations and evades the stiff equations as well but the numerical complexities are more involved. Liu and Hu proposed a local grid refinement method with a higher time step size and modified interpolation schemes coupled with dynamic weight functions (Liu and Hu 2014) to check the spurious pressure oscillations. A novel local grid refinement method is proposed which uses a marker paving algorithm based on iso-parametric mapping (Posa et al. 2017). Also, the divergence-free continuum Lagrangian velocity field is obtained by defining an edge-centered discrete vector potential using a staggered grid discretization and interpolating it in the conventional IB fashion (Bao et al. 2017). The spurious force oscillations are avoided using a smaller computing cell by a novel diffuse interface IBM technique (De 2018). Further, some other techniques are also used to address these issues. These involve using smoother delta functions (Yang et al. 2009; Taira and Colonius 2007), constraining the velocities at the immersed cells (Muldoon and

Acharya 2008), optimizing the interpolation functions for divergence minimization (Kang et al. 2009) and reconstructing of the interpolated solution at the intercepted cells using an extra set of equations (Liao et al. 2010). However, the inability to represent sharp interfaces, application of periodic boundary conditions, higher computational cost and much increased implementation efforts are associated with these reported special treatments to overcome improper mass conservation and unphysical pressure fluctuations. Therefore, it is important to discuss a simple and efficient scheme which can address these issues.

The majority of incompressible IBM-based flow solvers satisfy mass conservation by solving the Poisson equation for pressure or pressure correction. It is a computationally expensive process, and this step consumes a maximum amount of time in the whole calculation. In an IBM implementation, the other expensive steps are the search for intercepted cells and the velocity reconstruction steps. Therefore, large-scale simulations using IBM must be augmented with high-performance computing implementations. So, we have also included a discussion on the general-purpose computing on graphics processing units (GPGPU) implementation of the IBM solvers in this chapter. Earlier, computational fluid dynamics (CFD) codes for atmospheric flows and wind forecasting have been accelerated using GPU clusters (DeLeon and Felzien 2012) with a dual-level parallel implementation interleaving Message Passing Interface (MPI) with CUDA (2020). In one of the pioneering work of accelerating IBM codes in GPU, the open-source CUSP (2020) and Thrust libraries (2020) are used by Layton et al. (2011) to solve the 2D incompressible viscous flows using the GPU. The efficacy of OpenACC (2020), minimizing the programming effort, has been exploited by Kraus et al. (2014) to accelerate the flow solver ZFS on GPUs. They have reported that the OpenACC implementation shows a speedup of 2.44× as compared to a parallel MPI-based implementation. GPUs using CUDA are used to obtain high performance for a direct forcing IBM by Tutkun and Edis (2017).

The present chapter discusses a simple but efficient implementation of mass conserving sharp interface immersed boundary method for incompressible flows and also shows its acceleration in GPUs using OpenACC.

3.2 Immersed Boundary Method

The idea of using a Eulerian grid not conforming to the immersed body is first developed by Peskin for simulating cardiovascular flows (Peskin 1972). In this method, the entire simulation is carried out on a regular structured Cartesian grid, which is generated without considering the location of the curved or moving geometries residing within its bounds. The geometry, which is immersed in the flow domain, is represented by a separate Lagrangian description. The Eulerian flow grid is static irrespective of the state of motion of the geometry. This method is depicted in Fig. 3.1.

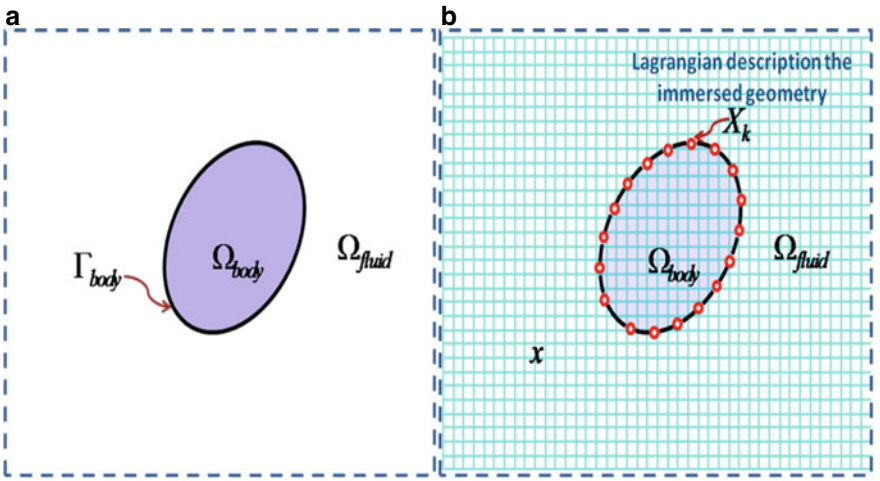


Fig. 3.1 **a** General representation of domain over which flow is to be computed. **b** Depiction of the basic principle of immersed boundary method for the considered flow domain

Since a nonconforming body grid is used for discretizing the flow equations, the application of boundary conditions through the Lagrangian surface marker points separates one IBM technique from another. Generally, the boundary conditions are applied by introducing the necessary forcing terms in the governing equations or by velocity or momentum interpolations. The near boundary grid points are identified first, and then, the forcing/interpolation is applied at those points. This indirect way of considering boundaries helps in avoiding complexities associated with the body-fitted approaches for arbitrary, moving and deforming geometries. IBM helps in reducing the costly dynamic meshing, and the issues associated with large deforming mesh and the solution interpolation steps can be bypassed. It can also help in achieving better computational efficiency because the domain decomposition and load balancing steps for the parallel processes are much simpler to implement in a Cartesian mesh framework (Yildirim et al. 2013; Anupindi et al. 2013).

As discussed above, the IBM uses a Cartesian structured mesh (Eulerian grid) framework on which the flow equations are solved. The immersed surface is represented by an unstructured triangular mesh in the three-dimensional space. The nodes that lie inside the geometry on the Cartesian grid are denoted as solid nodes, whereas the nodes lying outside the surface are called fluid nodes. The cells are identified as solid and fluid if all the nodes of a cell are inside or outside the immersed body, respectively. The cells encapsulating the immersed boundary have both solid and fluid corner nodes, and these cells are termed as intercepted (or immersed) cells. The fluid and solid nodes of intercepted cells are termed as immersed and ghost nodes, respectively. Figure 3.2 shows the cells and nodes as per the discussed nomenclature.

Although the method is simple and potentially efficient, the issues like conserving mass near the vicinity of immersed surface and the unphysical oscillations in pressure,

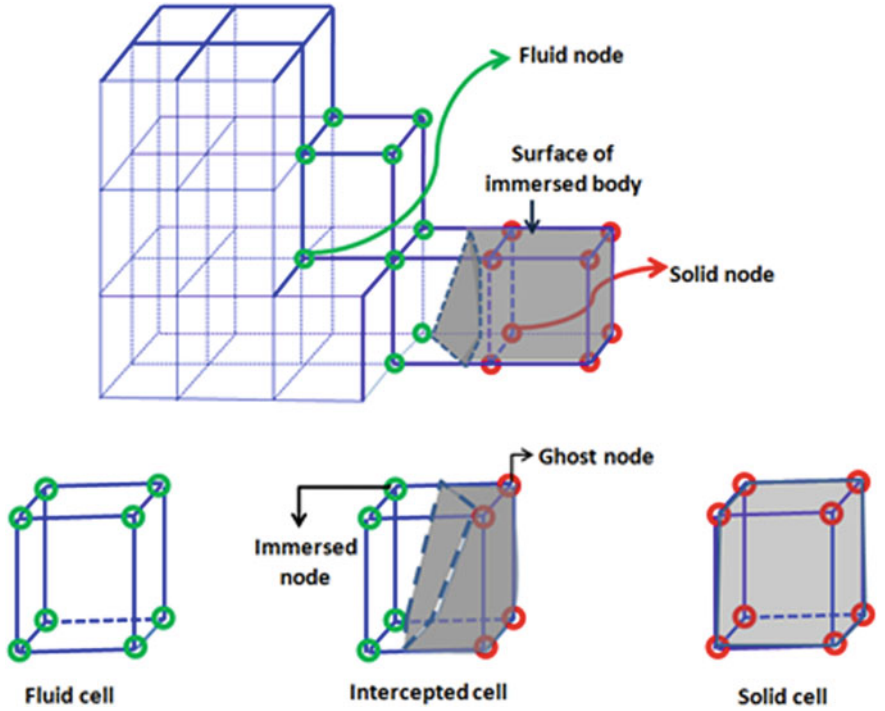


Fig. 3.2 Representation of the immersed body in the 3D view and different terminology associated with IBM. Reproduced with permission from Kumar and Roy (2016)

etc., are observed when the geometry is moving. It is important to resolve these issues to ensure the fidelity and accuracy of the IBM. In this regard, a simple and efficient MAC-SOLA-based sharp interface IBM (Kumar et al. 2016; Kumar and Roy 2016) is devised.

3.3 Computational Methodology

The present IBM implementation is based on a discrete forcing approach. At first, the fluid, solid and intercepted cells are segregated, and afterward, the Navier–Stokes momentum equations are solved on the fluid cells only. The velocity interpolation (forcing) is used to accordingly reconstruct the velocities at the intercepted cells (Fig. 3.3).

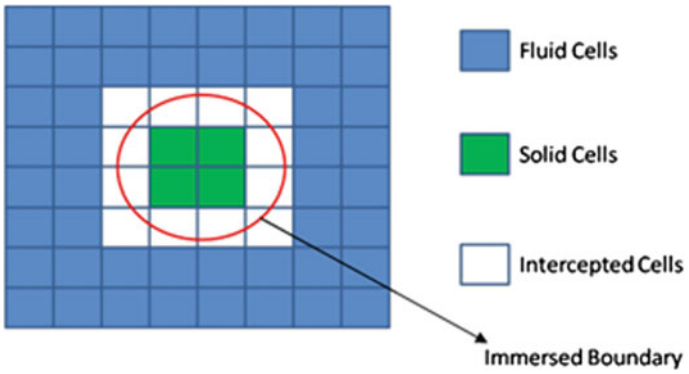


Fig. 3.3 Discrete forcing-based IBM

3.3.1 Governing Equations and Solution Methodology

The Navier–Stokes momentum and continuity equations for incompressible viscous flows of Newtonian fluids in the non-dimensional form, given by Eqs. 3.1 and 3.2, are solved on a Cartesian grid, where u is the velocity, p is the pressure, t is the time, and Re is the Reynolds numbers.

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x_j} u_i u_j = -\frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}} \frac{\partial}{\partial x_j} \frac{\partial}{\partial x_j} u_i \tag{3.1}$$

$$\frac{\partial u_j}{\partial x_j} = 0 \tag{3.2}$$

A Marker and Cell (MAC)-based projection method is used to discretize the momentum Eq. 3.3 on a staggered grid, and projected velocity, $\tilde{u}$, is computed at $n+1$ time step from the known values at n and $n-1$ time steps.

$$\tilde{u}_i^{n+1} = u_i^n - \delta t \left(\frac{\mathrm{d}p}{\mathrm{d}x} \right)_i^n - 0.5\delta t \{3(\text{conv.} - \text{diff.})_i^n - (\text{conv.} - \text{diff.})_i^{n-1}\} \tag{3.3}$$

where conv. are the convective terms discretized using a second-order upwind scheme, whereas a second-order central difference scheme is used to discretize the diffusive terms, diff. The temporal marching of the solution is accomplished by explicit Adams–Bashforth method.

The correct velocity is obtained by evoking the continuity equation which results in pressure correction Poisson equation given by Eq. 3.4.

$$\frac{\partial}{\partial x_i} \frac{\partial}{\partial x_i} p' = \frac{1}{\delta t} \frac{\partial \tilde{u}_i}{\partial x_i} \tag{3.4}$$

Once the pressure correction, p', is computed, pressure and velocity are corrected as follows:

$$p^{n+1} = p^n + p' \tag{3.5}$$

$$u^{n+1} = \tilde{u}^{n+1} - \delta t \frac{\partial p'}{\partial x_i} \tag{3.6}$$

Specifying the boundary conditions for the fixed Cartesian boundaries, boundary conditions are trivial but it is challenging near the immersed surface.

3.3.2 Search Algorithm (Tagging)

Identification of the intercepted, fluid and solid cells on the Cartesian grid, based on the location of the immersed body, is the first step in using IBM. This can be viewed as an alternative of the grid generation step in a body conformal grid solution technique. In this step, the complex geometry (Lagrangian grid) is mapped onto the rectangular (Eulerian) grid. The Eulerian domain has to be marked as inside or outside based on the position of geometry. Two of the most commonly used algorithms to map the geometry are (i) the ray-tracing (Khalighi et al. 2020) and (ii) normal-position vector dot product method (Kumar et al. 2016). The normal-position vector dot product method has edge over ray-tracing algorithm as ray-tracing often fails to accurately map the geometry in the cases of complex shapes (Problems in ray tracing 2020).

Since the geometry does not conform to the Cartesian grid, at first, a separate triangular mesh description (a B-Rep representation) is obtained to define the geometry. This can be done through CAD software. Now, the triangular mesh is considered as immersed in the Cartesian grid. The surface normal-position vector dot product method has been used to identify whether an Eulerian grid point is inside or outside of the triangular mesh. It is to be noted that this process is computationally intensive and demands an efficient search algorithm as both the meshes may comprise 10^5–10^6 nodes. The cell identification algorithm can be described in steps as follows (Fig. 3.4):

1. Calculate the unit normal vector (**n**) for each triangular element
2. For each node, based on the minimum length of the position vector (**p**) joining the node with the centroid of the surface triangular element, identify the closet triangular element
3. Determine $\mathbf{p} \cdot \mathbf{n}$ for that node
4. Mark nodes as Solid if $\mathbf{p} \cdot \mathbf{n} \leq 0$, else, Fluid
5. Mark the cells as fluid (if all eight corner nodes are fluid), solid (if all eight corner nodes are solid) and intercepted cells (if corner nodes comprise both solid and fluid).

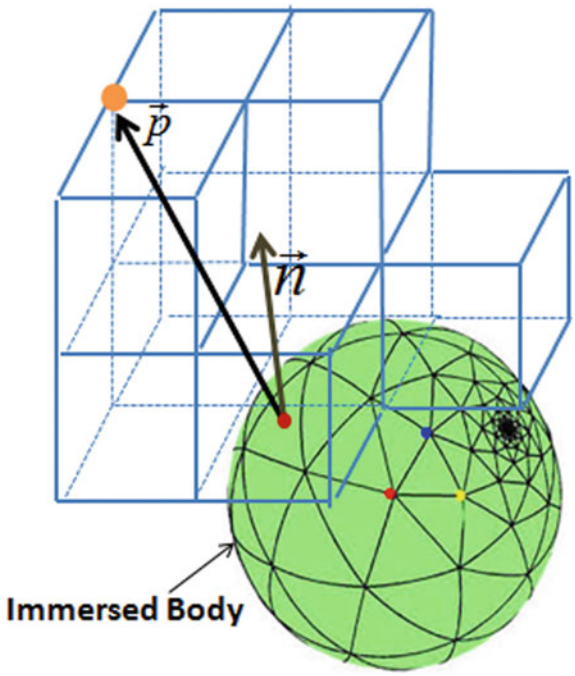


Fig. 3.4 Search algorithm—normal and position vector to a fluid mesh point from the immersed surface

In the case of moving boundaries, the search algorithm is executed at time step, which is obtained using a confined search. Selective retagging is used on the Cartesian grid after first time step. This retagging algorithm stores the location of the intercepted cells at every time iteration and performs search through the next two cells (in each direction) adjacent to the intercepted cell (as depicted in Fig. 3.5). It is to be mentioned that tagging (for moving or fixed cases) is done separately for all flow component variables of the staggered mesh system.

Further, while solving internal flows (like flow in S-bend pipe, cardiovascular flows, etc.) with high solid to fluid nodes ratio, unnecessary computational overheads due to large number of redundant solid nodes are considered as another bottleneck in the IBM (Anupindi et al. 2013; de Zélicourt et al. 2009). These solid nodes are redundant in case the solid is a rigid body, however, for internal flows in curved geometries, they can be very high in number. Therefore, it is desirable to opt out these redundant nodes completely from participating in the computations. One direct way to achieve this is to use if-else statements, but this simple logical operation adds significant computational overheads in case of large number of redundant nodes (de Zélicourt et al. 2009). Kumar et al. (2016) proposed a simple idea of reducing redundant nodes while solving internal flow by listing down the desirable nodes in a separate file and allowing the solution to loop over these nodes only (depicted in Fig. 3.6). For moving bodies, the list file is updated at the beginning of each time iteration using a fresh search (using the selective retagging as discussed previously)

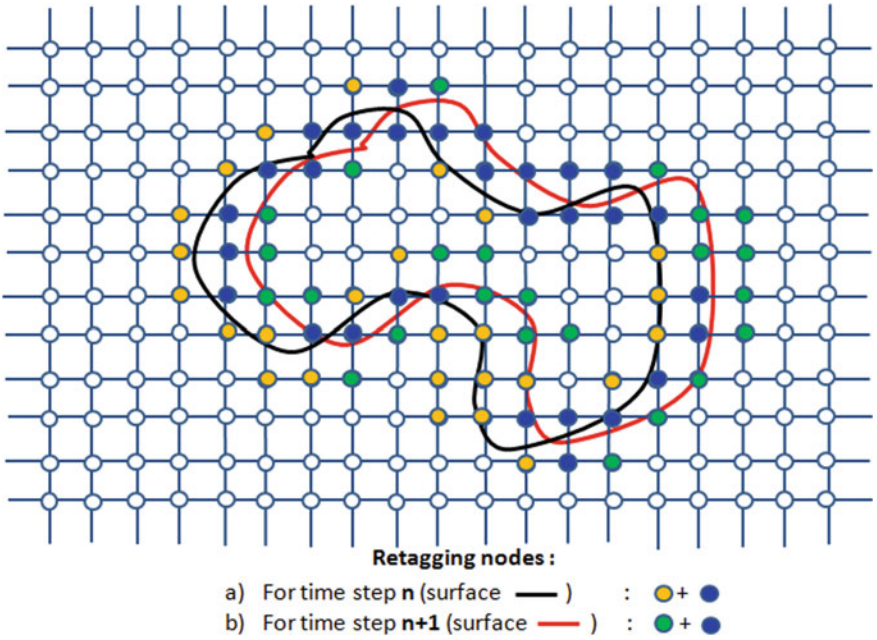


Fig. 3.5 Search algorithm—scheme depicting the short listing of nodes for retagging at the next time step in moving boundary problem

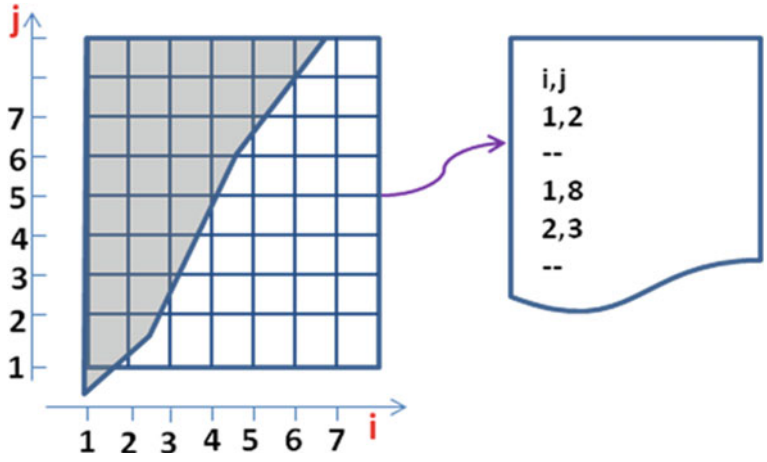


Fig. 3.6 Schematic representation of the scheme: listing the nodes in file over which solution loop is carried out

for fluid and immersed/ghost nodes. This implementation reduces the overall memory requirements and the floating-point operations significantly.

3.3.3 *Implementation of Boundary Condition*

This step is also termed as reconstruction of solutions (i.e., field variables) at the intercepted cells using interpolation/ extrapolation. Different strategies are recommended by the different authors for solution reconstruction (Iaccarino and Verzicco 2003; Tyagi and Acharya 2005; Gilmanov and Sotiropoulos 2005; Choi et al. 2007). In this work, a simple quadratic function ($f(n) = an^2 + bn + c$), where n is the coordinate along the surface normal direction (Fig. 3.7) while f is the flow variable: u, v, w or p along the surface normal is used for interpolation/extrapolation at

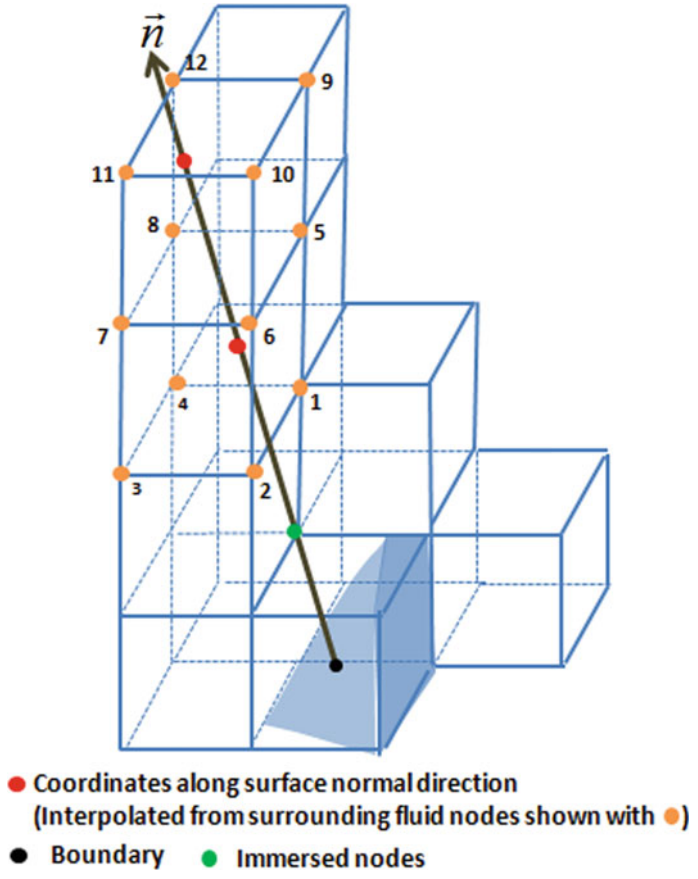


Fig. 3.7 Solution reconstruction scheme (i.e., interpolation) in three dimension

the immersed/ghost nodes. In the present scheme, the solutions are constructed at both faces of the intercepted cell, i.e., both in the fluid and solid (ghost) parts of the intercepted cell. The interpolation/extrapolation ensures that the velocity satisfies no-slip conditions on the solid surface (triangular mesh), and the pressure satisfies the following condition (Gresho and Sani 1987):

$$\frac{\partial p}{\partial \mathbf{n}} = -\mathbf{n}\frac{D\mathbf{u}}{Dt} \tag{3.7}$$

3.3.4 Coupled MAC-SOLA Solver

The divergence-free velocity is obtained by solving the pressure correction Poisson equation (Eq. 3.4).

While solving Poisson equation (Eq. 3.4), pressure correction values at the neighboring nodes are needed, and hence, the requirement of specifying pressure boundary conditions near the edges of the flow domain arises, which are not available as these points lie inside the solid. Whereas, Hirt et al. (1975) proposed SOLA, which assumes that the pressure in the neighboring cells is correct and velocities are iteratively updated based on local divergence only. Here, the pressure correction for a particular cell can be found out iteratively from the divergence of the local cell only (Eq. 3.8).

$$p' = -\frac{\omega \frac{\partial u_i'}{\partial x_i}}{2\delta t\left(\frac{1}{\delta x^2} + \frac{1}{\delta y^2} + \frac{1}{\delta z^2}\right)} \tag{3.8}$$

where ω is the over-relaxation factor. As a result, in this approach, the needs of the boundary conditions in the pressure of pressure correction (i.e., pressure in the neighboring solid cells) are avoided. Hence, this method can be easily deployed for arbitrarily intercepted cells in an IBM approach.

Therefore, a coupled MAC-SOLA scheme (Kumar et al. 2016; Kumar and Roy 2016) (in which MAC is applied in the fluid cells, for which neighboring pressure correction is available, while SOLA is applied at the intercepted cells) is a better option. In the coupled MAC-SOLA, at first, modified accelerated SOLA (Algorithm 1) is applied to obtain pressure correction, corrected pressure and velocity at the intercepted cells.

Algorithm 1—SOLA at intercepted cells

For each intercepted cell
  compute divergence from projected velocity
  compute pressure correction from Eq. 3.8
  update pressure using Eq. 3.9
  update velocity components at the interfaces using Eqs. 3.10 and 3.11
End For

$$p^{n+1} = p^n + p' \tag{3.9}$$

$$u_i^{n+1} = \tilde{u}_i^{n+1} + \gamma \left(\frac{\partial \tilde{u}}{\partial x}\right)_i \tag{3.10}$$

$$u_{i-1}^{n+1} = \tilde{u}_{i-1}^{n+1} - \gamma \left(\frac{\partial \tilde{u}}{\partial x}\right)_i \tag{3.11}$$

In Eqs. 3.10 and 3.11, γ is a parameter based on the spatial steps. As the pressure and velocities are updated iteratively, an over-relaxation factor $1 < \omega_0 < 2$ is used for fast convergence given by Eq. 3.12.

$$\gamma = -\frac{\omega_0}{2\delta x(\frac{1}{\delta x^2} + \frac{1}{\delta y^2} + \frac{1}{\delta z^2})} \tag{3.12}$$

MAC (Algorithm 2) is applied afterward to find the pressure correction and updated flow variables at the fluid cells.

Algorithm 2—MAC at fluid cells
apply pressure correction boundary conditions
compute divergence for each fluid cell
solve pressure correction Poisson equation Eq. 3.4
update pressure using Eq. 3.9
update velocity components at the interfaces using Eqs. 3.10 and 3.11
apply velocity boundary condition

The above MAC-SOLA computations are iteratively performed until maximum divergence in the intercepted cells (obtained through SOLA steps) is less than the set tolerance within a time step. Pressure is enforced at the nodes of intercepted cell using the interpolation scheme discussed before. It has to be mentioned that as the pressure correction is obtained for the fluid cells and intercepted cells using local velocity divergence, the corrected velocity field satisfies mass conservation and the pressure field is also free from unphysical fluctuations. Figure 3.8 shows the flowchart of the solver.

3.3.5 Computational Efficiency of MAC-SOLA

As the SOLA loops are iterative in nature, its convergence may add to the computational overhead of the coupled MAC-SOLA scheme. So, it will be important to benchmark the performance of a MAC-SOLA scheme (i.e., mass conserving scheme) over a straightforward MAC-based reconstruction scheme (without mass conservation). Kumar and Roy (2016) investigate several cases involving fixed and moving IBM simulation with and without mass conservation and infer that MAC-SOLA does not show reduced efficiency as compared with a MAC simulation. Furthermore, as the mesh size increases, the mass conserving MAC-SOLA scheme shows slightly improved computational efficiency than the MAC scheme (Fig. 3.9). This can be

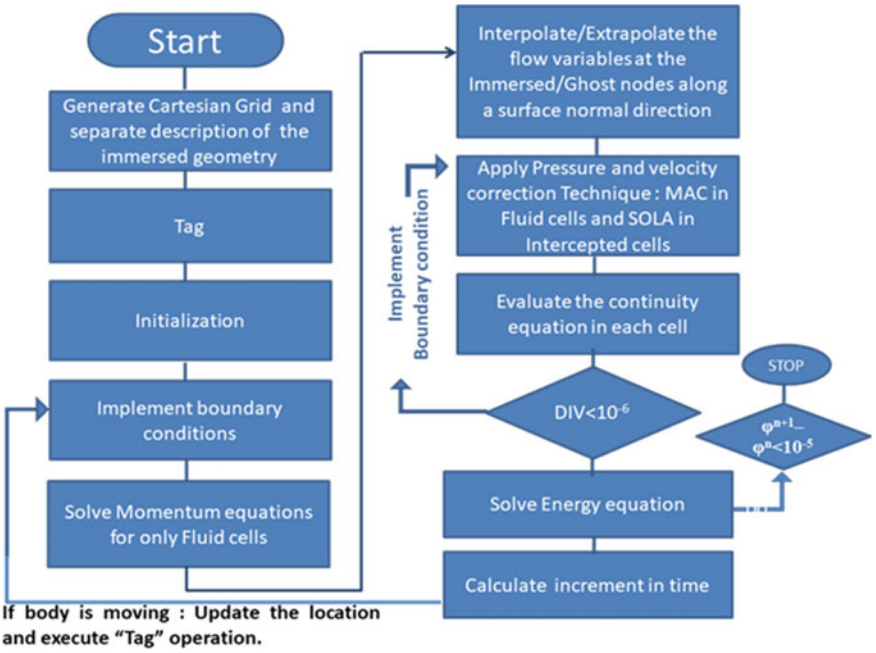


Fig. 3.8 Schematic representation of the scheme: listing the nodes in file over which solution loop is carried out

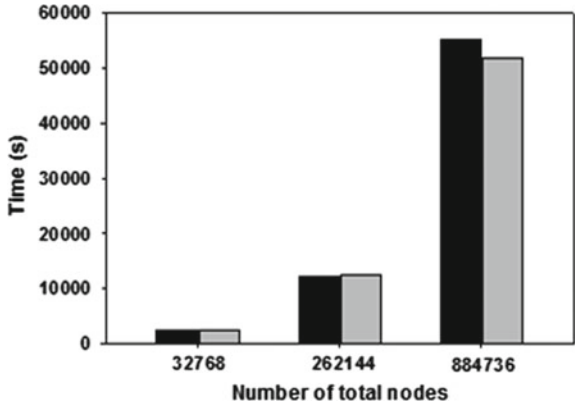


Fig. 3.9 Comparison of computational time required for different grid sizes with implementation of MAC (without mass conservation) and MAC-SOLA (with mass conservation) solver for flow over a moving sphere inside a cubic enclosure. Reproduced with permission from Kumar and Roy (2016)

attributed to the fact that as the intercepted pressure correction values are specified at the intercepted cell, the conditioning of the pressure correction factor improves, and hence, convergence is faster.

3.4 GPGPU Acceleration of IBM Solver

The MAC-SOLA IBM uses a SOLA loop over MAC computations. In case of incompressible flow solvers, the solution of Poisson equation of pressure or pressure correction is observed to be most expensive in the entire calculation. Further, the initial search and the forcing add to the computational cost. Therefore, it is necessary to use parallelization techniques if some real-life problem with large number of nodes needs to be simulated. The present section discusses the GPGPU acceleration of MAC-SOLA IBM solver using OpenACC. The parallelization strategy and the optimization steps used for accelerating the legacy code are discussed in detail by Raj et al. (2018).

Graphical processing units (GPUs) have found the niche in computational research over the last decade as general-purpose highly parallel computing units. GPUs consist of a large number of streaming multiprocessors (SM), each of which is a set of computing cores having a fixed number of registers. Each SM has a fast on-chip memory shared among its cores, whereas a slower off-chip memory, with a much larger capacity, known as device memory is shared across different SMs. The general-purpose computing on GPUs (GPGPU) uses GPU as a device and CPU as a host. A kernel is the part of or a program itself that is executed on GPU device using large number of threads organized into thread blocks and grids.

OpenACC is a compiler directive-based programming model. It facilitates quick porting of existing applications to accelerators like GPUs and multicore CPUs without significant programming effort. An additional information is provided to the compiler through these directives enabling the code optimization within the directives for a specific accelerator. OpenACC also allows the compilation of the same code for different accelerators. The two types of directives provided by OpenACC broadly are data directives and compute directives. More details on the OpenACC directives can be found in the OpenACC Programming and Best Practices Guide 2015) and the OpenACC specifications document (2020).

In the present implementation, GPGPU acceleration of MAC-SOLA legacy solver is accomplished via OpenACC. At first, all the variables are first copied to the GPU (device side), and necessary update clauses are used wherever necessary to move the data between host and device. This step ensures that the data copy between host and device is minimized. The parallelization of few important functionalities is discussed in the below subsections.

3.4.1 OpenACC Acceleration of Search Algorithm

The search algorithm is discussed before and like earlier, at first time step, the surface mesh file is read sequentially before classifying the cells as fluid, solid or intercepted. Once the grid/Cartesian and Lagrangian nodes are read from the input file

sequentially, OpenACC parallel loop constructs and the collapse clauses are used to parallelize the search algorithm at first time step.

In the case of moving boundaries, selective retagging is used after the first time step, and update clause is used for the array storing the cell id. The clause is used so that the data is updated on the host side which is required while running MAC-SOLA computations.

3.4.2 OpenACC Acceleration of MAC-SOLA Algorithm

The MAC algorithm is used to calculate the pressure correction for every fluid cell. It is solved using basic iterative methods—Jacobi and successive over-relaxation, whereas SOLA is applied as the boundary conditions at the intercepted cells iteratively. MAC computations are independent across each MAC-SOLA iteration and can be executed in parallel.

In the case of Jacobi method, the degree of parallelism is maximized, and a good occupancy is maintained by using the collapse clause of OpenACC. It is used to collapse the three dimensions (i, j, k) of the 3D mesh and distribute each iteration among the GPU threads. The convergence value is computed in parallel using the max reduction clause. The SOR method has a faster convergence than the Jacobi but is difficult to parallelize.

SOLA has data dependency on pressure correction, hence parallelization is not possible and the SOLA is run on CPU. Afterward, the update clause is used to update the variables on GPU.

Figure 3.10 shows the strategy used for accelerating the couple MAC-SOLA algorithm.

3.5 Results and Discussions

3.5.1 Mass Conservation and Pressure Fluctuation

3.5.1.1 Mass Conservation

To present the effectiveness of MAC-SOLA scheme in mass conservation, both the global and local mass conservation results are presented here. For global mass conservation, flow through a conical nozzle is simulated at Re equal to 30 (based on inlet diameter and velocity). The uniform inlet and Orlanski outflow boundary (Orlanski 1976) conditions are implemented at the inlet and outlet of the nozzle, respectively. The difference between mass flow rates through the inlet and outlet sections is quantified as the global mass loss. In Fig. 3.11, global mass loss versus volume mesh width (Δh) plots are depicted in a logarithmic scale, which shows a

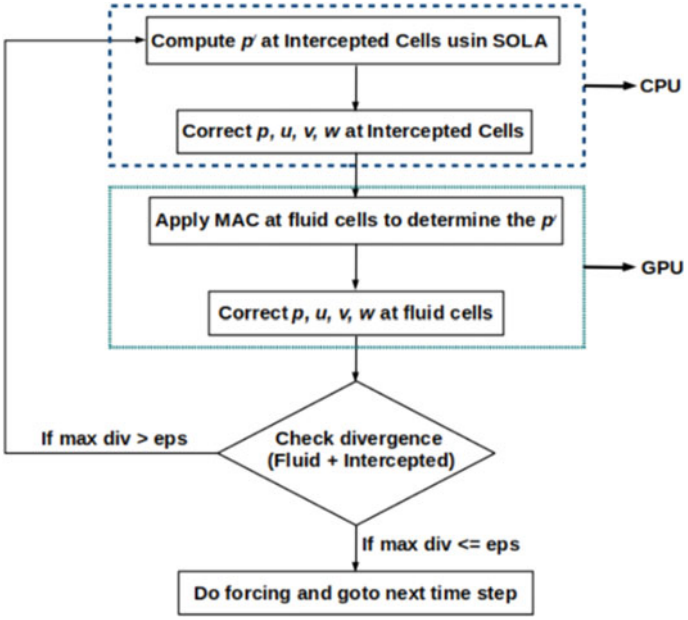


Fig. 3.10 GPU acceleration strategy for coupled MAC-SOLA scheme

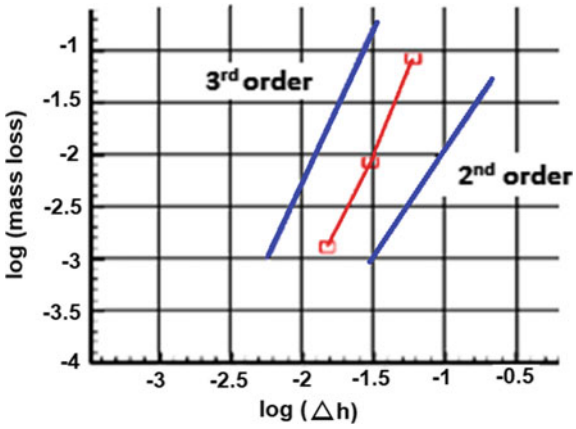


Fig. 3.11 Mass loss study for flow inside the conical channel—log (Δh) versus log (mass loss). Reproduced with permission from Kumar and Roy (2016)

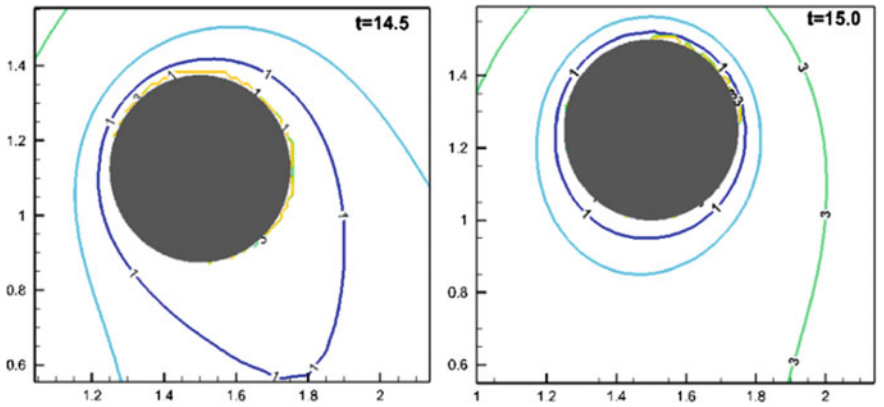


Fig. 3.12 Difference between mass in and out of each cell for transverse oscillation of cylinder inside a channel: at two time instants. Reproduced with permission from Kumar and Roy (2016)

good control in global mass conservation, and the order of accuracy lies between second and third as well. Further, in order to more critically investigate the mass conservation, the local mass loss (i.e., mass flow rates through the inlet faces minus mass flow rate through outlet faces of each intercepted cells) for the flow past a transversely oscillating cylinder is tested. The cylinder is oscillating in transverse direction using $y(t) = y_c + 0.125(1 - \cos 2\pi t)$ as equation of motion. In Fig. 3.12, local mass losses at two different time instants are shown. The results reveal that the maximum mass loss in an individual cell is of the order of 10^{-4}. The global mass loss (i.e., difference of mass loss between inlet and outlet cross section of the channel) is also of the order of 10^{-4}. Further, other moving cylinder cases (like in-line oscillation of cylinder: inside enclosed square and a channel) also showed a similar small value of mass loss (local or global).

Therefore, it can be inferred that for both moving and fixed boundary problems, the present scheme shows good mass conserving properties.

3.5.1.2 Pressure Fluctuation

To present the performance of coupled MAC-SOLA in concern to spurious pressure fluctuations, the time history of the average pressure drag over the in-line is plotted with different time steps and/or grid sizes, respectively (Fig. 3.13). The cases are simulated with and without mass conservation in the intercepted cell. In case of in-line oscillating cylinder, the cylinder is allowed to oscillate according to equation of motion: $x(t) = x_c(0) + A(1 - \cos 2\pi f t)$, where amplitude $A = 0.05D$ and D is the diameter of the oscillating cylinder. The Reynolds number, $\mathrm{Re} = \frac{u_0 D}{\nu}$ where $u_0 = 2\pi f A$ and Strohal number $St = fD/u_o$ are set equal to 78.5 and 1.27, respectively. The further setup details (like boundary condition, etc.) can be seen in

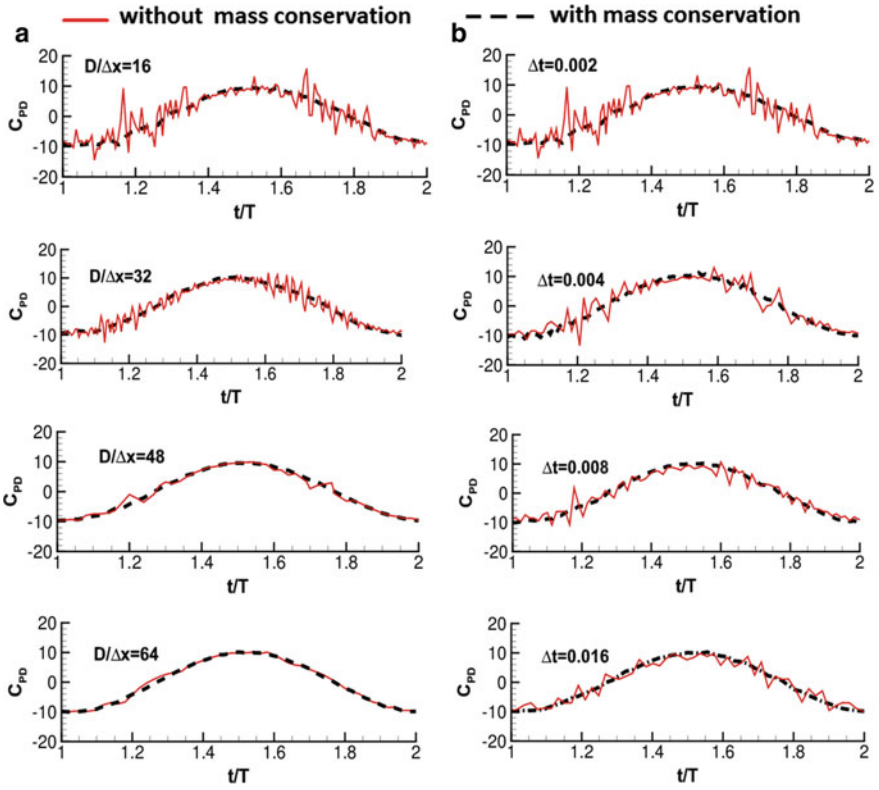


Fig. 3.13 **a** Time history of CPD with different grid size keeping time step equal to 0.002. **b** Time history of CPD with different time step keeping grid size equal to $D/\Delta x = 16$. Reproduced with permission from Kumar and Roy (2016)

Kumar and Roy (2016). The behavior of time history of the pressure drag coefficient ($C_{\mathrm{PD}} = \frac{F_D}{0.5\rho D^3 f^2}$) with four different grid sizes and time steps is presented in Fig. 3.13. Figure 3.13 shows that the coupled MAC-SOLA scheme (i.e., with mass conservation in intercepted cells) controls the spurious pressure fluctuation better than a scheme in which mass conservation at the intercepted cell (through SOLA steps) is not insured. Further, a smooth Fourier curve is fitted over the temporal variation of the pressure drag, and the root means square of the difference from the observed value is computed. The variation of this root mean square error (RMSE) is presented in Fig. 3.14. It is observed that the mass conserved scheme is able to reduce the RMSE approximately by five times in comparison with non-conserving scheme Fig. 3.14.

Overall, it inferred that better control over pressure fluctuations is achieved with mass conserving coupled MAC-SOLA scheme.

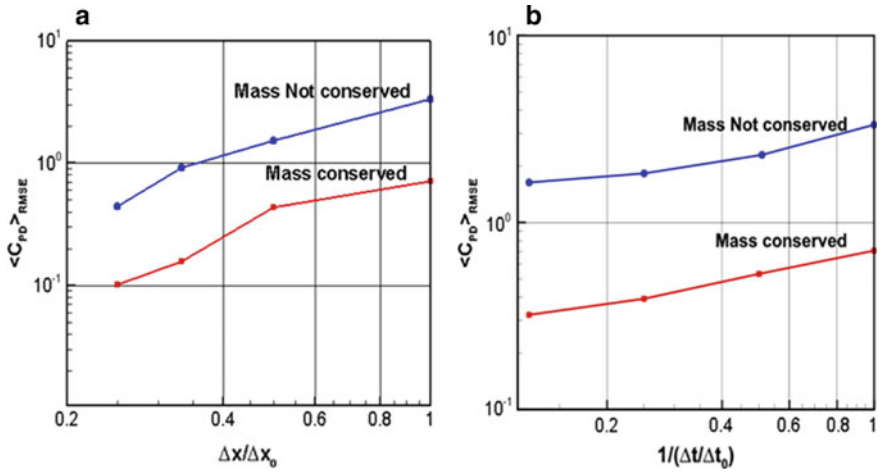


Fig. 3.14 RMSE of C_{PD} for different **a** grid size and **b** time step. Reproduced with permission from Kumar and Roy (2016)

3.5.2 Performance Results

3.5.2.1 Performance of Search Algorithm

The performance of search algorithm execution on single core Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz is compared with code execution on Tesla P100-PCIE- 12GB GPU. The compilers used are GNU for the CPU execution and pgf90 for the code execution on GPU. Figure 3.15 shows the comparison of search algorithm for both fixed and oscillating cylinder cases in a $35 \times 10D$ box (D = cylinder diameter). The Eulerian grid size used in the problem is $511 \times 211 \times 11$. An in-house grid generator is used to generate a non-uniform grid with dense clustering of nodes in the vicinity of cylinder. In case of confined search, a $31 \times 41 \times 11$ bounding box around the cylinder is used. It can be observed that speedup for both full and confined search in case of flow past a fixed cylinder is promising, but the performance degrades when the cylinder is oscillating. Since the search algorithm for moving body uses selective retagging, the scope of parallelism in performing search is minimized. Also, the time taken for the data copy/movement between GPU and CPU becomes significant. It is noted that OpenACC directives provide a speedup of more than $\approx 10\times$ in performing search.

3.5.2.2 Performance of MAC-SOLA Algorithm

The pressure correction Poisson equation to satisfy the mass conservation is solved using Jacobi and ω-SOR. Figure 3.16 compares the IB solver times for both the

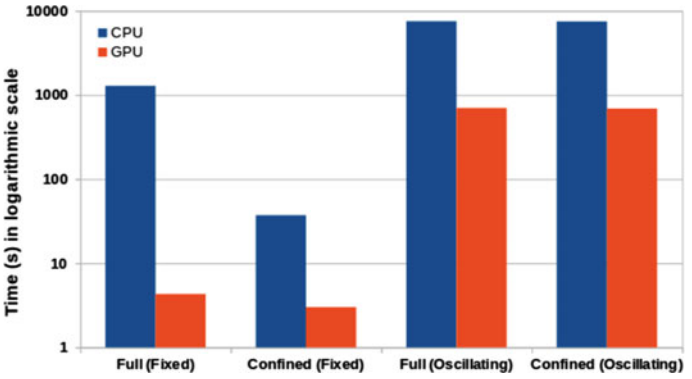


Fig. 3.15 Performance of search algorithm

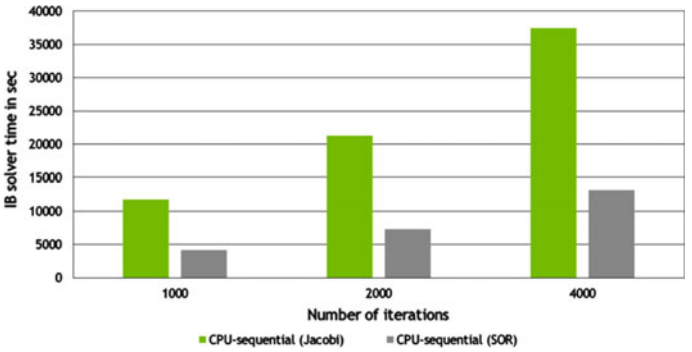


Fig. 3.16 Performance of MAC-SOLA algorithm. Reproduced with permission (Raj et al. 2018)

methods for flow past a cylinder on a Cartesian grid of size $462 \times 352 \times 9$. We see that the sequential IB solver with the SOR method for ceqcp is $2.9\times$ faster than the one with the Jacobi method. However, SOR is not easily amenable to parallelization (Pang et al. 2015). In SOR, there is a dependency of the (i,j,k) iteration on $(i-1,\ j-1,\ k-1)$ in a way that parallelism is limited to along the diagonals (in a diagonal scan). This limited degree of parallelism does not saturate the memory bandwidth, and the parallelized SOR solver performs worse than the parallel Jacobi solver. Hence, in the accelerated solver, we continue to use the Jacobi method for solving the mass continuity equations.

3.5.2.3 Performance of MAC-SOLA IBM Solver

The performance of the OpenACC accelerated IB solver on both multicore CPU and GPU architectures is analyzed. To compile the solver for a multicore CPU, we use the compilation flag -ta=multicore, while to compile for a GPU (Tesla P100), we

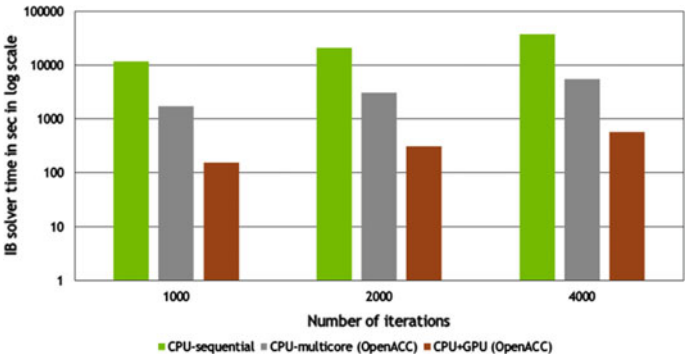


Fig. 3.17 Performance of MAC-SOLA IBM solver. Reproduced with Permission from Raj et al. (2018)

use the flag -ta=tesla:cc60. Our experiments are executed on a node with an Intel Xeon E5-2698 v3@2.3 GHz, dual socket 16 core CPU with 256 GB RAM, running CentOS 7.2. The CUDA and PGI versions used are CUDA 9.0.176 and PGI 17.10, respectively. We run our tests on three GPU architectures: Tesla P100, Tesla V100 and Tesla K80. Figure 3.17 shows the performance of the accelerated solver on GPU and multicore CPU architectures in comparison with the sequential solver. We see that accelerated solver run on the Tesla P100 GPU is 70x faster than the sequential solver. The solver run on the P100 GPU is also 10x faster than the accelerated solver run on the multicore CPU. The IB solver is memory intensive (memory operations are twice as many as flops). The large speedups of the GPU accelerated solver over the sequential solver are due to both the larger flops and memory bandwidth offered by the GPU. As the theoretical peak memory bandwidth of the Tesla P100 (720 GB/s) is about 10x more than the peak memory bandwidth of Intel Xeon E5-2698 (68 GB/s), we see large speedups on the GPU as compared to the multicore CPU.

3.5.3 Fixed: Flow Inside an S-bend Pipe

The three-dimensional flow inside a S-bend pipe having sweep angles of 22.5° (depicted in Fig. 3.18) is simulated Kumar et al. (2016). The simulation was run under the following conditions: Re = 790 (based on pipe diameter); uniform inlet velocity; no-slip conditions at enclosing wall; Orlanski outflow condition at outlet and grid size of D/40. The axial velocity profiles along the curvature plane at the three different cross sections (i.e., sections: AA, BB and CC) of the S-bend channel are plotted and compared with the corresponding results of Taylor et al. (2020) in Fig. 3.19. The computed results are in better agreement and present the efficacy of the coupled MAC-SOLA scheme.

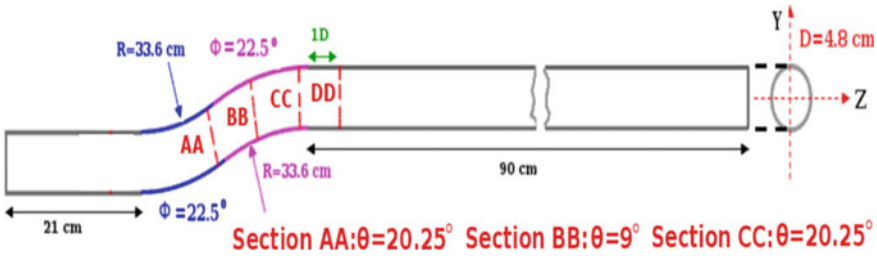


Fig. 3.18 Configuration of S-bend pipe with two successive bends with sweep angles of 22.5° over which simulation is performed. Reproduced with permission from Kumar et al. (2016)

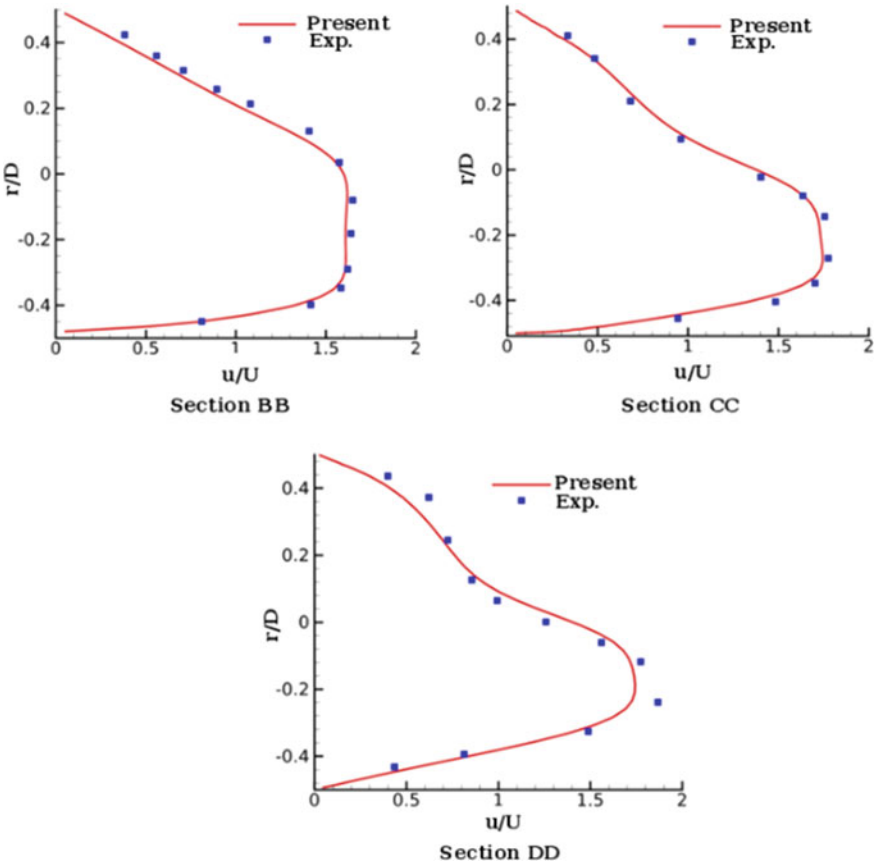


Fig. 3.19 Comparison of the axial velocity profiles in the curvature plane of S-bend with 22.5° sweep angle with experimental result (Taylor et al. 2020). Reproduced with permission from Kumar et al. (2016)

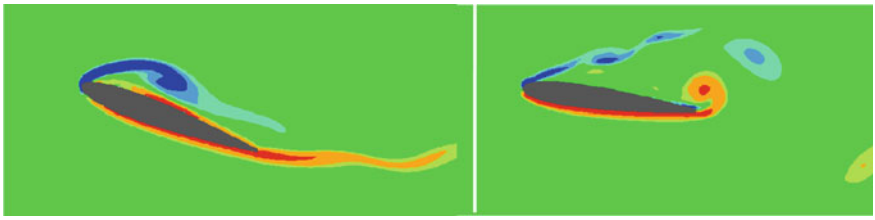


Fig. 3.20 Vorticity contours for oscillating airfoil at: $\alpha = 19.48°$ and $\alpha = 7.68°$. Reproduced with permission from Kumar and Roy (2016)

3.5.4 Moving: Oscillating Airfoil

In order to assess the efficacy of MAC-SOLA scheme, its performance in case of moving boundary problem is needed to be studied. For this, Mehta's experimental simulation over oscillating airfoil is simulated Mehta (1976). The oscillation of airfoil is governed by $\alpha = 10° - 10°\cos\omega t$, where $\omega = \frac{2ku_\infty}{c}$, $k = 0.5$, and it oscillates around the one-fourth chord axis. The Re is kept equal to 5000. The coefficients of lift at $\alpha = 19.48°$ and $\alpha = 7.68°$ are computed as 1.446 and 0.978, respectively, which are comparable to Mehta's reported values of 1.453 and 0.973. This agreement in the values of unsteady lift coefficient also affirms the the utility of this scheme in controlling the spurious pressure fluctuation over accelerating bodies. Further, it also exhibits the capability of present MAC-SOLA based IBM implementation in prediction of the unsteady aerodynamics. The vorticity iso-contours during upstroke and downstroke are shown in Fig. 3.20. The results also depict the formation of leading edge vortex during upstroke and subsequent departure of vortex while downstroke (also known as dynamic stall).

3.6 Conclusions

This chapter has presented a detailed discussion on implementation of the sharp interface immersed boundary method in which boundary conditions are accurately implemented over the complex/moving boundary while strongly enforcing the mass conservation at the intercepted cells. A coupled MAC-SOLA algorithm is presented which is simple to implement yet does not add any extra computational overhead. Accuracy in terms of global and local mass conservation is demonstrated. We further show that this algorithm can efficiently control the spurious pressure oscillations in the vicinity of the immersed surface even when using a coarse mesh and finer time steps. Further, the parallelization of the scheme is discussed. We have demonstrated optimization of the solver using GPGPU acceleration through OpenACC directives. Parallelizations of the computed heavy parts like IBM search or tagging and pressure correction solvers are discussed in detail.

Acknowledgements The authors would like to sincerely thank the co-authors of Raj et al. (2018)—Nagavijayalakshmi Vydyanathan and Bharatkumar Sharma. We would also like to express our gratitude toward Computer & Fluids and IEEE for granting us permission to use the excerpts and figures from papers Kumar et al. (2016), Kumar and Roy (2016), Raj et al. (2018).

References

Anupindi K, Delorme Y, Shetty DA, Frankel SH (2013) A novel multiblock immersed boundary method for large eddy simulation of complex arterial hemodynamics. J Comput Phys 254:200–218

Bao Y, Donev A, McQueen DM, Peskin CS (2017) An immersed boundary method with divergence-free velocity interpolation and force spreading. J Comput Phys 347:183–206

Choi JI, Oberoi RC, Edwards JR, Rosati JA (2007) An immersed boundary method for complex incompressible flows. J Comput Phys 224:757–784

CUDA c programming guide, http://docs.nvidia.com/cuda/cuda-c-programmingguide/index.html

CUSP. https://developer.nvidia.com/cusp

de Zélicourt D, Ge L, Wang C, Sotiropoulos F, Gilmanov A, Yoganathan A (2009) Flow simulations in arbitrarily complex cardiovascular anatomies-an unstructured cartesian grid approach. Comput Fluids 38:1749–1762

De AK (2018) A diffuse interface immersed boundary method for complex moving boundary problems. J Comput Phys 366:226–251

DeLeon ISR, Felzien K (2012) Toward a GPU-accelerated immersed boundary method for wind forecasting over complex terrain. In: Proceedings of the ASME 2012 fluids engineering summer meeting

Gilmanov A, Sotiropoulos F (2005) A hybrid cartesian/immersed boundary method for simulating flows with 3d, geometrically complex, moving bodies. J Comput Phys 207:457–492

Gresho PM, Sani RL (1987) On pressure boundary conditions for the incompressible navier-stokes equations. Int J Numer Methods Fluids 7:1111–1145

Hartmann H, Derksen J, Van den Akker H (2006) Mixing times in a turbulent stirred tank by means of les. AIChE J 52:3696–3706

Hirt C, Nichols B, Romero N (1975) Sola: a numerical solution algorithm for transient fluid flows. NASA STI/Recon Technical Report N 75:32418

Hou TY, Shi Z (2008) An efficient semi-implicit immersed boundary method for the navier-stokes equations. J Comput Phys 227:8968–8991

Iaccarino G, Verzicco R (2003) Immersed boundary technique for turbulent flow simulations. Appl Mech Rev 56:331–347

Kamakoti R, Shyy W (2004) Evaluation of geometric conservation law using pressure-based fluid solver and moving grid technique. Int J Numer Methods Heat Fluid Flow 14:851–865

Kang S, Iaccarino G, Ham F, Moin P (2009) Prediction of wall-pressure fluctuation in turbulent flows with an immersed boundary method. J Comput Phys 228:6753–6772

Khalighi B, Jindal S, Johnson JP, Chen KH, Iaccarino G (2009) Validation of the immersed boundary CFD approach for complex aerodynamic flows. In: Lecture notes in applied and computational mechanics, pp 21–38

Kraus J, Schlottke M, Adinetz A, Pleiter D (2014) Accelerating a c++ CFDcode with OpenAcc. In: 2014 first workshop on accelerator programming using directives, pp 47–54

Kumar M, Roy S (2016) A sharp interface immersed boundary method for moving geometries with mass conservation and smooth pressure variation. Comput Fluids 137:15–35

Kumar M, Roy S, Ali MdS (2016) An efficient immersed boundary algorithm for simulation of flows in curved and moving geometries. Comput Fluids 129:159–178

Layton SK, Krishnan A, Barba LA (2011) cuIBM A GPU-accelerated immersed boundary method. Comput Res Repository
Lee P, Grifth BE, Peskin CS (2010) The immersed boundary method for advection-electro diffusion with implicit time stepping local mesh refinement. J Comput Phys 229:5208–5227
Liao CC, Chang YW, Lin CA, McDonough J (2010) Simulating flows with moving rigid boundary using immersed-boundary method. Comput Fluids 39:152–167
Liu C, Hu C (2014) An efficient immersed boundary treatment for complex moving object. J Comput Phys 274:654–680
Mehta UB (1976) Dynamic stall of an oscillating airfoil. AGARD CP 227:1–32
Mittal R, Iaccarino G (2005) Immersed boundary methods. Ann Rev Fluid Mech 37:239–261
Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, Von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227(10):4825–4852
Muldoon F, Acharya S (2008) A divergence-free interpolation scheme for the immersed boundary method. Int J Numer Methods Fluids 56:1845–1884
OpenACC Programming and Best Practices Guide. https://www.openacc.org/resources
OpenACC: More science, less programming, https://www.openacc.org
Orlanski I (1976) A simple boundary condition for unbounded hyperbolic flows. J Comput Phys 21:251–269
Pang R, Xu J, Zhang Y (2015) Parallel solving method of SOR based on the numerical marine forecasting model. 15th IEEE/ACM international symposium on cluster, cloud and grid computing, pp 733–736
Peskin CS (1972) Flow patters around heart valves: a numerical method. J Comput Phys 10:252–271
Picano F, Breugem WP, Brandt L (2015) Turbulent channel flow of dense suspensions of neutrally buoyant spheres. J Fluid Mech 764:463–487
Posa A, Vanella M, Balaras E (2017) An adaptive reconstruction for Lagrangian, direct-forcing, immersed-boundary methods. J Comput Phys 351:422–436
Problems in ray tracing. http://www.computer-graphics.se/TSBK07-files/pdf13/13b.pdf
Raj A, Roy S, Vydyanathan N, Sharma B (2018) Acceleration of a 3D immersed boundary solver using OpenACC. In: IEEE 25th international conference on high performance computing eorkshops (HiPCW)
Roy S, Acharya S (2012) Scalar mixing in a turbulent stirred tank with pitched blade turbine: role of impeller speed perturbation. Chem Eng Res Des 90:884–898
Seo JH, Mittal R (2011) A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. J Comput Phys 230:7347–7363
Taira K, Colonius T (2007) The immersed boundary method: a projection approach. J Comput Phys 225:2118–2137
Taylor A, Whitelaw J, Yianneskis M (1984) Developing flow in s-shaped ducts. 2: circular cross-section duct. Final Report Imperial College of Science and Technology, London (England). Department of Mechanical Engineering, 1
The Immersed Boundary Approach to Fluid Flow Simulation–Article–ANSYS advantage–V3 I2. https://www.ansys.com/en-in/resource-library/article/immersed-boundary-approach-to-fluid-flow-simulation-ansys-advantage-v3-i2
The OpenACC application programming interface, version 2.6, November 2017. https://www.openacc.org/resources
Thrust, https://developer.nvidia.com/thrust
Tutkun B, Edis FO (2017) An implementation of the directforcing immersed boundary method using gpu power. Eng Appl Comput Fluid Mech 11(1):15–29
Tyagi M, Acharya S (2005) Large eddy simulation of turbulent flows in complex and moving rigid geometries using the immersed boundary method. Int J Numer Methods Fluids 48:691–722
Udaykumar H, Mittal R, Rampunggoon P, Khanna A (2001) A sharp interface cartesian grid method for simulating flows with complex moving boundaries. J Comput Phys 174:345–380

Verzicco R, Mohd-Yusof J, Orlandi P, Haworth D (2000) Large eddy simulation in complex geometric configurations using boundary body forces. AIAA J 38:427–433

Yang X, Zhang X, Li Z, He GW (2009) A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations. J Comput Phys 228:7821–7836

Ye T, Mittal R, Udaykumar H, Shyy W (1999) An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. J Comput Phys 156:209–240

Yildirim B, Lin S, Mathur S, Murthy JY (2013) A parallel implementation of fluid-solid interaction solver using an immersed boundary method. Comput Fluids 86:251–274

Chapter 4
Coupling the Curvilinear Immersed Boundary Method with Rotation-Free Finite Elements for Simulating Fluid–Structure Interaction: Concepts and Applications

Anvar Gilmanov, Henryk Stolarski, and Fotis Sotiropoulos

4.1 Introduction

Unsteady fluid–structure interaction (FSI) problems taking place in geometrically complex domains and involving large deformations of three-dimensional, thin structures are encountered in a broad range of engineering and biological problems across a range of Reynolds numbers and flow regimes. Examples range from inflating parachutes and flow-activated energy harvesting devices, to swimming aquatic organisms, to native as well as prosthetic heart valves, to name a few. The inherent complexity of such problems along with the highly nonlinear nature of the ensuing FSI, which is associated primarily with the large deformations of the solid, present unique challenges to numerical methods. Such challenges arise from, among others: (i) the need to model geometric and constitutive nonlinearities of the solid bodies; (ii) the often arbitrary complexity of the dynamically evolving flow domains, due to the arbitrarily large amplitude of the deformation thin flexible structures may undergo; and (iii) the challenges in obtaining robust and efficient FSI algorithms, especially in problems with low mass ratios (Sotiropoulos and Yang 2014; Baek and Karniadakis 2012) which are commonly encountered in cardiovascular flow simulations. These challenges along with recently developed approaches for tackling them constitute the main focus of this chapter.

There are two general approaches typically used for simulating complex flows with deformable boundaries: (1) the boundary conforming arbitrary Lagrangian Eulerian (ALE) approach; and (2) immersed boundary (IB) methods. The ALE

A. Gilmanov (✉) · H. Stolarski
University of Minnesota, Minneapolis, MN 55455, USA
e-mail: gilmanov.anvar@gmail.com

F. Sotiropoulos
Stony Brook University, Stony Brook, NY 11794-2200, USA

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_4

approach (Hirt et al. 1974; Donea et al. 1982) is well suited for resolving near-wall viscous regions in high Reynolds number flows due to its inherent body-fitted mesh structure that conforms to boundaries at all times. However, for significant movement of the boundaries, ALE methods are cumbersome to apply to problems with large deformations since they require frequent remeshing in order to prevent the mesh from becoming severely distorted. The remeshing procedure is computationally expensive making ALE methods inefficient in complex three-dimensional problems. Fixed, non-boundary conforming, grid methods provide another alternative to solving problems with deformable boundaries and complex geometry. Such methods are generally referred to as immersed boundary (IB) methods and are especially attractive for simulations of complex flows in engineering and biology because they do not require remeshing and can readily handle arbitrarily large deformations of the structures. The various types of IB methods have been recently reviewed by Sotiropoulos and Yang (2014). The interested reader is referred to this paper as well as the earlier review by Mittal and Iaccarino (2005) for details. Promising approaches that enhance the capabilities of IB methods in the simulation of fluid flow interacting with moving/deformable bodies at high Re numbers are methods involving adaptive mesh refinements (Vanella et al. 2010; Angelidis et al. 2016).

In this chapter, we focus our review of the literature exclusively on IB numerical approaches proposed for handling FSI of flexible structures in complex domains. We pay special attention to the distinction between discretization techniques used to handle the flow and those applied to structural governing equations, since a range of formulations have been proposed in the past. These include pure finite-difference (FD) (Griffith et al. 2009; Wiens and Stockie 2015; Zhu and Peskin 2002; Le et al. 2009; Luo et al. 2008) or finite element (FE) (Dettmer and Perić 2006; Barker and Cai 2010; Bazilevs et al. 2012) methods for both the flow and structural equations as well as mixed formulations combining FD (or finite volume) discretization for the flow with FE for the structural equations (Zheng et al. 2010; Farhat and Lakshminarayan 2014).

Diffused interface IB methods use FD for both the fluid and structural solvers (Griffith et al. 2009). In this approach, the loading on the structural surface, due to interaction with the fluid, is introduced by appropriately defined body forces in the momentum fluid equations. A number of successful applications of such methods, which we shall refer to herein as IB-FD-FD methods for their use of FD discretization for both the fluid and solid equations, have been reported over the years (Wiens and Stockie 2015; Zhu and Peskin 2002; Le et al. 2009). The accuracy of such methods can be improved by incorporating local mesh refinement as was done in Griffith et al. (2009). One potential difficulty with this class of methods, however, arises from treating the solid surface as diffused interface, which complicates the accurate calculation of the wall shear stress field on the surface. Yet, such detailed calculations may be required in cardiovascular flow problems, such as heart valve flow simulations, in which complex wall shear stress patterns on the valve leaflets have been linked with increased potential for aortic valve diseases and other aortopathies (Ge and Sotiropoulos 2010).

A sharp interface IB method using FD formulations for both the flow and structural equations was proposed by Luo et al. (2008). This method was formulated for linear viscoelastic solids and applied to simulate two-dimensional FSI in laryngeal aerodynamics. The same formulation was later modified to incorporate a FE formulation for the structural equations and applied to simulate FSI of a high mass ratio 3D flapping wing at very low Reynolds number (Re = 50) (Luo et al. 2010). Tian et al. (2014) further extended this method to simulate several complex FSI problems at low Reynolds numbers (Re $\sim$ 102). An ALE formulation utilizing the so-called embedded boundary approach was proposed by Farhat and Lakshminarayan (2014) for solving compressible FSI problems for external aerodynamics applications at high Reynolds numbers. This approach employs finite volume discretization for the fluid equations with finite elements for the structural equations. While this approach can work well for structures in unbounded domains, remeshing difficulties may arise when the structure is embedded within a complex confined domain. A pure finite-element-based formulation, for both the flow and the structural equations, was recently proposed by Kamensky et al. (2015). This method employs the *immersogeometric* FSI approach and was applied to simulate FSI of a bioprosthetic heart valve in a straight aorta.

In FSI simulations of biological tissues, e.g., heart valve leaflet interaction with blood flow, it is critical to use a relevant and efficient structural model that is able to realistically represent the deformation of the tissue under loads imposed by the pulsatile blood flow. Such undertaking, however, is not a trivial task since the large deformations of the tissue and its concomitant geometric nonlinearity pose major modeling challenges. To circumvent these challenges recent studies attempting to simulate FSI of tissue valves chose to either use simplified membrane-like materials (Borazjani 2013) or treat the valve leaflets as thick bodies (Tian et al. 2014). However, biological tissues of leaflets are normally thin and they exhibit significant bending. Therefore, a shell model for the solid body is a more appropriate choice (Kamensky et al. 2015; Sacks et al. 2009). Most finite element (FE) methodologies for handling shells, however, are computationally very demanding as they employ two or three nodal rotations alongside with three nodal translations, i.e., 5 or 6 degree of freedom per node. An exhaustive review of this large body of literature is beyond the scope of this chapter, but the reader is referred to a number of recent review papers on the topic (Stolarski et al. 1995; Gal and Levy 2006). Note that the efficiency of the FE shell model becomes of paramount concern in FSI simulations of complex problems where the need to couple the fluid and structural solvers together can dramatically increase the computational cost per time step. For that, in our work we have selected to adapt and incorporate in the FSI methodology a previously developed nonlinear, rotation-free triangular shell element formulation (Stolarski et al. 2013), which has already been shown to provide accurate and robust solutions of various thin shell FE problems. We have successfully coupled such an approach with the sharp interface curvilinear IB (CURVIB) method, previously developed by our group (Ge and Sotiropoulos 2007) to simulate FSI problems (Gilmanov et al. 2015, 2018). In this chapter, we review the basic features of this novel CURVIB-FE-FSI formulation.

The CURVIB method employs second-order accurate, central finite differencing discretization for the flow equations along with an efficient fractional step approach for satisfying the discrete continuity equation to machine zero in curvilinear grids. This method has also been extended to carry out large-eddy simulation (LES) of turbulent flows using wall models for reconstructing boundary conditions at the immersed boundary nodes (Kang et al. 2011). The LES version of the CURVIB method has been validated extensively for a broad range of complex turbulent flows. Some recent examples include: turbulent flow past an axial flow turbine in an open channel (Kang et al. 2014); open-channel turbulence interacting with a mobile sediment bed (Khosronejad and Sotiropoulos 2014), and complex rigid structures interacting with a free surface (Calderer et al. 2014). As such the CURVIB method provides an efficient and accurate approach for simulating geometrically complex flows across a range of Reynolds numbers. Furthermore, since the CURVIB method employs unstructured triangular meshes to discretize immersed boundaries, the method is ideally suited for coupling it with our efficient rotation-free FE shell model (Stolarski et al. 2013), which is ideally suited for handling FSI problems involving arbitrarily large deformations. To enable this coupling, we report herein on a number of algorithmic advances and significant improvements of our previously developed methodology. Our FE solver is highly efficient and versatile for thin bodies—it can be applied in analysis of a variety of structures including engineering structures such as shells, plates, beams and may incorporate various material properties, including those characterizing biological tissues such as heart valves and arterial walls.

In this chapter, we present the recently developed methodology and demonstrate its ability to simulate very challenging FSI problems involving large amplitude oscillations. The first problem is that of an inverted elastic flag, recently studied experimentally by Kim et al. (2013), which is especially challenging because: (1) the flow occurs at high Reynolds number and requires implementing the resulting CURVIB-FE-FSI formulation in conjunction with LES; and (2) depending on the elasticity of the flag the FSI problem exhibits dynamically rich variety of solutions (Kim et al. 2013). To the best of our knowledge, the first numerical solution of that problem was reported in Gilmanov et al. (2015). Here, we report simulations for a set of parameters under which the flag undergoes periodic oscillations and show that the computed motion of the flag is in excellent agreement with the measurements. In the second application problem, we demonstrate the ability of the coupled CURVIB-FE-FSI method to simulate the FSI of a tri-leaflet valve in an anatomic aorta. Our simulations capture the rich 3D vorticity dynamics during the opening and closing of the valve leaflets.

The chapter is organized as follows. In Sect. 4.2, we describe the governing equations for both fluid and solid structures. In Sect. 4.3, we present the numerical approach used to solve the coupled system of fluid and solid equations with appropriate boundary conditions. In Sect. 4.4, the flapping of an inverted flag is presented. In this section, we also demonstrate the applicability of the proposed FSI approach to simulate pulsatile blood flow in an anatomic aorta with a tri-leaflet heart valve using

both isotropic and nonlinear anisotropic materials (Gilmanov et al. 2018). Finally, in the Sect. 4.5, we summarize the major features of he presented approach.

4.2 Governing Equations

We consider FSI of a deformable body Ω_s submerged in an incompressible fluid occupying a volume Ω_f bounded by $\partial\Omega_f$, the method is applicable to multiple deformable *thin* bodies but for the ease of presentation and without loss of generality we present the method for a single body.

In what follows, we use bold symbols for vectors and bold underlined symbols for tensors and matrices. The regular and italic symbols are reserved for scalar and tensor components, respectively. The overbar notation indicates known and/or prescribed values.

4.2.1 The Equations for the Fluid Domain

In general, fluid boundaries can be presented as consisting of three non-overlapping parts: $\partial\Omega_f = \Gamma_f^N \cup \Gamma_f^D \cup \Gamma^{fsi}$. Here, Γ_f^D and Γ_f^N are the stationary boundaries in which Dirichlet and/or Neumann boundary conditions are specified. Γ^{fsi} is the interface between the fluid domain and the solid domain, i.e., the moving interface the configuration of which needs to be determined by solving the FSI problem.

The equations governing the motion of Newtonian incompressible fluid in a domain Ω_f the Navier–Stokes and continuity equations, which read in vector/tensor notation as follows:

$$\begin{aligned} \rho_f \frac{d\boldsymbol{v}}{dt} &= \nabla \cdot \underline{\boldsymbol{\sigma}}_f \text{ in } \Omega_f, \\ \nabla \cdot \boldsymbol{v} &= 0 \text{ in } \Omega_f. \end{aligned} \tag{4.1}$$

In the above equations, ρ_f is the mass density of the fluid, d/dt is the material or Lagrangian time derivative, $\boldsymbol{v}$ is the fluid velocity vector, and $\underline{\boldsymbol{\sigma}}_f$ is the fluid stress tensor. The above equations are subjected to various boundary conditions for the velocity $\boldsymbol{v}$ on the various segments comprising the fluid boundary. For example, on the Dirichlet portion of the boundary Γ_f^D Dirichlet boundary conditions and on the Neumann segment of the boundary Γ_f^N, a stress boundary condition of the following form may be applied:

$$\boldsymbol{v} = \bar{\boldsymbol{v}} \text{ on } \Gamma_f^D \quad \underline{\boldsymbol{\sigma}}_f \cdot \boldsymbol{n}_f = \bar{\boldsymbol{t}}_f \text{ on } \Gamma_f^N \tag{4.2}$$

where $\bar{\boldsymbol{v}}$ and $\bar{\boldsymbol{t}}_f$ are known functions, $\boldsymbol{n}_f$ is the normal unit vector to the Γ_f^N boundary. For FSI problems, the immersed deformable body has its own displacement field $\boldsymbol{u}$,

velocity field $\dot{\boldsymbol{u}}$, and stress field $\underline{\boldsymbol{\sigma}}_{\mathrm{s}}$. On Γ^{fsi} the velocity field and the normal stress field must be continuous. This physical requirement gives rise to the following set of boundary conditions on the FSI segment of the fluid boundary:

$$\boldsymbol{v} = \dot{\boldsymbol{u}}, \quad \underline{\boldsymbol{\sigma}}_{\mathrm{f}} \cdot \boldsymbol{n}_{\mathrm{f}} = \underline{\boldsymbol{\sigma}}_{\mathrm{s}} \cdot \boldsymbol{n}_{\mathrm{s}} \text{ on } \Gamma^{\mathrm{fsi}} \tag{4.3}$$

Here, $\boldsymbol{n}_{\mathrm{f}}$ and $\boldsymbol{n}_{\mathrm{s}}$ are the local normal unit vectors on the fluid and solid interfaces, respectively. Note, therefore, that on the Γ^{fsi} segment of the boundary both Dirichlet and Neumann conditions must be satisfied (given by Eq. 4.3) so that the problem is well posed and the Navier–Stokes Eqs. (4.1) supplied with boundary conditions (4.2) on $\Gamma_{\mathrm{f}}^{\mathrm{D}}$ and $\Gamma_{\mathrm{f}}^{\mathrm{N}}$ can be solved.

To facilitate the subsequent presentation of the FSI algorithm, we denote the governing equations for the fluid domain as an operator $\boldsymbol{\mathcal{F}}$, which receives the input information from the boundary conditions and yields the pressure p and velocity field $\boldsymbol{v}$ inside the fluid domain Ω_{f} as follows:

$$(p, \boldsymbol{v}) = \boldsymbol{\mathcal{F}}\left(\bar{\boldsymbol{v}}, \overline{\boldsymbol{t}_{\mathrm{f}}}, \dot{\boldsymbol{u}}, \underline{\boldsymbol{\sigma}}_{\mathrm{s}}\right) \text{ in } \Omega_{\mathrm{f}} \tag{4.4}$$

here $\dot{\boldsymbol{u}}$ and $\underline{\boldsymbol{\sigma}}_{\mathrm{s}}$ are applied at the boundary Γ^{fsi}. Equation (4.4), therefore, should be viewed as the operator notation for Eqs. (4.1–4.3).

4.2.2 The Equations for the Solid Domain

In the solid domain, we use the Lagrangian viewpoint to describe the motion of the solid undergoing large deformations. In this approach, the current position $\boldsymbol{r}$ of a material point at time t is related to its position $\boldsymbol{R}$ at the reference configuration by the mapping $\boldsymbol{\Phi}$:$\boldsymbol{r} = \boldsymbol{\Phi}(\boldsymbol{R})$. The gradient of that transformation (the so-called deformation gradient) is therefore: $\boldsymbol{F} = \partial \boldsymbol{\Phi} / \partial \boldsymbol{R}$. The displacement and velocity of a material point are defined as:

$$\boldsymbol{u} = \boldsymbol{r} - \boldsymbol{R}, \quad \dot{\boldsymbol{u}} = \mathrm{d}\boldsymbol{u}/\mathrm{d}t \tag{4.5}$$

The momentum equations for the solid part, formulated in the current configuration, have the following form (Kang et al. 2011):

$$\rho_{\mathrm{s}} \frac{\mathrm{d}\dot{\boldsymbol{u}}}{\mathrm{d}t} = \nabla \cdot \underline{\boldsymbol{\sigma}}_{\mathrm{s}} \text{ in } \Omega_{\mathrm{s}}, \tag{4.6}$$

where ρ_{s} is the current mass density of the material. Here, $\underline{\boldsymbol{\sigma}}_{\mathrm{s}}$ is the Cauchy stress tensor for the solid structure, with the symbol ∇ representing the gradient operator in the current configuration. The boundary of the solid structure can be represented as sum of non-overlapping parts $\partial\Omega_{\mathrm{s}} = \Gamma_{\mathrm{s}}^{\mathrm{N}} \cup \Gamma_{\mathrm{s}}^{\mathrm{D}} \cup \Gamma^{\mathrm{fsi}}$, where the indices D and N denote boundaries with Dirichlet and Neumann conditions, respectively:

$$\dot{\boldsymbol{u}} = \bar{\dot{\boldsymbol{u}}} \quad \text{on} \quad \Gamma_{\mathrm{s}}^{\mathrm{D}}, \quad \underline{\boldsymbol{\sigma}}_{\mathrm{s}} \cdot \boldsymbol{n}_{\mathrm{s}} = \bar{\boldsymbol{t}}_{\mathrm{s}} \quad \text{on} \quad \Gamma_{\mathrm{s}}^{\mathrm{N}} \tag{4.7}$$

where $\Gamma_{\mathrm{s}}^{\mathrm{D}}$ and $\Gamma_{\mathrm{s}}^{\mathrm{N}}$ represents the portions of the surface of the body in its current configuration where Dirichlet and Neumann conditions are applied, respectively, $\bar{\boldsymbol{t}}_{\mathrm{s}}$ is a traction vector acting on the surface, $\boldsymbol{n}_{\mathrm{s}}$ is a unit normal to the boundary and $\dot{\boldsymbol{u}}$ is the velocity prescribed on the surface.

For FSI problems, additional boundary conditions must be implemented on the Γ^{fsi}:

$$\dot{\boldsymbol{u}} = \boldsymbol{v} \text{ on } \Gamma^{\mathrm{fsi}}, \quad \underline{\boldsymbol{\sigma}}_{\mathrm{s}} \cdot \boldsymbol{n}_{\mathrm{s}} = \boldsymbol{t}_{\mathrm{f}} \text{ on } \Gamma^{\mathrm{fsi}} \tag{4.8}$$

here Γ^{fsi} is part of the moving structure surface the configuration of which needs to be determined by solving the FSI problem, $\boldsymbol{t}_{\mathrm{f}} = \underline{\boldsymbol{\sigma}}_{\mathrm{f}} \cdot \boldsymbol{n}_{\mathrm{f}}$ is a traction vector which acts on this part of surface from the fluid, $\underline{\boldsymbol{\sigma}}_{\mathrm{f}}$ and $\boldsymbol{n}_{\mathrm{f}}$ are the stress tensor and surface normal unit vector from the fluid. We will discuss later how to define the traction vector for thin surfaces.

The solid momentum equations and the boundary conditions can be recast in terms of an operator $\mathfrak{H}$, which incorporates both the (kinematic and dynamic) boundary conditions and constitutive equations to yield the velocity $\dot{\boldsymbol{u}}$ and displacement field $\boldsymbol{u}$

$$(\boldsymbol{u}, \dot{\boldsymbol{u}}) = \mathfrak{H}\left(\bar{\boldsymbol{v}}, \bar{\boldsymbol{t}}, \boldsymbol{v}, \boldsymbol{t}_{\mathrm{f}}\right) \text{ in } \Omega_{\mathrm{s}}, \tag{4.9}$$

here $\boldsymbol{v}$ and $\boldsymbol{t}_{\mathrm{f}}$ are applied at the boundary Γ^{fsi} .

4.3 Numerical Algorithms for Fluid–Structure Interaction

A sharp interface IB algorithm for solving FSI problems with thin deformable structures embedded in a fluid domain requires developing and integrating the following algorithmic components: (1) an algorithm for solving the fluid flow equations (Sect. 4.3.1); (2) an algorithm for solving the thin shell structural equations (Sect. 4.3.2); (3) an approach for defining the action from the thin shell onto the surrounding fluid by identifying the IB nodes in the vicinity of the body where boundary conditions need to reconstructed (Sect. 4.3.3); (4) an approach for calculating the action from the fluid to the thin shell body computing the forces due to pressure and shear (Sect. 4.3.4); and (5) an algorithm that integrates the fluid and solid solvers into a coupled FSI formulation. In this section, we discuss the approaches we adopt in this work to develop these algorithmic components (Sect. 4.3.5).

4.3.1 The Fluid Solver $\mathcal{F}$

The fluid solver is based on the CURVIB approach (Ge and Sotiropoulos 2007) which uses the hybrid stagger/non-staggered approach originally proposed by Gilmanov and Sotiropoulos (2005) to solve the governing equations in generalized curvilinear grids (Ge and Sotiropoulos 2007). The Navier–Stokes and continuity equations (4.1) are partially transformed in generalized curvilinear coordinates and read in tensor form (repeated indices $j = 1, 3$ assumes summation) as follows:

$$\frac{\partial}{\partial \xi^j}\left(\frac{V^j}{J}\right) = 0, \quad \frac{\partial v_q}{\partial t} + C\left(v_q\right) + G_q(p) - \frac{1}{\mathrm{Re}} D\left(v_q\right) = 0, \quad q = 1, 2, 3, \tag{4.10}$$

where the Cartesian velocity vector is denoted as $\boldsymbol{v}(v_1, v_2, v_3)$, p, the pressure divided by the density ρ_f, $V^j = v_r \xi^j_{x_r}$ is the jth contravariant velocity component in the general curvilinear coordinate system $\xi(\xi_1, \xi_2, \xi_3)$, J is the Jacobian of the geometric transformation $J = \partial(\xi_1, \xi_2, \xi_3)/\partial(x_1, x_2, x_3)$, and $g^{rm} = \xi^r_{x_q} \xi^m_{x_q}$ is the contravariant metric tensor. The convective $C\left(v_q\right)$, gradient $G_q(p)$, and viscous $D\left(v_q\right)$ operators in Eq. (4.10) are defined in curvilinear coordinates as (the repeated indexes r, m imply summation over the values 1, 2, 3):

$$\begin{aligned} C\left(v_q\right) &= J \frac{\partial}{\partial \xi^r}\left(\frac{V^r}{J} v_q\right), \quad q = 1, 2, 3, \\ D\left(v_q\right) &= J \frac{\partial}{\partial \xi^r}\left(\frac{g^{rm}}{J} \frac{\partial v_q}{\partial \xi^m}\right), \\ G_q(p) &= J \frac{\partial}{\partial \xi^r}\left(\frac{\xi^r_{x_q}}{J} p\right). \end{aligned} \tag{4.11}$$

The above equations are discretized via a hybrid staggered/non-staggered approach using three-point central differencing for all spatial derivatives and integrated in time via a second-order accurate fractional step, pressure projection method. The momentum equations are solved with a Jacobian-free solver, while flexible generalized minimal residual (FGMRES) method with multigrid pre-conditioner is used to solve the Poisson equation to satisfy the discrete continuity equation to machine zero (see Ge and Sotiropoulos 2007 for details).

Complex immersed boundaries are handled using a sharp interface IB method with velocity reconstruction along the local normal to the body (Ge and Sotiropoulos 2007; Gilmanov and Sotiropoulos 2005; Borazjani et al. 2008). Some details concerning the reconstruction method for thin flexible boundaries will be provided in a subsequent section of this chapter.

The CURVIB method has been recently extended to carry out LES of turbulent flows in geometrically complex domains. The details of the LES version of our flow solver can be found in Kang et al. (2011, 2014). Here, it suffices to mention

that the dynamic Smagorinsky model (Germano et al. 1991) is used for subgrid-scale closure with three-point central, second-order accurate finite differencing for the convective terms. Boundary conditions at IB nodes in the vicinity of complex immersed boundaries are reconstructed using a wall model approach adapted for the CURVIB method by Kang et al. (2011). In this chapter, we will report the first application of the LES version of method to simulate FSI of a flexible structure at high Reynolds number.

4.3.2 *The Solid Solver $\mathfrak{H}$: Finite Element Model for Thin Shells*

The momentum equations for the solid (Eq. 4.6) can be expressed in various weak formulations using the principle of virtual work. In this work, we select the Lagrangian version of the weak form, which is related to the initial configuration, uses the second Piola–Kirchhoff stress tensor $\underline{\boldsymbol{S}}$ and the variation of the Green–Lagrange strain tensor $\underline{\boldsymbol{E}}$. By virtue of how they appear in the principle of virtual work given below, these two tensors constitute a dual set in the reference configuration representing volume V_0 bounded by the surface boundary A_0. This version of the weak form reads as follows:

$$\iiint_{V_0} \left(\delta \underline{\boldsymbol{E}}^{\mathrm{T}} \underline{\boldsymbol{S}} + \delta \boldsymbol{u}^{\mathrm{T}} \rho_{\mathrm{s}} \ddot{\boldsymbol{u}}\right) \mathrm{d}V_0 - \iint_{A_0} \delta \boldsymbol{u}^{\mathrm{T}} \boldsymbol{t}_0 \mathrm{d}A_0 = 0. \tag{4.12}$$

In the above equation, ρ_{s} is the constant density of the solid in the original configuration, $\boldsymbol{t}_0$ represent the surface loads in that configuration, and $\ddot{\boldsymbol{u}}$ is the acceleration. To focus on the essential features of the algorithm in the illustrative examples presented in this chapter the Neo–Hookean (Macosko 1994) constitutive equation is used. Thus, in any fixed, local coordinate system the stress and strain tensors are related as follows:

$$\underline{\boldsymbol{S}}^{\mathrm{loc}} = \underline{\boldsymbol{D}}^{\mathrm{loc}} \underline{\boldsymbol{E}}^{\mathrm{loc}} \tag{4.13}$$

with

$$\underline{\boldsymbol{D}}^{\mathrm{loc}} = \frac{Y}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}, \tag{4.14}$$

where Y is the Young's modulus and ν is the Poisson's ratio, index loc indicates that constitutive equation is described in a local Cartesian system. Having the stresses defined in that specific system, they can be transformed to any other system by the usual transformation roles for tensors.

Although, to simplify our exposition of the concepts, in the examples presented here the constitutive equations defined above are used, the overall methodology described here has been recently combined with complex material behavior, more appropriate for biological applications. Gilmanov et al. (2016) incorporated a general, hyperelastic constitutive model in the rotation-free, large deformation, shell finite element (FE) formulation and applied it to dynamic simulations of an aortic heart valve. In a forthcoming paper (Gilmanov et al. 2018), we incorporate the rotation-free thin shell FE method for nonlinear, anisotropic, hyperplastic tissues (Gilmanov et al. 2016) in the CURVIB-FE-FSI framework (Gilmanov et al. 2015). The main goal of that paper was to provide quantitative illustrations of the significant effects that the material properties of the heart valve leaflets have on hemodynamics.

We consider only thin shell models for the solid domain. In the Kirchhoff–Love model of thin shells (Timoshenko and Woinowsky-Krieger 1959) the position vector $\boldsymbol{R}$ of any point within the volume of the shell in the reference configuration is defined in terms of the surface curvilinear coordinates and the local normal distance ζ to the middle surface, with $-h_0/2 \leq \zeta \leq h_0/2$, and h_0 being the thickness of the shell. The position of the points in the current configuration of the shell $\boldsymbol{r}$ is defined in the same way and can be mapped back to the reference configuration using the same local normal distance to the middle surface ζ. For the Kirchhoff–Love model of thin shells the components of the Green–Lagrange strain tensor in the entire volume of the shell can be expressed via the deformation of the shell's middle surface as follows (Stolarski et al. 2013):

$$E_{ij} = E_{ij}^{\mathrm{m}} + \zeta E_{ij}^{\mathrm{b}}. \tag{4.15}$$

Here E_{ij}^{m} are the membrane and E_{ij}^{b} the bending components of the strain tensor.

We adopt here the model developed by Stolarski et al. (2013), which employs triangular finite elements and approximates the shell curvature tensor without using the rotational degrees of freedom. To accomplish that the curvature of a given element is associated with nodal displacements of that element as well as with nodal displacements of the three surrounding elements, this permits definition of a complete quadratic polynomial, representing configuration of that group of four elements (called "the patch") in a moving with the shell rectilinear coordinate system. This polynomial, simply by its differentiation, permits for simple, and accurate, approximations of the element curvature tensor at any stage of the large deformation process. Most importantly, since the above approach is used only to compute the bending strains within the element and the computation of the membrane strains is based on the flat geometry of the element, the non-physical membrane locking is automatically avoided. For the derivation and the details of the method the reader is referred to Stolarski et al. (2013).

The outlined approach leads to the discretized FE version of the governing equations for the structure. By virtue of Eq. (4.15), the weak formulation of Eq. (4.6) can be written in the following form, containing the sum over all elements $e = 1, \ldots, \overline{E}$ of the triangulated domain:

$$\sum_{e=1}^{\overline{E}}\left(\iiint\limits_{\_V_0}\left(\delta\boldsymbol{u}_e^{\mathrm{T}}\left(\underline{\boldsymbol{B}}^{\mathrm{m}}+\zeta\,\underline{\boldsymbol{B}}^{\mathrm{b}}\right)^{\mathrm{T}}\underline{\boldsymbol{S}}-\delta\boldsymbol{u}_e^{\mathrm{T}}\rho_{\mathrm{s}}\boldsymbol{b}+\delta\boldsymbol{u}_e^{\mathrm{T}}\rho_{\mathrm{s}}\ddot{\boldsymbol{u}}_e\right)\mathrm{d}V_0-\iint\limits_{A_0}\delta\boldsymbol{u}_e^{\mathrm{T}}\boldsymbol{t}\,\mathrm{d}A_0\right)=0, \tag{4.16}$$

where $\boldsymbol{u}_e$, $\ddot{\boldsymbol{u}}_e$ are the displacement and acceleration within the element e, $\underline{\boldsymbol{B}}^{\mathrm{m}}$ and $\underline{\boldsymbol{B}}^{\mathrm{b}}$ are membrane and bending strain-displacement matrices, respectively. Thus, the vector of internal forces is

$$\boldsymbol{f}_e^{\mathrm{int}}=\iiint\limits_{V_{0e}}\left[\left(\underline{\boldsymbol{B}}^{\mathrm{m}}\right)^{\mathrm{T}}\underline{\boldsymbol{T}}^{\mathrm{T}}\underline{\boldsymbol{D}}^{\mathrm{loc}}\boldsymbol{T}\,\underline{\boldsymbol{E}}^{\mathrm{m}}+\zeta^2\left(\underline{\boldsymbol{B}}^{\mathrm{b}}\right)^{\mathrm{T}}\underline{\boldsymbol{T}}^{\mathrm{T}}\underline{\boldsymbol{D}}^{\mathrm{loc}}\underline{\boldsymbol{T}\,\boldsymbol{E}}^{\mathrm{b}}\right]\mathrm{d}V_0, \tag{4.17}$$

where superscript indices (m) and (b) indicate membrane and bending-related matrices in the curvilinear coordinate system on the surface (Stolarski et al. 2013), while matrix $\underline{\boldsymbol{T}}$ represents the necessary transformation of tensors to make the representation of the stresses and strains in the correctly related coordinate systems. Because of the space restrictions, the detailed formulation of all matrices we employ is not given here. Instead, we refer the Readers to the recently published paper (Stolarski et al. 2013), where all such details of the formulation are presented.

The element vector of the nodal external forces $\boldsymbol{f}_e^{\mathrm{ext}}$ and the element mass matrix $\boldsymbol{M}_e$ resulting from the presented formulation take the following form

$$\boldsymbol{f}_e^{\mathrm{ext}}=\iint\limits_{A_{0e}}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{t}\,\mathrm{d}A_0,\quad \underline{\boldsymbol{M}}_e=\iiint\limits_{V_{0e}}\rho_{\mathrm{s}}\boldsymbol{N}^{\mathrm{T}}\boldsymbol{N}\mathrm{d}V_0, \tag{4.18}$$

where $\boldsymbol{t}$ is a traction vector, and $\boldsymbol{N}$ is a vector of linear basis functions (Stolarski et al. 2013). Here, the external forces $\boldsymbol{f}_e^{\mathrm{ext}}(\boldsymbol{u},\boldsymbol{t})$ depend both on structure displacements $\boldsymbol{u}$ and the applied fluid traction $\boldsymbol{t}$. Assembly of the above vectors and matrices leads to the following final form of the structural domain equations:

$$\boldsymbol{f}^{\mathrm{int}}(\boldsymbol{u})+\underline{\boldsymbol{M}}\ddot{\boldsymbol{u}}=\boldsymbol{f}^{\mathrm{ext}}(\boldsymbol{u},\boldsymbol{t}). \tag{4.19}$$

In this chapter, three types of boundary conditions for the shell are used: free, hinged, and fixed boundary conditions. Detailed description and implementation of these boundary conditions one can find in Stolarski et al. (2013).

In the numerical integration of some dynamic, nonlinear problems with high frequency modes, a dissipative mechanism is needed in Eq. (4.19) to dump spurious oscillations and help get converged solutions (Smith and Griffith 2004). If dissipation is to be included in the system, a term related to the velocities has to be added in Eq. (4.19) as follows:

$$\boldsymbol{f}^{\mathrm{int}}(\boldsymbol{u})+\underline{\boldsymbol{M}}\ddot{\boldsymbol{u}}+\underline{\boldsymbol{D}}\dot{\boldsymbol{u}}=\boldsymbol{f}^{\mathrm{ext}}(\boldsymbol{u},\boldsymbol{t}), \tag{4.20}$$

where the matrix $\underline{\boldsymbol{D}}$ defines the dissipation term.

The detailed description of the solid solver algorithm with all matrices can be found in Stolarski et al. (2013). A short description of this algorithm is presented below.

We employ the Newmark time integration algorithm (Newmark 1959) to solve solid structure Eq. (4.20), which is formulated as follows:

$$\begin{aligned}\dot{\boldsymbol{u}}^{n+1} &= \dot{\boldsymbol{u}}^{n} + \Delta t(1-\gamma)\ddot{\boldsymbol{u}}^{n} + \Delta t\gamma\ddot{\boldsymbol{u}}^{n+1},\\ \boldsymbol{u}^{n+1} &= \boldsymbol{u}^{n} + \Delta t\boldsymbol{u}^{n} + \Delta t^{2}\left(\frac{1}{2}-\omega\right)\ddot{\boldsymbol{u}}^{n} + \Delta t^{2}\omega\ddot{\boldsymbol{u}}^{n+1},\end{aligned} \tag{4.21}$$

where Δt is the time step, subscript n, $n+1$ indicates time level $t^{n+1} = t^{n} + \Delta t$, and γ, ω are parameters that determine the stability and accuracy of the scheme. Implicit schemes are unconditionally stable for $2\omega \geq \gamma \geq 0.5$. The Newmark scheme has second-order accuracy for $\gamma = 0.5, \omega = 0.25$ (Smith and Griffith 2004). From Eq. (4.21), one gets the following formulas for the velocity and acceleration vectors

$$\begin{aligned}\ddot{\boldsymbol{u}}^{n+1} &= \frac{1}{\omega\Delta t^{2}}\left(\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}\right) - \frac{1}{\omega\Delta t}\dot{\boldsymbol{u}}^{n} - \left(\frac{1}{2\omega}-1\right)\ddot{\boldsymbol{u}}^{n},\\ \dot{\boldsymbol{u}}^{n+1} &= \frac{\gamma}{\omega\Delta t}\left(\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}\right) - \left(\frac{\gamma}{\omega}-1\right)\dot{\boldsymbol{u}}^{n} - \Delta t\left(\frac{\gamma}{2\omega}-1\right)\ddot{\boldsymbol{u}}^{n},\end{aligned} \tag{4.22}$$

which, when inserted in Eq. (4.20), yield

$$\begin{aligned}&\boldsymbol{f}^{\mathrm{int}}\left(\boldsymbol{u}^{n+1}\right) + \underline{\boldsymbol{M}}\left[\frac{1}{\omega\Delta t^{2}}\left(\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}\right) - \frac{1}{\omega\Delta t}\dot{\boldsymbol{u}}^{n} - \left(\frac{1}{2\omega}-1\right)\ddot{\boldsymbol{u}}^{n}\right]\\ &+ \underline{\boldsymbol{D}}\left[\frac{\gamma}{\omega\Delta t}\left(\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n}\right) - \left(\frac{\gamma}{\omega}-1\right)\dot{\boldsymbol{u}}^{n} - \left(\frac{\gamma}{2\omega}-1\right)\ddot{\boldsymbol{u}}^{n}\right] = \boldsymbol{f}^{\mathrm{ext}}.\end{aligned} \tag{4.23}$$

When the unknown $\boldsymbol{u}^{n+1}$ is retained in the left-hand side of the equation and the known variables are gathered in the right-hand side, the following discrete equation is obtained

$$\begin{aligned}&\boldsymbol{f}^{\mathrm{int}}\left(\boldsymbol{u}^{n+1}\right) + \frac{1}{\omega\Delta t^{2}}\underline{\boldsymbol{M}}\boldsymbol{u}^{n+1} + \frac{\gamma}{\omega\Delta t}\underline{\boldsymbol{D}}\boldsymbol{u}^{n+1}\\ &= \underline{\boldsymbol{M}}\left[\frac{1}{\omega\Delta t^{2}}\boldsymbol{u}^{n} + \frac{1}{\omega\Delta t}\dot{\boldsymbol{u}}^{n} + \left(\frac{1}{2\omega}-1\right)\ddot{\boldsymbol{u}}^{n}\right]\\ &\quad + \underline{\boldsymbol{D}}\left[\frac{\gamma}{\omega\Delta t}\boldsymbol{u}^{n} + \left(\frac{\gamma}{\omega}-1\right)\dot{\boldsymbol{u}}^{n} + \Delta t\left(\frac{\gamma}{2\omega}-1\right)\ddot{\boldsymbol{u}}^{n}\right] + \boldsymbol{f}^{\mathrm{ext}}.\end{aligned} \tag{4.24}$$

The last equation constitutes a nonlinear system of algebraic equations that has to be solved at each time step. This system is solved using the Newton linearization approach. Denoting by $\boldsymbol{u}_{i}^{n+1}$ the value of $\boldsymbol{u}^{n+1}$ at iteration i, the following equation is obtained by linearizing $\boldsymbol{f}^{\mathrm{int}}\left(\boldsymbol{u}_{i}^{n+1}\right) = \boldsymbol{f}^{\mathrm{int}}\left(\boldsymbol{u}_{i-1}^{n+1} + \Delta\boldsymbol{u}_{i}^{n+1}\right)$:

$$\boldsymbol{f}^{\mathrm{int}}\left(\boldsymbol{u}_{i}^{n+1}\right) \approx \boldsymbol{f}^{\mathrm{int}}\left(\boldsymbol{u}_{i-1}^{n+1}\right) + \underline{\boldsymbol{K}}_{i-1}^{n+1}\Delta\boldsymbol{u}_{i}^{n+1} = 0, \tag{4.25}$$

where $\boldsymbol{K}$ is the tangent stiffness matrix. The increment $\Delta \boldsymbol{u}_i^{n+1}$ between the iteration $i-1$ and i is the solution of the following system of linear algebraic equations, resulting from Eq. (4.24).

$$\left(\underline{\boldsymbol{K}} + \frac{1}{\omega \Delta t^2}\underline{\boldsymbol{M}} + \frac{\gamma}{\omega \Delta t}\underline{\boldsymbol{D}}\right)_{i-1}^{n+1} \Delta \boldsymbol{u}_i^{n+1} = \boldsymbol{R}_{i-1}^{n+1}, \tag{4.26}$$

with the residual

$$\begin{aligned}\boldsymbol{R}_{i-1}^{n+1} = &-\boldsymbol{f}^{\text{int}}\left(\boldsymbol{u}_{i-1}^{n+1}\right) - \left[\frac{1}{\omega \Delta t^2}\underline{\boldsymbol{M}} + \frac{\gamma}{\omega \Delta t}\underline{\boldsymbol{D}}\right]\boldsymbol{u}_{i-1}^{n+1} \\ &+ \underline{\boldsymbol{M}}_{i-1}^{n+1}\left[\frac{1}{\omega \Delta t^2}\boldsymbol{u}_{i-1}^{n} + \frac{1}{\omega \Delta t}\dot{\boldsymbol{u}}_{i-1}^{n} + \left(\frac{1}{2\omega} - 1\right)\ddot{\boldsymbol{u}}_{i-1}^{n}\right] \\ &+ \underline{\boldsymbol{D}}_{i-1}^{n+1}\left[\frac{\gamma}{\omega \Delta t}\boldsymbol{u}_{i-1}^{n} + \left(\frac{\gamma}{\omega} - 1\right)\dot{\boldsymbol{u}}_{i-1}^{n} + \Delta t\left(\frac{\gamma}{2\omega} - 1\right)\ddot{\boldsymbol{u}}_{i-1}^{n}\right] + \boldsymbol{f}^{\text{ext}},\end{aligned} \tag{4.27}$$

in which the forces $\boldsymbol{f}^{\text{ext}}(\boldsymbol{u}^n, t^n)$, $\boldsymbol{f}^{\text{int}}(\boldsymbol{u}^n)$ are computed according to the explicit formulas presented in the preceding sections. The matrix D presented in Eq. (4.20) can be independently defined as linear combination of mass and stiffness matrices $\underline{\boldsymbol{M}}$, $\underline{\boldsymbol{K}}$ (the so-called proportional damping)

$$\underline{\boldsymbol{D}} = f_m \underline{\boldsymbol{M}} + f_k \underline{\boldsymbol{K}}, \tag{4.28}$$

where f_m and f_k are constants and are called "Rayleigh" damping coefficients (Smith and Griffith 2004).

The solution of the above linear equations, $\Delta \boldsymbol{u}_i^{n+1}$, is used to update displacements, velocities, and accelerations as follows:

$$\begin{aligned}\boldsymbol{u}_i^{n+1} &= \boldsymbol{u}_{i-1}^{n+1} + \Delta \boldsymbol{u}_i^{n+1}, \\ \dot{\boldsymbol{u}}_i^{n+1} &= \frac{\gamma}{\omega \Delta t}\left(\boldsymbol{u}_i^{n+1} - \boldsymbol{u}_i^{n}\right) - \left(\frac{\gamma}{\omega} - 1\right)\dot{\boldsymbol{u}}_i^{n} - \Delta t\left(\frac{\gamma}{2\omega} - 1\right)\ddot{\boldsymbol{u}}_i^{n}, \\ \ddot{\boldsymbol{u}}_i^{n+1} &= \frac{1}{\omega \Delta t^2}\left(\boldsymbol{u}_i^{n+1} - \boldsymbol{u}_i^{n}\right) - \frac{1}{\omega \Delta t}\dot{\boldsymbol{u}}_i^{n} - \left(\frac{1}{2\omega} - 1\right)\ddot{\boldsymbol{u}}_i^{n}.\end{aligned} \tag{4.29}$$

The iterative process is declared converged when a specified tolerance of the iterative process is met, and the algorithm is advanced to the next time level.

The conjugate gradient (CG) method (Smith and Griffith 2004) is used to solve the linear system of Eq. (4.26). Overall, the cost of using CG is relatively low. The overall computational cost, however, depends on the number of Newton iterations to update the displacement, velocity, and acceleration of the nodal points of the structural mesh.

4.3.3 *Representation of Immersed Thin Structures in the Fluid Domain*

A key feature of our method is that it employs a FE formulation for thin shells in which the structural equations are formulated in terms of variables defined on its mid-surface (see Sect. 4.3.2). This approach is accurate and, thus, appropriate for analysis of thin structures, but it also augments the efficiency of the overall FSI methodology.

Whether or not the thickness of the structures is accounted for in the FSI analysis presents important algorithmic challenges for the CURVIB method. This is because the CURVIB method is designed to use normal vectors to the immersed surface to: (i) identify the position of background grid nodes relative to the fluid/structure interface by finding fluid nodes (IB nodes) in the immediate proximity of the interface as illustrated in Fig. 4.1; (ii) reconstruct velocity boundary conditions at immersed boundary (IB) nodes along the local normal to the boundary; and (iii) calculate the loads on the structure imparted by fluid stresses for FSI problems (see Sect. 4.3.4). Bodies with nonzero thickness, which have been handled in all previous applications of the CURVIB method, are closed surfaces (i.e., the topologically equivalent to a sphere). This implies that at every point on the surface of the body there is a unique normal vector that points toward the fluid side of the interface, namely the

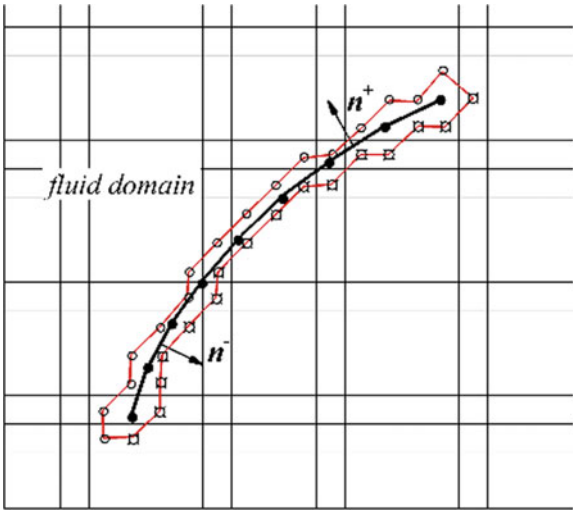


Fig. 4.1 The background grid where the governing equations for the fluid are solved along with the thin body immersed and associated IB nodes. Solid circles and solid line are vertices and elements of the solid structure, respectively. Open circles are IB nodes from positive side and open circles with small primes are IB nodes from negative side of the surface. The red line marks the interface between the fluid domain and the layer of IB nodes surrounding the thin structure. Figure reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

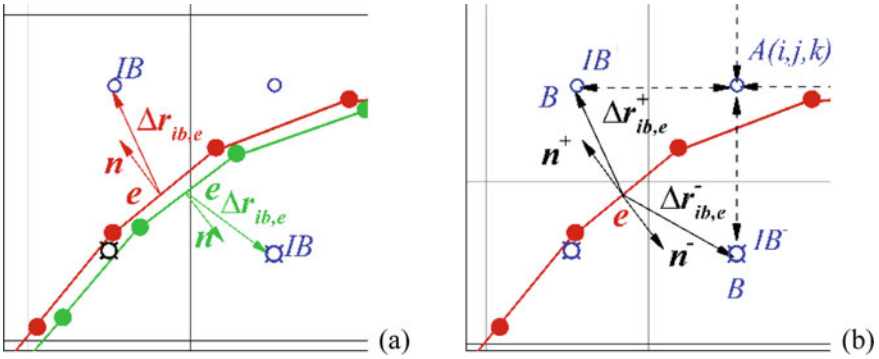


Fig. 4.2 Two different approaches for considering thin immersed body in the fluid. **a** A closed body for which only the outward normal vector points toward the fluid side of the interface. The standard CURVIB formulation has been developed for such bodies. **b** A surface with boundary (the mid-surface of the thin structure). A two-side surface for which both the positive and negative surface normal vectors point toward the fluid side of the interface. In both sketches open circles denote background grid nodes (IB marks the immersed boundary nodes where boundary conditions are reconstructed), closed circles are Lagrangian points discretizing the body, and e is the center of a structure shell element. Dashed lines indicate the direction of the searching algorithm from the background (fluid) grid node $A(i, j, k)$ to the adjacent points $B(i \pm 1, j \pm 1, k \pm 1)$. The points IB^{+} and IB^{-} in (**b**) indicate IB nodes from positive and negative side of the surface, respectively. Figures reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

positive wall normal vector (see Fig. 4.2a). On the other hand, when the structure is represented only by its mid-plane, as in the present FE thin shell model, the resulting surface is what is referred to in topological terms as a surface with boundary (i.e., the topological equivalent of a disk). For such a case, it is readily apparent from Fig. 4.2b that at each point on the surface both the positive and negative wall normal vectors point toward the fluid side of the interface. Consequently, the standard node classification and boundary condition reconstruction algorithms used in the CURVIB method (Borazjani et al. 2008) cannot be readily applied and need to be modified. In this section, we describe the algorithmic changes we have implemented to the CURVIB method to enable its coupling with the thin shell FE formulation.

At each triangular element e on the mid-surface of the thin structure, we calculate the positive $\boldsymbol{n}_e^{+}$ and negative $\boldsymbol{n}_e^{-}$ surface normal vectors to identify the positive and negative, respectively, sides of the surface. The normal vectors are calculated at the center of the element e, and the positive normal is defined as the outward normal of the triangle with clockwise nodal numbering. It is thus evident that $\boldsymbol{n}_e^{+} = -\boldsymbol{n}_e^{-}$. The triangulated surface is tracked with a set of Lagrangian points (the nodes of the triangles), which are used to define the boundary conditions (position and velocity of each Lagrangian node) for the fluid solver.

To find the IB nodes for a given configuration of the thin structure mid-surface we begin by checking the intersection between the lines connecting the centers of fluid cells in the vicinity of the body (dashed lines) and the surface (solid lines) as

shown in Fig. 4.2b. In three dimensions, the intersection is found by checking six surrounding grid lines from the local grid point $A(i, j, k)$ and the adjacent points $B(i \pm 1, j \pm 1, k \pm 1)$ (Fig. 4.2b). A fluid node is also considered as an IB node if the distance between the point and the center of an element e is less than one grid cell size Δh, i.e., $\left|\Delta \boldsymbol{r}_{ib,e}\right| < \min\ \Delta h$, where $\Delta \boldsymbol{r}_{ib,e} = (\mathbf{r}_{\text{ib}} - \mathbf{r}_e)$, $\boldsymbol{r}_{\text{ib}}$ is the position of the IB node and $\boldsymbol{r}_e$ is the position vector of the center of the triangular element (see Fig. 4.2b). The IB node is assigned to correspond to the surface element e on the solid surface.

To implement this algorithm in parallel computing, we utilize the bounding box search approach (Borazjani et al. 2008) in order to reduce the involvement of unnecessary surface triangles in the searching algorithm. We cover the entire structure with a volume of size $[x_{\min} - x_{\max}; y_{\min} - y_{\max}; z_{\min} - z_{\max}]$. This volume is further divided into $N_i \times N_j \times N_k$ smaller bounding boxes uniformly. The choice of N_i, N_j, N_k depends on the number of triangulated elements and the number of grid points per processors. In our simulations, N_i, N_j, N_k are typically chosen to be less than 50. Our line intersection strategy above is applied only for fluid points and triangulated elements that belong to the same bounding box or adjacent ones.

To handle the aforementioned difficulty arising in our thin body approach, due to the fact that both positive and negative wall normal vectors at every element point toward the fluid, we separate the *IB* nodes in two categories: (i) positive IB^+ and (ii) negative IB^- nodes. Note that this type of separation is crucial for the load calculation discussed in Sect. 4.3.4. To determine whether an *IB* node is on the positive (+) or negative (−) side of the surface, we compute the dot product of the positive local normal vector with the vector connecting the center of the triangular surface element with the closest IB node: if $\left(\boldsymbol{n}_e^+ \cdot \Delta \boldsymbol{r}_{ib,e}\right) > 0$ then the IB node is located on the positive side, otherwise it is on the negative side (see Fig. 4.2b). We also note that the relationship between an IB node and the corresponding surface element e is not unique as there could be several *IB* nodes that correspond to the same solid surface element. This is particularly true when the background fluid grid size is much smaller than the triangulated cell of the solid surface. In the subsequent section, we discuss how we handle surface elements that do not have a unique *IB* node associated with them insofar as the calculation of the forces acting on that element is concerned.

For Reynolds numbers for which the grid spacing is sufficiently fine to resolve the near wall flow, velocity boundary conditions are reconstructed at the IB nodes using linear interpolation along the local normal of the solid surface (Gilmanov et al. 2003). In our fractional step method for solving Eq. (4.1) (Ge and Sotiropoulos 2007), this relationship is implicitly incorporated into the nonlinear momentum equation and is enforced at all times. For high Reynolds number simulations a wall model is used to reconstruct boundary conditions at the IB nodes as described in Kang et al. (2014).

4.3.4 Calculation of Loads on the Interface $\Gamma^{\mathbf{fsi}}$

To enable the coupling of the fluid and structural domains in our FSI algorithm, the flow imparted loads must be calculated on the solid surface in order to properly define the problem for the structural solver.

A major challenge in load calculation is the need to efficiently calculate the traction vector $\boldsymbol{t}_{\mathrm{f}}$ on the fluid–solid interface in parallel environment since the solid body can span across partitioned computational domains, which are assigned to different processors. It is thus necessary to develop a scalable algorithm to collectively compute the local loading at each processor and assemble all the information to have a complete loading distribution $\boldsymbol{t}_{\mathrm{f}}$ on the interface Γ^{fsi}. Here, we utilize the layer of the *IB* nodes discussed in Sect. 4.3.3 above and illustrated in Fig. 4.1. We calculate separately the pressure and viscous forces on this layer of IB nodes for each processor and the final loading condition on the *IB* nodes is assembled from all portions of all processors.

Since the fractional step method we employ only requires velocity boundary conditions at the *IB* nodes (Ge and Sotiropoulos 2007), the pressure p at these nodes is not available. For that we calculate pressure at the IB nodes using interpolation along the normal direction in the similar fashion for the velocity components as reported in Gilmanov and Sotiropoulos (2005).

We note that we fully retain the sharp interface nature of our method in the calculation of the traction vectors even though the thin body is represented by its mid-surface. This is accomplished by using the previously discussed positive and negative wall normal vectors to independently calculate and store the forces acting on the (+) and (−) sides of each element on the interface using one-sided interpolation directed from the element toward the respective (+) or (−) side of the fluid nodes. Consequently, the so-calculated + and − traction vectors at each surface element exhibit a discontinuity across the thin body, which is an important physical feature of the problem preserved by this approach. The shear stress tensor components $\tau_{f,ij}$ are evaluated locally at every fluid node using the second-order differencing to compute the velocity gradients:

$$\tau_{f,ij} = \mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right). \tag{4.30}$$

Depending on the grid resolution, the components of the shear stress tensor $\boldsymbol{\tau}_{\mathrm{f}}$ are interpolated along the normal direction $\boldsymbol{n}^{\pm}$ in similar fashion as the pressure (Gilmanov and Sotiropoulos 2005) or reconstructed using a wall model (Kang et al. 2012) to obtain values at the $\mathrm{IB}^{\pm}$ nodes $\boldsymbol{\tau}_{\mathrm{ib}}^{\pm}$. Finally, the fluid stress tensor $\boldsymbol{\sigma}_{\mathrm{ib}}^{\pm}$ at the $\mathrm{IB}^{\pm}$ nodes is evaluated as follows:

$$\underline{\boldsymbol{\sigma}}_{\mathrm{ib}}^{\pm} = -p_{\mathrm{ib}}^{\pm}\underline{\boldsymbol{I}} + \underline{\boldsymbol{\tau}}_{\mathrm{ib}}^{\pm}, \tag{4.31}$$

where $\underline{\boldsymbol{I}}$ is the unit tensor. After the fluid stress tensor $\underline{\boldsymbol{\sigma}}_{\mathrm{ib}}^{\pm}$ has been obtained, a one-sided projection procedure, from the corresponding $+$ or $-$ *IB* nodes to the actual solid surface is required to find the stress tensor on Γ^{fsi}. This procedure is described as follows. We already mentioned above in the Sect. 4.3.3 that the relationship between *IB* nodes and a surface element e is not unique as there could be several *IB* nodes (say $\overline{N}_e^{\pm}$ such nodes exist) that are associated with the same surface element e. Note that only either positive or negative *IB* nodes are involved in the load calculation process of positive or negative stresses, correspondingly. Therefore, for such cases and in order to calculate the fluid stress tensor $\underline{\boldsymbol{\sigma}}_e^{\pm}$ on the surface of the body Γ^{fsi} an interpolation procedure is implemented from the surrounding $\mathrm{IB}^{\pm}$ nodes to the solid surface element as follows:

$$\underline{\boldsymbol{\sigma}}_e^{\pm} = \sum_{\mathrm{ib}=1}^{\overline{N}_e^{\pm}} \underline{\boldsymbol{\sigma}}_{\mathrm{ib}}^{\pm} / \left|\Delta \boldsymbol{r}_{ib,e}^{\pm}\right| / \sum_{\mathrm{ib}=1}^{\overline{N}_e^{\pm}} 1 / \left|\Delta \boldsymbol{r}_{\mathrm{ib},e}^{\pm}\right|. \tag{4.32}$$

Finally, the net loading at each triangular element on the thin structure mid-surface is defined as the sum of loads from both sides of the middle surface $\boldsymbol{t}_e = \boldsymbol{t}_e^{+} + \boldsymbol{t}_e^{-}$, where $\boldsymbol{t}_e^{\pm} = \underline{\boldsymbol{\sigma}}_e^{\pm} \cdot \boldsymbol{n}_e^{\pm}$ (see Fig. 4.3). Note that the total traction vector of the load $\boldsymbol{t}_e$ is calculated using all (positive and negative) IB nodes associated with the center of the triangular element e. In our FE solver, however, the traction vector is required at the vertices of the triangular elements $\boldsymbol{t}_v$. Thus, an interpolation procedure is implemented to transfer the traction vector from the elements to the nodes using

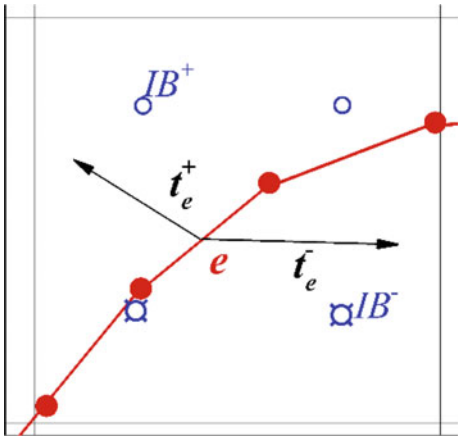


Fig. 4.3 The thin structure in our formulation is treated as a sharp interface. The schematic illustrates that at each element e on the surface, two traction vectors are computed from the positive $\boldsymbol{t}_e^{+}$ and negative $\boldsymbol{t}_e^{-}$ sides of the interface. The total traction vector $\boldsymbol{t}_e = \boldsymbol{t}_e^{+} + \boldsymbol{t}_e^{-}$ is used to compute the local load imparted by the flow on the structure on each surface element. Figure reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

distance weighted average: $\boldsymbol{t}_v = \boldsymbol{I}(\boldsymbol{t}_e) = \sum_{e=1}^{\bar{e}} \left(\boldsymbol{t}_e / \left|\Delta \boldsymbol{r}_{v,e}\right|\right) / \sum_{e=1}^{\bar{e}} \left(1 / \left|\Delta \boldsymbol{r}_{v,e}\right|\right)$, where the summation is implemented over all elements e adjacent to the vertex v and $\left|\Delta \boldsymbol{r}_{v,e}\right|$ is a distance from the vertex v to the center of the element e.

4.3.5 The Algorithm for Coupling the Fluid $\mathcal{F}$ and Solid $\mathfrak{H}$ Solvers

The governing equations of the fluid (Eq. 4.1) and solid (Eq. 4.6) domains as well as the continuity conditions on the interface constitute a modularly partitioned fluid–structure interaction problem, which can be solved by coupling together two independent solvers: the fluid solver $\mathcal{F}$ and the solid solver $\mathfrak{H}$. We use the conventional Dirichlet–Neumann partition (Felippa et al. 2001) to couple the system of fluid–solid equations. This means that the fluid equations are solved by enforcing Dirichlet boundary condition, while the solid equations are solved by prescribing the load on the interface Γ^{fsi} (Fernandez et al. 2007). The need to enforce the continuity of the velocity and the normal stresses on the interface Γ^{fsi} requires that both the displacement and velocity of the solid body $\boldsymbol{u}$ and $\dot{\boldsymbol{u}}$ must be tracked. We define $\boldsymbol{Q}$ as the solution of solid solver $\mathfrak{H}$ (Eq. 4.9):

$$\boldsymbol{Q} = (\boldsymbol{u}, \dot{\boldsymbol{u}}) \tag{4.33}$$

From Eq. (4.4) and (4.9), the FSI coupling can be formulated as a fixed-point operator for $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \mathfrak{H} \circ \mathcal{F}(\boldsymbol{Q}) \tag{4.34}$$

To facilitate the description of the FSI algorithm, let us assume, without loss of generality, that the pressure and velocity fields $\boldsymbol{v}^n$, p^n for the fluid alongside with the displacements and velocities of the solid structure $\boldsymbol{u}^n$, $\dot{\boldsymbol{u}}^n$ are known at time step n. The fluid and structural equations, Eqs. (4.35–4.36), are solved to obtain the structural displacement and velocity as well as the fluid pressure and velocity fields at time step t^{n+1} with the current boundary conditions on $\Gamma_{\text{f}}^N \cup \Gamma_{\text{f}}^D \cup \Gamma^{\text{fsi}}$ via a series of subiterations (l) to satisfy Eq. (4.34). We seek the solution of the discrete fluid operator $\mathcal{F}$ at time step t^{n+1} as:

$$\mathcal{F}\left(\boldsymbol{v}^{n+1}, p^{n+1}, \boldsymbol{u}^{n+1}, \dot{\boldsymbol{u}}^{n+1}\right) = 0, \ \text{in}\, \Omega_{\text{f}}, \tag{4.35}$$

and the solution of the discrete solid operator $\mathfrak{H}$ as follows:

$$\mathfrak{H}\left(\boldsymbol{u}^{n+1}, \dot{\boldsymbol{u}}^{n+1}, \boldsymbol{t}^{n+1}\right) = 0, \ \text{in}\, \Omega_{\text{s}}, \tag{4.36}$$

where $\boldsymbol{t}^{n+1} = \boldsymbol{\mathcal{L}}\left(\boldsymbol{v}^{n+1}, p^{n+1}\right)$ is the traction vector imparted by the fluid on the body surface Γ^{fsi}. The function $\boldsymbol{\mathcal{L}}\left(\boldsymbol{v}^{n+1}, p^{n+1}\right)$ represents the loading on the structure surface from the pressure and velocity fields in fluid. The approach we employ to calculate $\boldsymbol{\mathcal{L}}$ in the discrete space is described in Sect. 4.3.4, Eqs. (4.30–4.32). The fixed-point subiteration procedure to find $\boldsymbol{Q}$ at time steps $n + 1$ can thus be written as follows:

$$\boldsymbol{Q}_{l+1}^{n+1} = \boldsymbol{\mathfrak{H}}^{n+1} \circ \boldsymbol{\mathcal{F}}^{n+1}\left(\boldsymbol{Q}_{l}^{n+1}\right), \tag{4.37}$$

where $l + 1$ is the new iterate of $\boldsymbol{Q}^{n+1}$. In our fixed-point iteration, the fluid solver $\boldsymbol{\mathcal{F}}$ uses displacements and velocity of the solid structure $\boldsymbol{u}_{l}^{n+1}, \dot{\boldsymbol{u}}_{l}^{n+1}$ and gives new fluid velocity $\boldsymbol{v}_{l+1}^{n+1}$ and pressure field p_{l+1}^{n+1} by solving Eq. (4.35). The solid solver $\boldsymbol{\mathfrak{H}}$ in turn uses the so updated fluid velocities, and pressure field to advance the solution of displacements and velocity of the solid structure $\boldsymbol{u}_{l+1}^{n+1}, \dot{\boldsymbol{u}}_{l+1}^{n+1}$. Subiterations ($l$) are implemented every time step to satisfy the coupled system of equations and advance the solution to time step $n + 1$:

$$\begin{aligned}
&\left(\boldsymbol{v}_{l+1}^{n+1}, p_{l+1}^{n+1}\right) = \boldsymbol{\mathcal{F}}\left(\boldsymbol{v}_{l}^{n+1}, p_{l}^{n+1}, \boldsymbol{u}_{l}^{n+1}, \dot{\boldsymbol{u}}_{l}^{n+1}\right), \\
&\left(\boldsymbol{u}_{i+1}^{n+1}, \dot{\boldsymbol{u}}_{i+1}^{n+1}\right)_{l+1} = \boldsymbol{\mathfrak{H}}\left(\left(\boldsymbol{u}_{i}^{n+1}, \dot{\boldsymbol{u}}_{i}^{n+1}\right)_{l}, \boldsymbol{t}_{l}^{n+1}\right), \quad l = 0, 1, 2, \ldots; i = 1, 2, 3 \ldots
\end{aligned} \tag{4.38}$$

where index l is the number of fixed-point iteration and all variables at level $l = 0$ are at the previous time step n, index i is the number of Newton iteration for structural equations. The subiterations continue until an appropriate norm of the error of the flow and structural variables between levels $l + 1$ and l has been reduced to a desired tolerance and the above equations have been satisfied at level $n + 1$. The above procedure is generally described as a strongly coupled FSI algorithm and ensures that the continuity of the stress at the fluid–structure interface is satisfied within the desired convergence threshold. If we just apply the above algorithm for one subiteration ($l = 0$), the requirement for the continuity of the stress is enforced only within an error that depends on the accuracy of the temporal discretization scheme. Such an algorithm is generally far more efficient than the strongly coupled approach and is referred to as loosely coupled iteration. Generally, loosely coupled FSI schemes are robust for problems involving large mass ratio (structural density considerably larger than the fluid density) while strongly coupled iterations are required to enhance robustness for problems with mass ratios of order one or lower (Sotiropoulos and Yang 2014; Baek and Karniadakis 2012).

For mass ratio problems of order one ($\rho_{\text{f}}/\rho_{\text{s}} \approx 1$), which arise in simulations of heart valves, the Aitken nonlinear relaxation technique is also implemented to accelerate the convergence of the strongly coupled FSI algorithm (Borazjani et al. 2008; Küttler and Wall 2008). The convergence tolerance for the structural and strongly coupled FSI solvers is of order 10^{-8} in terms of the L_{∞} norm.

We note that for all the cases we simulate in this work, although the underlying FSI dynamics is complex, the degrees of freedom (DOF) for the structural mesh

are quite low (in the order of thousands) compared to the fluid DOFs (in the order of millions) since the structures are relatively small and simple. Thus, the cost for solving the solid equations is small in comparison with the fluid equations. For that the solid solver is implemented as a serial code possessed by the root processor. All processors involved in the solution of the fluid equations receive the same image $\boldsymbol{Q}$ of the structure from the root processor.

Our code is parallelized and uses the Petsc Library. The simulations we report herein have been carried out on a cluster with dual 8-core AMD 6112. To estimate the efficiency of the code, we report the CPU time per node of the computational grid, per processor, and per time step. For the inverted flag problem we report in Sect. 4.4.2 below this quantity is equal to $t_{\text{CPU}}/(\text{Nodes} \cdot \text{Procs} \cdot n \text{ time}) \approx 3 \times 10^{-2}\,\mu\text{s}$.

4.4 Application to Complex FSI Problems

In this section, we demonstrate the predictive capabilities of the proposed CURVIB-FE-FSI algorithm by applying it to simulate two quite challenging both involving FSI with thin flexible structures. The first is the large amplitude vibrations of an inverted flexible flag, which has been studied experimentally by Kim et al. (2013). In the second example, we demonstrate the ability of the CURVIB-FE-FSI algorithm to simulate pulsatile, physiologic flow through a tri-leaflet aortic valve placed in an anatomic aorta. This second problem is more challenging because it is geometrically more complex, is characterized by low mass ratio ($\rho_s/\rho_f \sim 1$) and imposes a more stringent overall test for the stability and robustness of the FSI solver. Note that both of these problems were first presented in Gilmanov et al. (2015).

4.4.1 Oscillations of a Flapping Inverted Flag

The computational challenges in this problem are related to the large amplitude oscillations of the flag as well as to high Reynolds number of the flow. This problem was also investigated in recently published laboratory experiment (Kim et al. 2013). The problem is referred to as the inverted flag because the flag, a thin flexible sheet of length L, is mounted on its trailing edge with its leading edge free to move in response to a uniform incoming flow u_∞ (see Fig. 4.4a). Kim et al. (2013) carried out a series of experiments by varying u_∞ and/or the structural properties of the flag and identified a dynamically rich phase space of flag responses. They showed that the non-dimensional parameter that governs the dynamics of the FSI problem is the nondimensional bending stiffness $\beta = B/\rho_f u_\infty^2 L^3$, where B is a flexural rigidity $B = Y h_0^3/12\left(1-\nu^2\right)$ of the flag, ρ_f is the fluid density, Y is the Young's modulus, ν is the Poisson's ratio, and h_0 is the thickness of the plate. Kim et al. (2013) identified three regimes of flag response as a function of β: (1) the straight mode, where the flag is too rigid to be deflected by the flow and remains straight (large values of β);

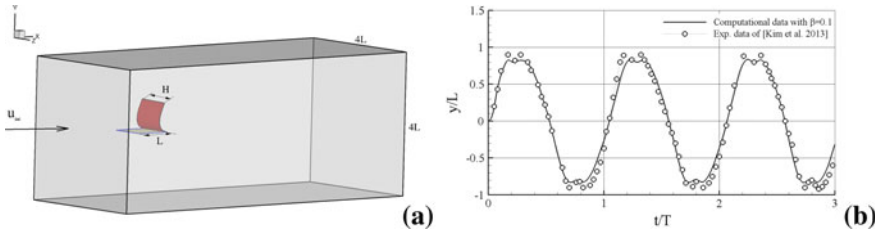


Fig. 4.4 **a** Computational domain used to simulate the inverted flag problem. The side $x_{\min}$ and all four sides $y_{\min}, y_{\max}, z_{\min}, z_{\max}$ are Dirichlet boundaries with inflow $u = u_\infty$ and no-slip boundary conditions $\boldsymbol{v} = 0$, respectively. On the boundary $x_{\max}$ Neumann condition $\partial \boldsymbol{v}/\partial n = 0$ is implemented; **b** Comparison of calculated (solid line) and measured (Khosronejad and Sotiropoulos 2014) time histories of the flag leading edge displacements for flapping mode with $\beta = 0.1$. Open circles are experimental data. Figures reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

(2) the flapping mode, where the flag undergoes large amplitude flapping oscillations (intermediate values of β); and (3) the deflected mode, where the flag is so flexible that it is deflected by the flow toward one side and remains fixed at this position at all times (small β values). Here, we report simulations for $\beta = 0.1$, which is in the flapping regime. This regime is quite challenging from the FSI simulation standpoint as it involves very large amplitude oscillations. The specific β value is selected because for this value the experiment of Kim et al. (2013) revealed a complex dynamic response of the flag characterized by rich flapping dynamics including several local minima and maxima of the flag leading edge position during the cycle of flapping motion. We carry out simulations for the following values of the various governing parameters for this problem: $Y = 2.38 \times 10^9$ Pa, $\nu = 0.38$, $\rho_s = 1.2 \times 10^3$ kg/m^3, $h_0 = 8 \times 10^{-4}$ m, $H = L = 0.3$ m, $u_\infty = 6.7$ m/s, $\mu = 1.92 \times 10^{-5}$ Pa s, $\rho_f = 0.98$ kg/m^3, hence $B = 0.118$ N m and $\beta = 0.1$. The corresponding Reynolds number, based on the inflow velocity and flag length, is $Re = u_\infty \rho_f L/\mu = 99{,}505$, and, therefore, the massively separated flow in the wake of the flapping flag is expected to be turbulent. For that we employ the CURVIB-FE-FSI method in LES mode with three-point central differencing for the convective terms in the flow equations, the dynamic Smagorinsky subgrid-scale model (Germano et al. 1991) for closure, and the wall model of (Wang and Moin 2002) to reconstruct boundary conditions on the flag as adapted for the CURVIB method by Calderer et al. (2014). The plate surface is discretized with 206 triangle elements and the background fluid grid is discretized with a uniform Cartesian mesh with $561 \times 201 \times 201$ in the stream wise (x), vertical (y), and transverse (z) directions, respectively. The non-dimensional time step is equal to $\widetilde{\Delta t} = 0.01$.

Figure 4.4b compares the measured (Kim et al. 2013) and computed time histories of the flag leading edge deflection. It is seen that the computed results are in excellent agreement with the experimental measurements. The simulations not only capture the amplitude and period of oscillations with good accuracy but also resolve the two local deflection maxima (minima) that occur in the vicinity of maximum ($y_{\min}$ or $y_{\max}$) flag

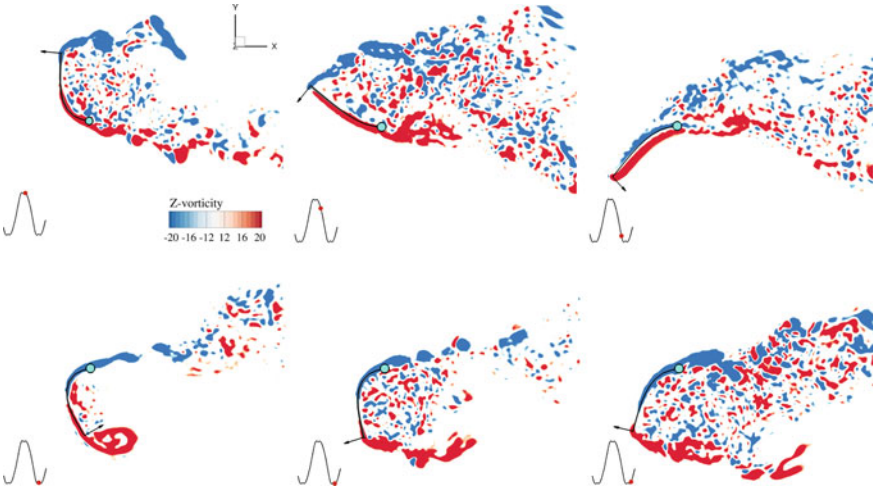


Fig. 4.5 Snapshots of the simulated inverted flag flow fields during a half period of oscillation. Contours are the out-of-plane vorticity component (z-vorticity) are plotted at various instants in time. The corresponding flag shape is also shown and the corresponding time instant is marked with a red dot in the inset. Light blue circle indicates the fixed trailing edge and the arrow indicates direction of moving leading edge. Figures reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

deflection. A more quantitative comparison with the measurements reveals that the maximum discrepancy between experiments and simulations, which occurs around maximum and minimum tip deflection, does not exceed 7% of the measured values.

Figure 4.5 depicts the calculated instantaneous out of plane vorticity field at various instants during the flapping cycle. These snapshots as well as video animations of the vorticity field (not shown herein) clearly show that the flapping dynamics is correlated with the formation of a large leading edge vortex as the flag tip approaches maximum deflections. The vortex begins to form as the flag moves upward as a result of shear-layer roll-up and leads to massive separation and shedding of vorticity in the wake at maximum deflections. The resulting wake is very complex and exhibits a large-scale meandering motion as a result of the continuous flapping motion of the flag. The computed results shown in Fig. 4.5 are in good overall qualitative agreement with the flow visualizations reported by Kim et al. (2013) and their overall description of the underlying wake dynamics as obtained in their experiments.

To elucidate the three-dimensional structure of this highly unsteady and massively separated wake, we plot in Fig. 4.6 several snapshots of the Q-criterion (Hunt et al. 1988). It is evident from this figure that the flow is dominated by shear-layer roll-up off the sharp edges of the flag, which leads to the formation of an arch vortex along the leading edge and intertwined spiral vortex tubes shed off the two sides of the flag. These structures separate from the flag and break up into small-scale turbulence in the wake.

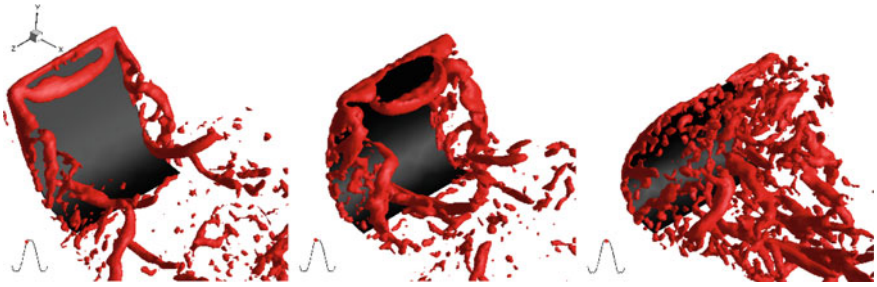


Fig. 4.6 Snapshots of Q-criterion iso-surfaces (Hunt et al. 1988) at three time instants showing the inverted flag near the maximum (y_{max}) deflection. These snapshots elucidate the 3D coherent structures in the wake of the flapping flag. The red dot in the inset of each figure identifies the corresponding instant. Figures reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

To our knowledge the results reviewed herein Gilmanov et al. (2015) elucidated for the first time the three-dimensional structure of the wake of a flapping inverted flag and clearly illustrated the ability of our CURVIB-FE-FSI method to solve a very complex, high Reynolds number problem involving complex large amplitude vibrations of a thin structure. Even though not shown herein, we have carried out simulations for values of β in all three experimentally identified flag response regions and our results are in very good agreement with the experiments of Kim et al. (2013).

4.4.2 FSI Simulation of Tri-leaflet Aortic Valve

In this section, we demonstrate the ability of the method to simulate physiologic flow through a tri-leaflet aortic valve located in an anatomically realistic aorta. The flow through the aorta is driven by a prescribed physiologic flow wave form at the aorta inlet, the response of the valve leaflets and associated flow field are simulated by the new CURVIB-FE-FSI algorithm.

We consider a tri-leaflet aortic heart valve and model it as a thin shell using the rotational free FE formulation of Stolarski et al. (2013) as described in Sect. 4.3.2 above. We use in these simulations a model suitable for a prosthetic polymeric aortic valve with isotropic material and the Neo–Hookean constitutive equation. The geometric and material characteristics of the valve are specified from values available in the literature to correspond to a prosthetic polymeric valve (Carmody et al. 2006) and are as follows: valve diameter $d_0 = 0.0254$ m, leaflet thickness $h_0 = 6.0 \times 10^{-4}$ m, Young modulus $Y = 1$ MPa, Poisson coefficient $\nu = 0.35$, and density $\rho_s = 1.2 \times 10^3$ kg/m^3. As shown in Fig. 4.7a the valve is placed in an anatomic aorta, which has been reconstructed from patient-specific MRI data.

The pulsatile flow wave form we prescribe as inflow boundary condition at the inlet of the aorta domain is shown in Fig. 4.7a. The corresponding heart beat is

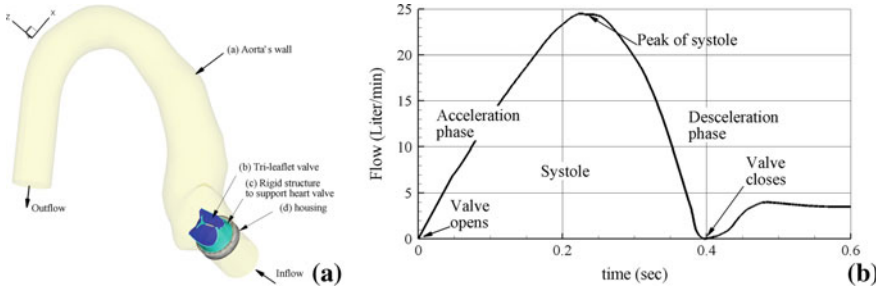


Fig. 4.7 **a** Computational domain for the FSI simulations of a tri-leaflet heart valve in an anatomic aorta. At inflow pulsatile physiological flow shown in (**a**) is simulated; at outflow Neumann boundary condition $\partial \boldsymbol{v}/\partial n = 0$ is implemented; on the aorta wall no-slip boundary condition is implemented; **b** physiological incoming flow wave form specified at the inlet of the aorta. Figures reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

equal to 70 bpm, which gives a period of the cardiovascular cycle $T = 0.857$ s. The valve diameter d_0 is used as the characteristic length scale and the peak systolic velocity of $U_0 = 0.8$ m/s is used as the velocity scale. Using the viscosity of blood $\mu = 3.52 \times 10^{-3}$ Pa s, and blood density $\rho_f = 1050$ kg/m^3, gives a peak systolic Reynolds number Re $= 6000$, which well within the physiologic range (Carmody et al. 2006). The characteristic time scale is equal to $T_0 = d_0/U_0 = 3.1 \times 10^{-2}$ s and thus the non-dimensional period of cardiac cycle is $\widetilde{T} = T/T_0 = 0.857/3.1 \times 10^{-2} =$ 27.6 non-dimensional time units. The non-dimensional time step for the simulations is set equal to $\tilde{t} = 0.01$, which corresponds to discretizing the cardiac cycle with $N_{\text{time}} = \widetilde{T}/\tilde{t} = 2760$ computational time steps. Since the density ratio for this problem is of order one, the strong coupling FSI iteration is required for stable and robust simulations. In all subsequently presented simulations 4–10 strong coupling iterations are sufficient to reduce the residuals by 8 orders of magnitude.

The overall computational setup is shown in Fig. 4.7a and consists of (a) the anatomic aorta, (b) the flexible tri-leaflet prosthetic heart valve, (c) the rigid valve support structure, and (d) housing. A curvilinear boundary-fitted grid is used to discretize aorta domain with $101 \times 101 \times 601$, in the two transverse and stream wise directions, respectively. The valve leaflets are discretized with 476 triangle elements.

The flow wave form shown in Fig. 4.7b, which corresponds to the systolic phase of the cardiac cycle during which the aortic valve opens and closes, is used to specify time-dependent Dirichlet conditions for the velocity at the inlet. At the outlet of the aorta zero-gradient Neumann condition $\partial \boldsymbol{v}/\partial n = 0$ is applied for all three velocity components along with a correction of the so-resulting velocity field to enforce global mass conservation. No-slip and no-flux boundary conditions are applied on all solid surfaces.

We note that in our numerical method the discrete continuity equation is satisfied to machine zero at each time step, thus preserving the incompressible nature of the flow locally and globally. This is accomplished by solving the Poisson equation in

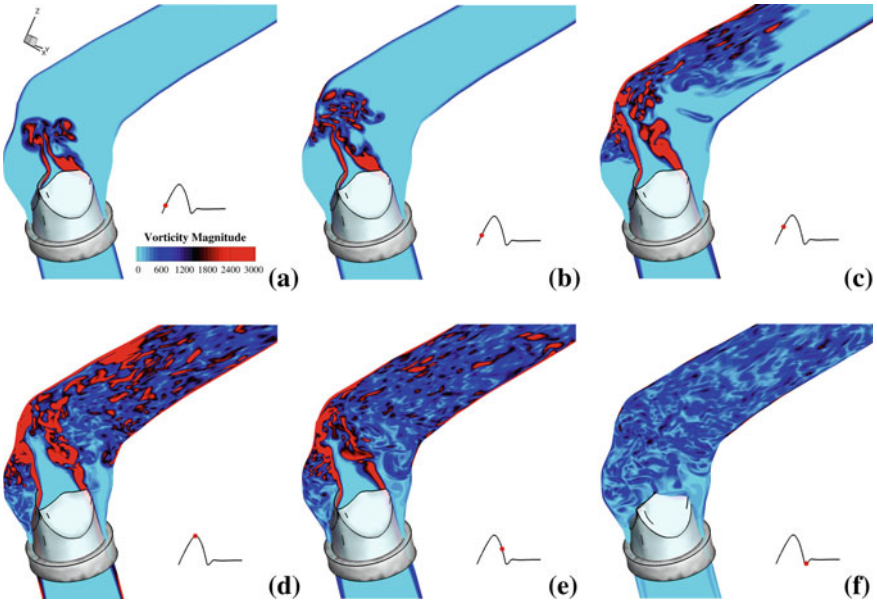


Fig. 4.8 Instantaneous contours of vorticity magnitude on a plane through the aorta plane of symmetry during systolic phase showing the opening and closing process of the aortic heart valve. The red dot in the inset of each figure identifies the corresponding instant during the cardiac cycle. Figures reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

the projection step of the fractional step method with the residual reaching machine zero at every physical time step. For more details, we refer the reader to Kang et al. (2011).

The calculated flow fields for one simulated systolic cardiac cycle are shown in Fig. 4.8. Contours of instantaneous vorticity magnitude are plotted in this figure on a plane passing through the center of the aorta. As seen in this figure, a well-defined vortex ring forms as soon as the valve opens at early systole (Fig. 4.8a). Shear layers connecting the aortic valve vortex ring with the valve leaflets are also evident in Fig. 4.8a. As the valve leaflets continue to open, the vortex ring advances and impinges on the curved aorta wall and breaks up. The valve leaflet shear layers intensify as the flow rate through the valve increases and the flow in the wake of the valve leaflets is seen to break up into small-scale turbulence at approximately halfway within the accelerating phase of the cardiac cycle (Fig. 4.8c). By the time the peak systolic flow is reached and the valve has opened fully, the flow in the aorta is seen to have transitioned to a fully turbulent state downstream of the valve leaflets (Fig. 4.8d). This state persists even after the valve closes and the flow structures in the aorta gradually decay (Fig. 4.8f).

The results shown in Fig. 4.8 reveal significant differences between the simulated flow patterns reported in Le and Sotiropoulos (2013) for a mechanical bi-leaflet

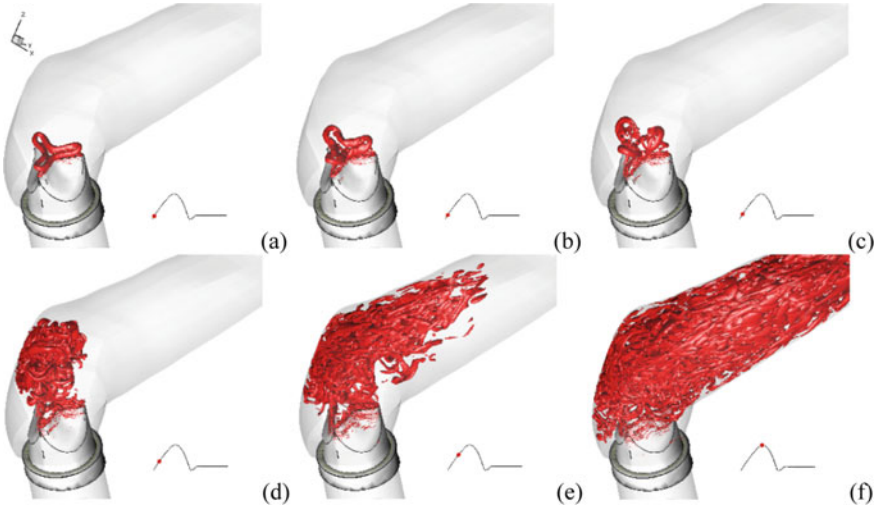


Fig. 4.9 Instantaneous iso-surfaces of the Q-criterion (Hunt et al. 1988) at various instant in time during valve opening. The red dot in the inset of each figure identifies the corresponding instant during the cardiac cycle. Figures reprinted with permission from Gilmanov et al., Journal of Computational Physics, 300, 814–843 (2015). Copyright 2015, American Institute of Physics

heart valve (MBHV) in the same anatomic aorta. More specifically, when a MBHV is implanted in the aortic position the turbulent state downstream of the valve leaflets does not emerge until shortly after peak systole. For the tri-leaflet valve, however, Fig. 4.8 clearly shows that the flow transitions to turbulence well before peak systole is reached. This finding should be attributed to the complex vortex dynamics induced by the shape of the tri-leaflet valve orifice as it opens and the interaction of the aortic valve vortex ring with the aorta wall.

To illustrate the three-dimensional dynamics of coherent structures as the valve opens, we plot in Fig. 4.9 instantaneous snapshots of the Q iso-surface (Hunt et al. 1988). As seen in Fig. 4.9a, as the valve opens the shear layer from the valve leaflets rolls up to form a three-lobed vortex ring that follows the shape of the valve orifice. As the valve continues to open, this vortex ring becomes distorted as each one of its three lobes, forming at the valve commissures, bends forward and propagates at faster speed than the rest of the ring. This complex deformation of the aortic valve ring is clearly evident in Fig. 4.9c where three distinct vortex loops are seen to have formed. Each loop forms because of the faster propagation and associated stretching of the corresponding lobe of the initial ring. Essentially the vortex interactions and instabilities revealed by our simulations are similar to those observed in pulsatile flow through corrugated nozzles (New and Tsovolos 2012). These instabilities along with the subsequent impingement of the three-lobed aortic valve ring on the aorta wall are ultimately responsible for the relatively early transition to turbulence of the flow in the wake of a tri-leaflet valve.

As mentioned earlier, in all of the above simulations, a linear, isotropic Saint-Venant (StV) material model was used, which has been also employed in the past, e.g., in Tepole et al. (2015) and may be applicable only to a certain type of prosthetic valves. Recently, the May-Newman and Yin (MNY) model (May-Newman and Yin 1998) and its applicability to heart valve has been discussed in Gilmanov et al. (2016) to simulate natural heart valves. However, the analysis presented in that paper did not involve fluid flow and, consequently, no FSI algorithm was used. Instead, the dynamic behavior of leaflets was investigated by prescribing time-varying pressure loading taken from experiments. For the set of parameters used, we found in Gilmanov et al. (2016) that heart valve with the StV material model is more obstructive to the blood flow in comparison with the heart valve with the MNY material model. The complete FSI simulations (Gilmanov et al. 2018) led to the same conclusion that the StV heart valve is more obstructive to the blood flow and creates more complex blood flow patterns. To qualitatively compare the opening kinematics of StV and MNY heart valves and the associated differences in hemodynamic patterns, in Fig. 4.10 the results of FSI simulations with StV and MNW heart valves are shown. The instantaneous vorticity contours for the two considered cases (StV and MNY) are shown in the first and second columns of Fig. 4.10. To illustrate the three-dimensional dynamics of the coherent structures as the valve opens, we plot in Fig. 4.10 (third and fourth columns) the instantaneous Q iso-surfaces (Hunt et al. 1988). Figure 4.10a shows that for the time interval shown, the StV heart valve opens only partially and obstructs the blood flow, causing significant vorticity generation downstream of the valve leaflets. At exactly the same time, the MNY heart valve is fully open and vortices are shed from the fully formed valve orifice (Fig. 4.10a–c). One can see that with the StV material, the jet spreads into the aorta faster than that for the MNY valve. Starting at $t \approx 0.12$ s, the large-scale coherent structures arising in the StV valve material disintegrate into small turbulent structures (Fig. 4.10a). For the MNY material on the other hand, the vortices remain coherent, which indicates that the MNY valve flow remains laminar for a longer period of the cardiac cycle than the StV valve flow. In fact, only starting at approximately $t \approx 0.192$ s (Fig. 4.10c), the large-scale vortex structure arising in the MNY valve disintegrates as it begins to interact with the aortic wall.

Helical flow patterns have been observed in the aortic arch, which are clearly seen from the movies (not shown here) of Q-structures spreading into the aorta. As mentioned earlier, these flow patterns are dependent on the kinematics of the valve which, in the coupled FSI analysis, is dependent on the properties of the leaflet material. We have shown that as the StV valve opens, the shear layer induced by the valve leaflets rolls up to form a three-lobed vortex ring (Fig. 4.9), which corresponds to the shape of the valve orifice. For the MNY valve, however, the opening process is faster and the resistance to the blood flow is reduced, which leads to a toroidal shape of the vortex. As the valve continues to open (Fig. 4.10b and c), the vortex ring for the StV valve becomes distorted but for the MNY valve remains toroidal and coherent. As discussed above, for the MNY valve, the coherent vortex ring begins to get disorganized only later due to its interaction with aortic wall (Fig. 4.10c). It is clearly seen that this large coherent structure produced by the MNY valve propagates into the aorta along a helical path (Fig. 4.10).

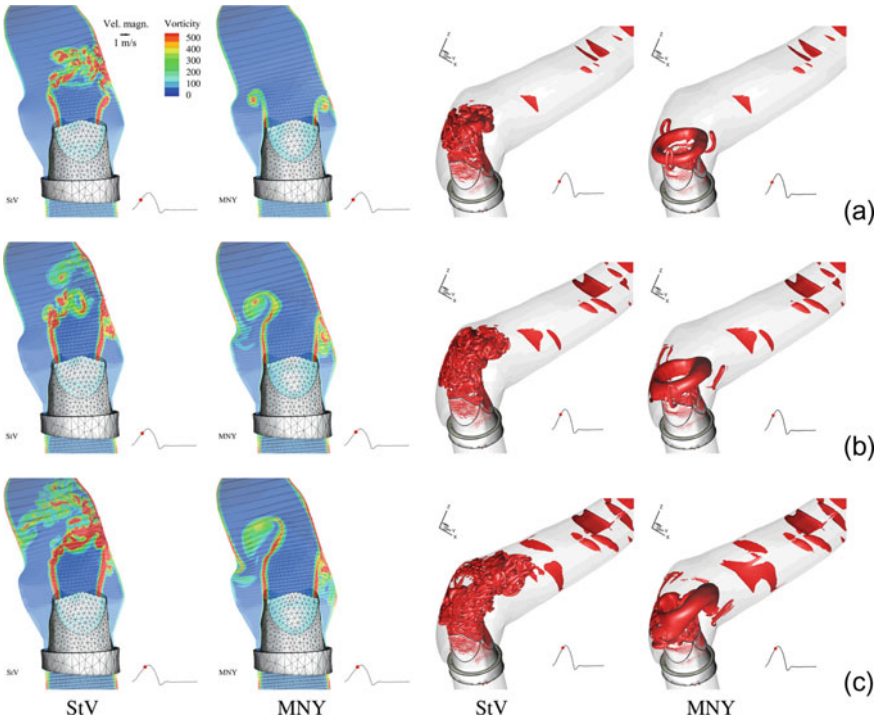


Fig. 4.10 Comparison of instantaneous contours of vorticity on a plane through the aorta during systolic phase showing the opening process of the StV aortic valve (first column) and MNY aortic valve (second column). The third and fourth columns are the instantaneous iso-surfaces of the Q-criterion (Hunt et al. 1988) for StV and MNY models, respectively. The dot in the inset of each figure identifies the corresponding instant during the cardiac cycle: **a** $t_a = 0.128$ s, **b** $t_b = 0.16$ s, and **c** $t_c = 0.192$ s. Figures reprinted with permission from Gilmanov et al., Journal of Biomechanical Engineering, 140 (2018). Copyright 2018, American Institute of Physics

To our knowledge the results we have presented herein [originally reported in Gilmanov et al. (2018)] represent the first FSI simulation of a tri-leaflet heart valve whose material is nonlinear and anisotropic and which is interacting with an anatomic aorta at physiologic conditions. The ability of the method to resolve the very complex flow patterns and associated vorticity dynamics as the valve leaflets open and close illustrates its potential as a powerful tool for patient-specific simulations of native and prosthetic heart valves.

4.5 Conclusions

We have presented a recently developed computational approach for simulating fluid–structure interaction (FSI) problems in complex domains with thin flexible

solid structures. It is based on integrating the sharp interface CURVIB solver, previously developed for FSI problems with rigid structures (Borazjani et al. 2008), with an accurate and efficient rotation-free FE formulation for thin shells (Stolarski et al. 2013) into a coupled FSI framework that is able to handle very large deformations/displacements of thin shell structures. The inverted flag case in particular, which to the best of our knowledge we simulated numerically for the first time (Gilmanov et al. 2015), illustrates the ability of our method to simulate with LES a dynamically rich, high-Reynolds number FSI problem. Comparisons with the measurements of Kim et al. (2013) for this case revealed the ability of the method to capture even subtle features of the flag kinematics, such as the existence of multiple local extrema near the location of maximum deflection, and reproduce wake structures similar to those visualized in the laboratory. We subsequently reviewed results from recent application of our method (Gilmanov et al. 2015, 2018) to simulate the dynamics of a tri-leaflet aortic heart valve placed in an anatomic aorta to demonstrate the capability of the method to solve complex FSI problems in realistic cardiovascular anatomies and at physiologic conditions. Our simulations elucidated the rich vorticity dynamics during the opening of the valve leaflets. The differences for StV and MNY valves' motion and their deformation were shown to give rise to significantly different hemodynamics both near the valve and in the ascending aorta. For the StV valve, the vortex ring is seen to grow in complexity rapidly and ultimately break into turbulence much sooner during the accelerating phase of systole than for the MNY valve. Our simulations show that the heart valve with the StV material model is more obstructive to the blood flow in comparison with the heart valve with the MNY material model.

Acknowledgements We acknowledge the financial support through a grant from the Lillehei Heart Institute at the University of Minnesota, NSF grants IIP-1318201 and CBET-1509071, and US Department of Energy grant (DE-EE 0005482). Computational resources have been provided by the Minnesota Supercomputing Institute and supercomputers of Saint Anthony Falls Laboratory, University of Minnesota.

References

Angelidis D, Chawdhary S, Sotiropoulos F (2016) Unstructured Cartesian refinement with sharp interface immersed boundary method for 3D unsteady incompressible flows. J Comput Phys 325:272–300

Baek H, Karniadakis GE (2012) A convergence study of a new partitioned fluid–structure interaction algorithm based on fictitious mass and damping. J Comput Phys 231:629–652

Barker AT, Cai X (2010) Scalable parallel methods for monolithic coupling in fluid–structure interaction with application to blood flow modeling. J Comput Phys 229:642–659

Bazilevs Y, Hsu M-C, Scott MA (2012) Isogeometric fluid–structure interaction analysis with emphasis on non-matching discretizations, and with application to wind turbines. Comput Methods Appl Mech Eng 249:28–41

Borazjani I (2013) Fluid–structure interaction, immersed boundary-finite element method simulations of bio-prosthetic heart valves. Comput Methods Appl Mech Eng 257:103–116

Borazjani I, Ge L, Sotiropoulos F (2008) Curvilinear immersed boundary method for simulating fluid–structure interaction with complex 3d rigid bodies. J Comput Phys 227(16):7587–7620
Calderer A, Kang S, Sotiropoulos F (2014) Level set immersed boundary method for coupled simulation of air/water interaction with complex floating structures. J Comput Phys 277:201–227
Carmody CJ, Burriesci G, Howard IC, Patterson EA (2006) An approach to the simulation of fluid–structure interaction in the aortic valve. J Biomech 39:158–169
Dettmer W, Perić D (2006) A computational framework for fluid–structure interaction: finite element formulation and applications. Comput Methods Appl Mech Eng 195(41):5754–5779
Donea J, Giuliani S, Halleux JP (1982) An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid–structure interactions. Comput Methods Appl Mech Eng 33:689–723
Farhat C, Lakshminarayan VK (2014) An ALE formulation of embedded boundary methods for tracking boundary layers in turbulent fluid–structure interaction problems. J Comput Phys 263:53–70
Felippa CA, Park KC, Farhat C (2001) Partitioned analysis of coupled mechanical systems. Comput Methods Appl Mech Eng 190(24–25):3247–3270
Fernandez MA, Gerbeau J-F, Grandmont C (2007) A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid. Int J Numer Methods Eng 69:794–821
Gal E, Levy R (2006) Geometrically nonlinear analysis of shell structures using a flat triangular shell finite element. Arch. Comput. Methods Eng. 13:331–388
Ge L, Sotiropoulos F (2007) A numerical method for solving the 3 D unsteady incompressible Navier-Stokes equations in curvilinear domains with complex immersed boundaries. J Comput Phys 225:1782–1809
Ge L, Sotiropoulos F (2010) Direction and magnitude of blood flow shear stresses on the leaflets of aortic valves: is there a link with valve calcification? J Biomech Eng 132
Germano M, Piomelli U, Moin P, Cabot WH (1991) A dynamic subgrid-scale eddy viscosity model. Phys Fluids 3(7):1760–1765
Gilmanov A, Sotiropoulos F (2005) A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. J Comput Phys 207(2):457
Gilmanov A, Sotiropoulos F, Balaras E (2003) A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids. J Comput Phys 191:660–669
Gilmanov A, Le Bao T, Sotiropoulos F (2015) A numerical approach for simulating fluid-structure interaction of flexible thin shells undergoing arbitrarily large deformations in complex domains. J Comput Phys 300:814–843
Gilmanov A, Stolarski H, Sotiropoulos F (2016) Non-linear rotation-free shell finite-element models for aortic heart valves. J Biomech 50:56–62
Gilmanov A, Stolarski H, Sotiropoulos F (2018) Flow-structure interaction simulations of the aortic heart valve at physiologic conditions: the role of tissue constitutive model. J Biomech Eng 140:1003–1012
Griffith BE, Luo X, McQueen DM, Peskin CS (2009) Simulating the fluid dynamics of natural and prosthetic heart valves using the immersed boundary method. Int J Appl Mech 1(01):137–177
Hirt CW, Amsden AA, Cook JL (1974) An arbitrary Lagrangian-Eulerian computing method for all flow speeds. J Comput Phys 14(3):227–253
Hunt JCR, Wray AA, Moin P (1988) Eddies, streams, and convergence zones in turbulent flows. In: Proceedings of the 1988 summer program, Stanford N.A.S.A. Centre for Turbulence Research, CTR-S88, vol 736, pp 193–208
Kamensky D, Hsu MC, Schillinger D, Evans JA, Aggarwal A, Bazilevs Y, Sacks MS, Hughes TJR (2015) An immersogeometric variational framework for fluid–structure interaction: application to bio prosthetic heart valves. Comput Methods Appl Mech Eng 284:1005–1053
Kang S, Lightbody A, Hill C, Sotiropoulos F (2011) High-resolution numerical simulation of turbulence in natural water ways. Adv Water Resour 34:98–113
Kang S, Borazjani I, Colby J, Sotiropoulos F (2012) Numerical simulation of 3d flow past a real-life marine hydrokinetic turbine. Adv Water Resour 39:33–43

Kang S, Yang X, Sotiropoulos F (2014) On the onset of wake meandering for an axial flow turbine in a turbulent open channel flow. J Fluid Mech 744:376–403
Khosronejad A, Sotiropoulos F (2014) Numerical simulation of sand waves in a turbulent open channel flow. J Fluid Mech 753:150–216
Kim D, Cossé J, Cerdeira CH, Gharib M (2013) Flapping dynamics of an inverted flag. J Fluid Mech 736
Küttler U, Wall W (2008) Fixed-point fluid–structure interaction solvers with dynamic relaxation. Comput Mech 43:61–72
Le TB, Sotiropoulos F (2013) Fluid–structure interaction of an aortic heart valve prosthesis driven by an animated anatomic left ventricle. J Comput Phys 244:41–62
Le DV, White J, Peraire J, Lim KM, Khoo BC (2009) An implicit immersed boundary method for three-dimensional fluid–membrane interactions. J Comput Phys 228(22):8427–8445
Luo H, Mittal R, Zheng X, Bielamowicz SA, Walsh RJ, Hahn JK (2008) An immersed-boundary method for flow–structure interaction in biological systems with application to phonation. J Comput Phys 227:9303–9332
Luo H, Yin B, Dai H, Doyle JF (2010) A 3d computational study of the flow–structure interaction in flapping flight. Technical Report. In: 48th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, 4–7 Jan 2010, Orlando, Florida
Macosko CW (1994) Rheology: principles, measurements, and applications. Wiley VCH, New York
May-Newman K, Yin F (1998) A constitutive law for mitral valve tissue. ASME J Biomech Eng 120(1):38–47
Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261
New TH, Tsovolos D (2012) Vortex behaviour and velocity characteristics of jets issuing from hybrid inclined elliptic nozzles. Flow Turbul Combust 89(4):601–625
Newmark N (1959) A method of computation for structural dynamics. J Eng Mech Div 85:67–94
Sacks MS, Schoen FJ, Mayer JE Jr (2009) Bioengineering challenges for heart valve tissue engineering. Annu Rev Biomed Eng 11:289–313
Smith IM, Griffith DV (2004) Programming the finite element method. Willey, New York
Sotiropoulos F, Yang X (2014) Immersed boundary methods for simulating fluid–structure interaction. Prog Aerosp Sci 65:1–21
Stolarski H, Belytschko T, Lee S-H (1995) A review of shell finite elements and co-rotational theories. Comput. Mech. Adv. 2:125–212
Stolarski H, Gilmanov A, Sotiropoulos F (2013) Non-linear rotation-free 3-node shell finite-element formulation. Int J Numer Methods Eng 95:740–770
Tepole AB, Kabari H, Bletzinger K-U, Kuhl E (2015) Isogeometric Kirchhoff-Love shell formulations for biological membranes. Comput Methods Appl Mech Eng 293:328–347
Tian FB, Dai H, Luo H, Doyle JF, Rousseau B (2014) Fluid–structure interaction involving large deformations: 3d simulations and applications to biological systems. J Comput Phys 258:451–469
Timoshenko S, Woinowsky-Krieger S (1959) Theory of plates and shells. McGraw–Hill, New York
Vanella M, Rabenold P, Balaras E (2010) A direct-forcing embedded-boundary method with adaptive mesh refinement for fluid-structure interaction problems. J Comput Phys 229(18):6427–6449
Wang M, Moin P (2002) Dynamic wall modeling for large-eddy simulation of complex turbulent flows. Phys Fluids 14:2043–2051
Wiens JK, Stockie JM (2015) An efficient parallel immersed boundary algorithm using a pseudo-compressible fluid solver. J Comput Phys 281:917–941
Zheng X, Xue Q, Mittal R, Beilamowicz S (2010) A coupled sharp-interface immersed boundary-finite element method for flow–structure interaction with application to human phonation. J Biomech Eng 132:111003
Zhu L, Peskin CS (2002) Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method. J Comput Phys 179(2):452–468

Chapter 5
Handling Slender/Thin Geometries with Sharp Edges in Sharp Interface Immersed Boundary Approach

Pradeep Kumar Seshadri and Ashoke De

Nomenclature

$\hat{n}_v$	Angle weighted pseudo-normal
$\hat{n}_e$	Angle weighted edge normal
$\hat{n}_s$	Outward surface normal
$\bar{\alpha}$	Mean incidence
$\Delta\alpha$	Angular amplitude
α	Pitch angle
f^*	Reduced frequency
t^*	Non-dimensional time
U_∞	Free stream velocity

Abbreviations

IBM	Immersed boundary method
NS	Navier–Stokes
FVM	Finite volume method
AMR	Adaptive mesh refinement
LEV	Leading edge vortex

P. K. Seshadri · A. De (✉)
Department of Aerospace Engineering, Indian Institute of Technology Kanpur, Kanpur 208016, India
e-mail: ashoke@iitk.ac.in

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_5

5.1 Introduction

5.1.1 General Overview

In recent times, non-boundary conforming approaches like immersed boundary method enjoy widespread popularity for its ability to model flow past arbitrarily complex geometries. A major task in such approaches is to inject the description of immersed object onto the underlying background mesh (Cartesian grids). Based on how the forcing is introduced to satisfy the boundary conditions at the immersed interface, the approach can be broadly classified as continuous forcing (diffused interface) (Kumar et al. 2015; Peskin 2002) or discrete forcing (sharp interface) (Choi et al. 2007; Gilmanov and Sotiropoulos 2005; Kumar and Roy 2016; Udaykumar et al. 2001). In the former, a forcing term is added to continuous Navier–Stokes (NS) equation before discretization, while in the latter, the solution field near the interface is directly reconstructed or the cells that are intercepted by immersed surface is reconstituted into non-rectangular control volumes in order to enforce strict conservation laws.

The forcing term in diffused interface approach ensures the satisfaction of interface boundary condition by using Dirac delta function. This ends up spreading the force term over several neighbouring grid nodes. This results in an increase in the effective width of immersed body. Thus, capturing sharp features of geometry becomes difficult with this approach. On the other hand, the sharp interface approach (solution reconstruction as well as cut cell strategy) allows for the exact imposition of boundary condition. The focus of this study is on the solution reconstruction-based sharp interface approach.

Solution reconstruction-based sharp interface approach because of its non-intrusive character is emerging as an attractive class of immersed boundary approach as it can be implemented on any existing flow solver with very little modification. Unlike the cut cell-based approach which involves highly complex geometric operations (especially with regard to moving body problems as it needs to reconstitute the boundary intercept cell at every time instance), flow reconstruction schemes are much simpler in its implementation and formulation. It does not even lead to a significant increase in computational cost. Usually, the flow is reconstructed along surface normal of the immersed object using various interpolation schemes (depending on the flow physics). This class of approach too encounters issues when handling moving body problems. It suffers from spurious force and pressure oscillations. These are attributed to abrupt forcing point role reversals as the immersed object moves through the background mesh.

5.1.2 Handling Thin/Sharp Bodies

A major difficulty in almost all variants of immersed boundary approach is to ensure accurate representation of complex geometries. For instance, the diffused interface approach discussed earlier will smear out the discontinuity around these sharp edges over a number of grid cells (Zhu and Peskin 2002). This would alter the geometry, change the pressure distribution. Kang et al. (2000) pointed out that such smeared out pressure profiles can cause parasitic currents when it is used to make velocity field divergence-free. Cut cell approach may get into stability-related problems as thin/sharp geometries will lead to arbitrarily small cells. Ensuring conservation laws for such small cells is difficult. The irregularity in flux stencils for such small cells can lead to spurious oscillations of pressure and wall shear stresses. Thus, special treatments like cell merging (Seo and Mittal 2011), cell clustering (Muralidharan and Menon 2018) and hybrid of ghost cell and cut cell algorithms (Ji et al. 2008) are proposed to address some of these issues.

In case of solution reconstruction-based approach, the challenge is twofold in representing thin/sharp geometries,

1. Due to infinite curvature at sharp corners, accurate and consistent inside/outside node classification becomes challenging.
2. Lacks enough number of nodes for accurately reconstructing the flow field around sharp corners. Capturing sharp discontinuities becomes difficult as lack of enough number of points reducing order of accuracy of flux terms. Thus, demands more grid resolution for resolving the flow field.

Several works have tried to address this issue. From the standpoint of accurately representing the immersed surface, works of Gilmanov and Sotiropoulos (2005), Choi et al. (2007), Yang and Stern (2013), Senocak et al. (2015), have provided detailed descriptions regarding algorithms that can be utilized for geometric pre-processing. But most of the research work have tried to address issues from solver's standpoint. For instance, Das et al. (2018) suggest ad hoc corrections around sharp corners, reducing order of accuracy of reconstruction schemes. Ghias et al. (2007) and Onishi et al. (2013) proposed arbitrary dummy cell approach which stores the value of virtual ghost cell points which can be utilized in flux calculations, thereby preserving the order of accuracy of the schemes. Balaras and Vanella (2009) and Liu and Hu (2018) proposed adaptive mesh refinement strategies to improve grid resolution near sharp boundaries.

5.1.3 Objective

The objective of the current study is to present a simple and robust set of procedure that can be followed in order to efficiently handle sharp edges. Through a case study involving dynamic stall in oscillating airfoil, the article tries to highlight the possible

issues arising from the nature of the geometry, limitations of the algorithm (both computational geometry as well as solution reconstruction) and possible errors at the implementation level. To demonstrate the capability of our algorithm and also to highlight issues described earlier, we contrast our results obtained with the help of the algorithm proposed by Gilmanov and Sotiropoulos (2005) in his 2005 work.

5.2 Numerical Details

5.2.1 Flow Solver Details

An in-house density-based finite volume flow solver is used in the study. The 3D unsteady Navier–Stokes equation is solved in generalized curvilinear co-ordinate system using a co-located multiblock grid structure. For simulating incompressible and low Mach number flow, a preconditioning strategy is adopted. Low-diffusion flux-splitting scheme is used for discretizing convective fluxes and central difference scheme for viscous fluxes. Time marching is through a dual time stepping approach. A second-order backward three-point differencing is used for discretizing physical time step, while explicit Euler is used for local pseudo-time stepping. Parallel processors communicate using MPI. Further details about the solver can be found in Das and De (2015).

5.2.2 Immersed Boundary Pre-processing Procedure

Immersed Geometry Description

Immersed body is represented by unstructured triangular meshes. A shared list of vertices and a list of triangular elements storing pointers for the vertices are a common way of representing the triangulations. File formats like STL and Neutral use such element-vertex connectivity data structure. As long as meshes are static, this minimum information is sufficient for most of the geometric operations.

In dealing with moving body and fluid–structure interaction problems where large deformation can lead to bad elements, gaps or cracks, sometime even fragmentation, complexity level of geometric operations increases. These operations often require adjacent queries to be answered, local mesh to be edited to discard bad elements and so on. To perform them in an efficient and robust way, a comprehensive data structure is needed.

Half-edge data structures are the most popular data structure among the available for two reasons: one for its fixed size (no dynamic arrays) and another for its performance regarding all the adjacent related queries in constant time. We use a compact array-based half-edge mesh data structure as proposed by Alumbaugh and

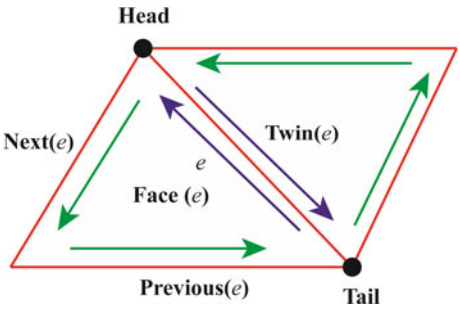


Fig. 5.1 Representation of half-edge data structure

Jiao (2005). This half-edge data structure augments and is constructed efficiently from the standard element-vertex connectivity. Figure 5.1 shows the representation of half-edge data structure. For a given half edge 'e', there is a twin in the adjacent triangle that is oriented in opposite direction. Face containing half edge 'e' has two other half edges, denoted as next and previous. Based on these informations, the following adjacent queries can be answered.

1. For a given vertex, which triangle element uses it?
2. For a given vertex, which edges are incident on it?
3. For a given triangle, what are edges that border it?
4. For a given triangle, what are its adjacent triangles?
5. For a given edge, what are the triangles it shares its edge with?

Some of these queries are needed to efficiently calculate the angle weighted pseudo-normal for vertices and edges which are crucial for treating sharp edges. More on this can be found in Sect. 5.2.3.

Node Classification Algorithm

One of the crucial steps in immersed boundary approach is to accurately classify the nodes as solid, fluid and immersed boundary nodes. In immersed boundary literature, one can find two different approaches to classify the nodes: one, using signed distance function (Choi et al. 2007; Gilmanov and Sotiropoulos 2005; Mittal et al. 2008) and another using ray-casting algorithm (Borazjani et al. 2008; De Tullio et al. 2006). For an immersed body which is closed, smooth and has orientable surfaces, one can use the dot product between line projected from given point 'P' (see Fig. 5.2a) onto the surface (at point 'P_0') and its surface normal. Depending on the sign of the dot product, a given node can be classified as solid or fluid node. But let us assume a situation wherein the immersed boundary has sharp edges as in airfoil (shown in Fig. 5.2a). Consider point A from the shaded region. In order to classify the node, project a line from point 'A' to the surface. Notice that the line falls at the vertex of the surface where the surface normal is discontinuous. The dot product between surface normal $\hat{n}_2$ and the projected line are in the opposite direction, and thus the exterior fluid point will be marked as interior solid point wrongly. The signed distance approach to classify any point in the shaded region shown in Fig. 5.2a will fail. It is worth to note that the triangular meshes are not C^1 continuous at its vertex

and edges, and thus, the normal is undefined at those points. On the other hand, consider the ray-casting approach as shown in Fig. 5.2b, c. In this approach, a ray is cast from a point (say F as in Fig. 5.2b), and the number of intersections it makes with the surface is counted. In this study, we use Moller and Trumbore ray/triangle intersection algorithm (Möller and Trumbore 2005) which solves a linear system of equations to find the barycentric co-ordinates (u, v, w) (see Fig. 5.2c) and the distance from the origin to point of intersection 'P'. As long as the computed value fulfils the barycentric criteria, the intersection point is within the bounds of the triangle. Depending on whether the ray intersects the surface at odd or even number of times, the nodes are classified as exterior or interior. An axis-aligned bounding box (Fig. 5.2b) is implemented to reduce the number of intersection tests as a large number of grid nodes are located outside it. All the nodes outside the bounding box are classified as fluid nodes.

While both signed distance approach and ray-casting approach classifies the nodes as fluid or solid, a separate algorithm is required to tag the immersed boundary nodes. These nodes are the nearest neighbour fluid nodes to the surface on which the solution reconstruction is performed. In the present study, in order to tag immersed boundary nodes, a loop over all solid nodes is performed checking the status of immediate neighbouring nodes. If the immediate neighbour is fluid, then this node is tagged as immersed boundary nodes. Similarly, a loop over all the immersed boundary nodes is performed to identify its immediate solid neighbours. Those solid neighbours are tagged as ghost nodes, which will be used for field extension approach in case of moving body problem.

Closest point computation

Once the classification is done and immersed boundary nodes are identified, an important task is to find the closest surface point to the given immersed node. This is carried out in two-step process. First step involves finding a minimum bounding sphere for triangular element (see Fig. 5.3a) and storing its centre and radius. This radius is then compared with the distance between grid node and centre of this sphere. The one with minimum difference is chosen. In the next step, we use David Eberly's 'distance between point and triangle in 3D' algorithm (Eberly 1999) which defines a squared distance function (Q) for any point on the triangle, T to the point P.

$$Q(u, v, w) = |T(u, v, w) - P|^2 \tag{5.1}$$

This function is a quadratic in barycentric co-ordinates (u, v, w). The closest point is given by the global minimum of Q which occurs when the gradient of Q equals zero. The challenge is to find whether this point is closest to the edge (R2, R3 and R4), vertex (R5, R6, R7) or to the actual face (R1) itself. In all the three cases, finding distance from a point to triangle translates into finding distance to a line, a point or plane, respectively. The index of the closest triangle face, edge or vertex is stored along with point of intersection.

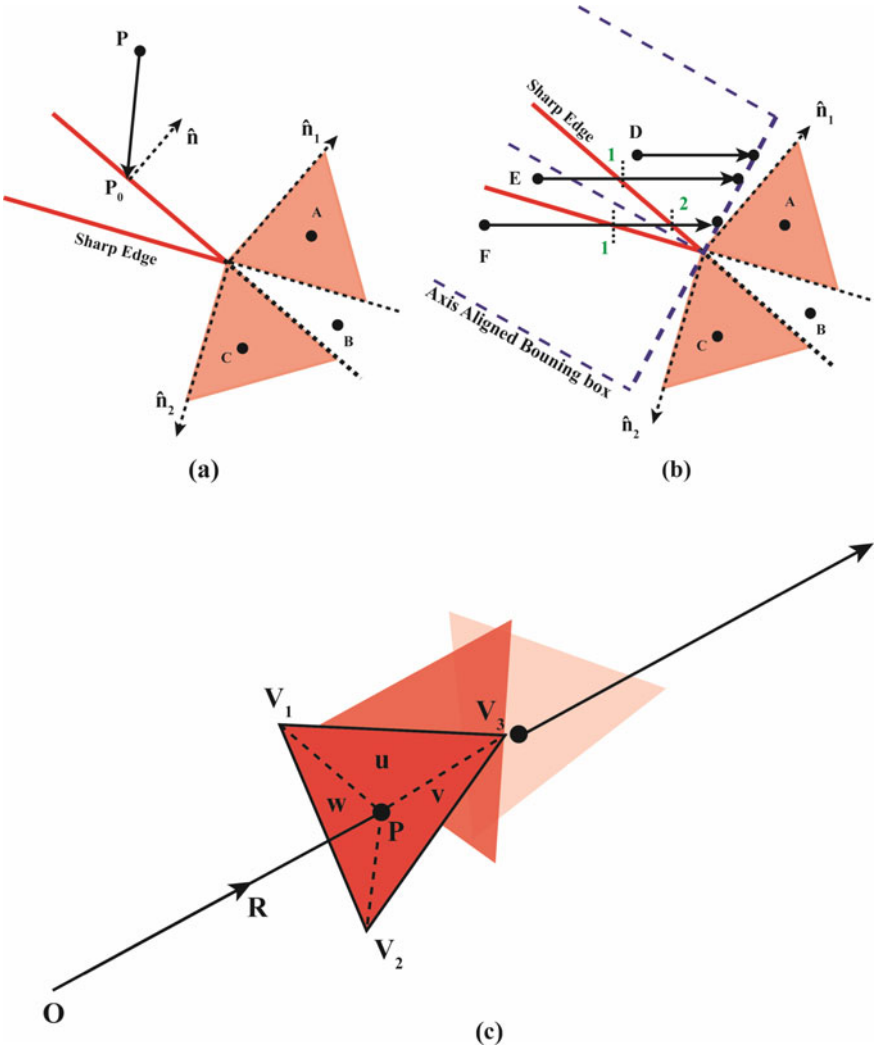


Fig. 5.2 **a** Signed distance calculation for sharp edges. **b** Ray-casting approach for node classification. **c** Ray cast from origin O passes through a number of triangles

5.2.3 Solution Reconstruction

Angle Weighted Pseudo-normals

Before moving further, we would like to re-emphasize few observations from the above discussion.

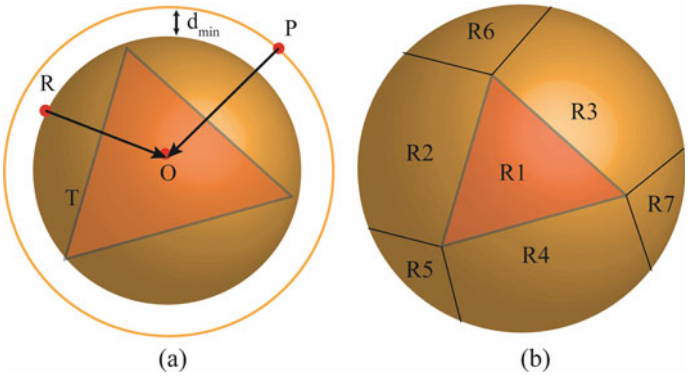


Fig. 5.3 **a** Triangle element (T) bounded by minimum bounding sphere with radius OR and a slightly larger sphere with radius OP. Here, P denotes the nearest neighbour grid node. **b** Seven regions where the projected point from P could lie

1. For a given point, closest distance to a triangle could be a vertex, an edge or face itself.
2. Except for triangle face, vertex or an edge has no well-defined normal as they are not C^1 continuous.
3. Apart from the mesh boundary, if the immersed object itself has concave regions, its exact representation becomes further difficult.

While many of the sharp interface immersed boundary literature (Choi et al. 2007; Yang and Stern 2013; Senocak et al. 2015) makes these observations, most of their concerns are regarding improving the accuracy of node classification. When it comes to solution reconstruction procedure, they apply their reconstruction stencil parallel to surface normal alone irrespective of the fact that the immersed boundary node is close to the edge or vertex. Thus, when sharp-edged regions are encountered, the solution accuracy gets deteriorated as they do not have well-defined normal. Thus, many of these studies tend to focus on improving flux accuracy (Onishi et al. 2013) (by introducing dummy cells around the sharp edge regions), experimenting with different interpolation schemes [linear (Gilmanov et al. 2003), quadratic (Gilmanov and Sotiropoulos 2005), bilinear, trilinear (Mittal and Iaccarino 2005), logarithmic, providing tangential correction (Choi et al. 2007)], adopting local/adaptive mesh refinement (Balaras and Vanella 2009) approaches.

In this study, we define angle weighted pseudo-normal for vertices and edges of all the triangles based on the work of Bærentzen and Aanaes (2005). These pseudo-normal vectors augment the surface normal. Whenever the closest point on the surface is computed, we also store the information regarding which edge/vertex/face is associated with the immersed node with the help of half-edge data structure described earlier in the immersed geometry description of Sect. 5.2.2. While solution reconstruction stencils are applied, they are applied in the direction parallel to associated face/edge/vertex normal.

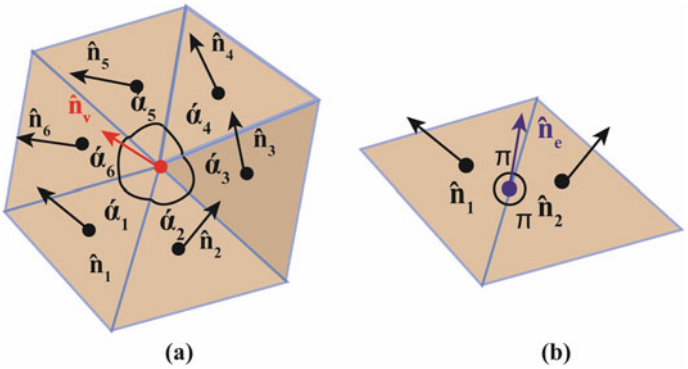


Fig. 5.4 Representation of **a** angle weighted Vertex normal, **b** angle weighted edge normal

Figure 5.4a shows the representation of angle weighted pseudo-normal of a vertex which is defined as

$$\hat{n}_v = \frac{\sum_i \alpha_i \hat{n}_i}{\left\| \sum_i \alpha_i \hat{n}_i \right\|} \tag{5.2}$$

where 'i' denotes the number of incident faces, and α_i is the incident angle.

In case of an edge (see Fig. 5.4b) between face 1 and 2, the angle weighted normal is defined as

$$\hat{n}_e = \pi \hat{n}_1 + \pi \hat{n}_2 \tag{5.3}$$

Direction of Reconstruction Stencil

Figure 5.5 illustrates the direction along which the solution reconstruction stencil is applied. The points P1, P2 and P3 are closest to with vertex V, edge e and face F of the triangular elements, respectively. For reconstructing the solution at node P1, a line parallel to angle weighted vertex pseudo-normal $\hat{n}_v$ projected onto the surface at P1′ is constructed. Similarly for P2, a line parallel to edge normal $\hat{n}_e$ is projected onto the surface at P2′. For P3, a line parallel to surface normal $\hat{n}_s$ is constructed.

Reconstruction Stencils

Flow field variables such as pressure, velocity and temperature are reconstructed at immersed boundary nodes as well as at ghost nodes (in case of moving body problem). A quadratic stencil is applied on the line projected from these nodes to the immersed surface such that it satisfies the boundary conditions at the immersed surface (Seshadri and De 2018). Dirichlet boundary condition is applied for the velocity, and Neumann boundary condition is applied for pressure and temperature.

Sharp interface immersed boundary approach encounters issues of mass conservation and spurious oscillations while modelling moving body problems as the

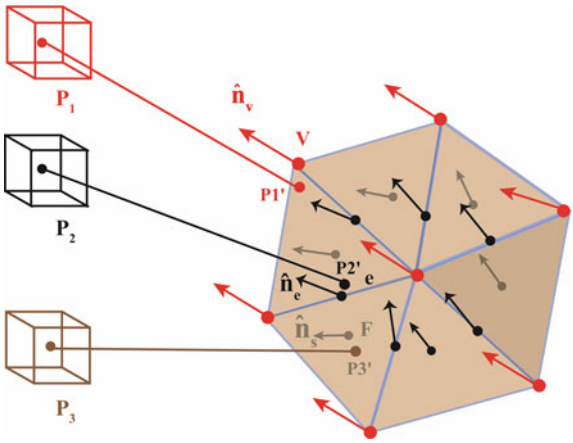


Fig. 5.5 Representation of direction along which reconstruction stencil is applied

approach fails to uphold geometric conservation law. As the immersed object moves through the background Cartesian mesh, the role of these Cartesian grid nodes change suddenly and abruptly at each time instance, i.e. from solid to fluid or fluid to solid. This spatial and temporal discontinuity induces errors resulting in spurious, high-frequency oscillations. In order to address this issue, solid nodes that are immediate neighbour to the immersed surface are marked as ghost nodes, and solutions from the outside field are extrapolated so that when the role change happens, there is a continuity maintained.

5.3 Results and Discussion

In order to demonstrate the efficiency and robustness of our algorithm in handling sharp edges, a detailed comparative study with respect to Gilmanov et al.'s (2005) algorithm is presented in this section. A test case involving dynamic stall of an oscillating airfoil is chosen for this purpose.

5.3.1 Algorithm 1: Gilmanov et al.'s Algorithm

The algorithm is summarized through the following pseudo-code.

1. Input: background Cartesian grid and triangulated surface geometry of immersed body (STL/Neutral format)
2. Determine the face centres of triangular elements and outward surface normal.
3. Locate all Cartesian grid nodes that are in the immediate vicinity of immersed body and within a small prescribed threshold search radius.

4. Gilmanov et al. set search radius approximately equal to the near-body grid spacing. This is found to be inadequate. By examining the grid spacing of the cells that are adjacent to a given node, a maximum of grid spacing is set to be the search radius.
5. For each near-body Cartesian grid nodes, locate the surrounding triangular surface elements. Again within the sphere of search radius prescribed earlier.
6. For a given Cartesian grid node, calculate the signed distance to the face of the associated surface elements.
7. Examine the signs to identify whether the Cartesian grid nodes are inside or outside the immersed body.
8. After the classification of all near-boundary nodes into either immersed nodes or solid nodes, nodes that are interior can also be easily identified by searching along the grid lines. All nodes within two solid nodes will also be solid nodes.
9. For reconstructing solution field, the immersed nodes are projected on to the surface parallel to the surface normal.
10. A quadratic interpolation stencil is imposed along the projected line to obtain the solution field that satisfies the boundary conditions at the interface.
11. For moving body problems, the solution fields are extended inside the solid body by populating the ghost nodes with the information from the fluid region through extrapolation. Again, a quadratic stencil is imposed.

5.3.2 Algorithm 2: Our Present Algorithm

The summary of our algorithm presented here is given below

1. Input: background Cartesian grid and triangulated immersed surface (STL/Neutral format).
2. Based on the element-vertex connectivity information obtained from the surface mesh, establish a half-edge data structure that provides edge connectivity information for robust geometric operations.
3. Determine face centres, surface normal, angle weighted vertex and edge-based pseudo-normal.
4. In order to classify the Cartesian grid nodes as internal or external to the immersed body, first compute bounding box that contains all the vertices defining the immersed surfaces.
5. All nodes that fall outside the bounding box are fluid nodes.
6. Rays are cast from the Cartesian grid nodes that fall within the bounding box to the end of the bounding box in a predefined chosen direction.
7. Check for ray–triangle intersection. If the rays cast from a given Cartesian grid node which does not intersect with triangle, then it is classified as a fluid node.
8. If the ray intercepts with triangle, then the number of interceptions is counted. If the count is odd, it is a solid node and if it is even it is a fluid node.

9. The immersed nodes are identified as fluid nodes that are in the immediate vicinity of solid nodes. The ghost nodes are identified as solid nodes that are in the immediate vicinity of immersed boundary nodes.
10. In order to find the closest surface point to the Cartesian grid node, first determine the distance between Cartesian nodes and face centre of triangular elements. As a first check, this distance is compared to the radius of minimum sphere bounding each of those triangular elements.
11. The triangles which are closer to the nodes are chosen. A quadratic distance function is constructed between the point and triangle. The closest point is obtained by finding global minimum of the quadratic function.
12. The projected point can be close to the edge or vertex and many a times fall on actual face itself. The minimum distance and location of the projected point are stored along with the closest edge, vertex and face information.
13. For solution reconstruction, a line starting from immersed surface passing through immersed node is constructed. This line is constructed such that it is parallel to surface normal or angle weighted vertex or edge normal depending on what is closest to the immersed node.
14. Quadratic interpolation and extrapolation (in case of moving body problems) strategy is followed as described in case 1.

5.3.3 Dynamic Stall of an Oscillating Airfoil

A flow past NACA0012 airfoil pitching about its half chord ($x/c = 0.5$) is chosen as a test case for demonstrating the capabilities of our present algorithm to handle sharp edges even in case of moving body problems. The parameters are chosen from the PIV study of Ohmi et al. (1991). The airfoil begins moving impulsively at $t^* = 0$ (non-dimensional time, $t^* = tU_\infty/c$) and ends at $t^* = 5.0$. The Reynolds number (based on chord length) is 3000. The flow is simulated at a free stream Mach number which is 0.3. The mean incidence $\bar{\alpha}$ and angular amplitude $\Delta\alpha$ of the airfoil are 30° and 15°, respectively. The expression gives the instantaneous angle of attack governing the pitching motion is $\alpha = \bar{\alpha} - \Delta\alpha \cos(2\pi f t)$. The reduced frequency of the pitching oscillation $f^* = fc/2U_\infty$ is 0.1. The Eulerian fluid domain is of size 40c × 30c with 425 × 317 nodes in X–Y plane. The grid is uniformly refined locally near the immersed boundary.

First column in Fig. 5.6 presents the results of Ohmi et al. (1991) obtained by experimental study. The streamline pattern shows the time evolution of unsteady wake past the pitching airfoil. The airfoil impulsively starts its pitching motion from minimum incidence at $t^* = 0$. As the airfoil pitches up, the flow remains attached up to $t^* = 1$. Then, the flow starts separating at the leading edge resulting in the formation of leading edge vortex (LEV). This LEV grows till the airfoil reaches the end of upstroke at $t^* = 2.5$. As the stroke reverses, the growth of LEV stops and it is shed when the airfoil reaches the end of its downstroke at $t^* = 5.0$.

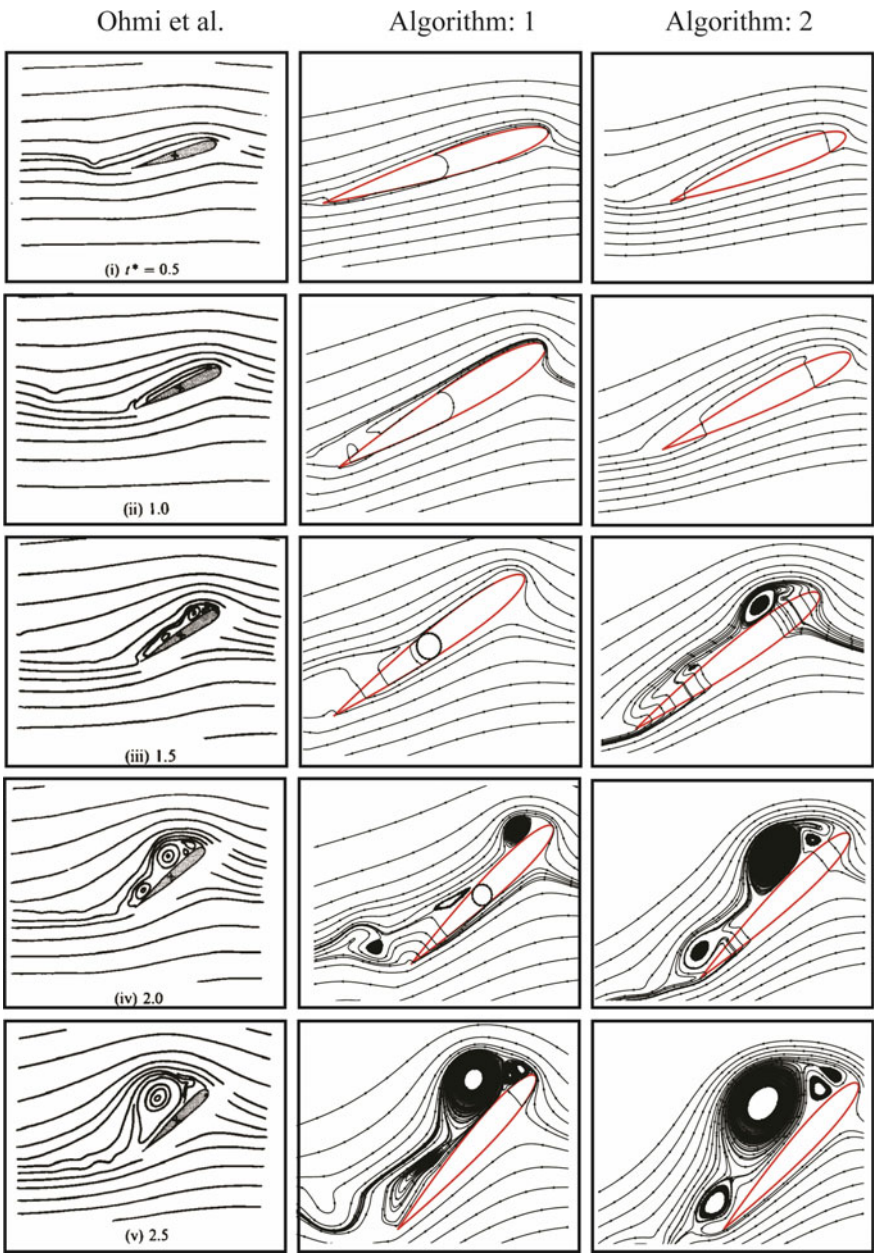


Fig. 5.6 Time evolution of wake past a pitching NACA 0012 airfoil: first column shows PIV results of Ohmi et al. (1991); second column shows Case 1: Gilmanov et al. formulation; third column shows Case 2: present formulation

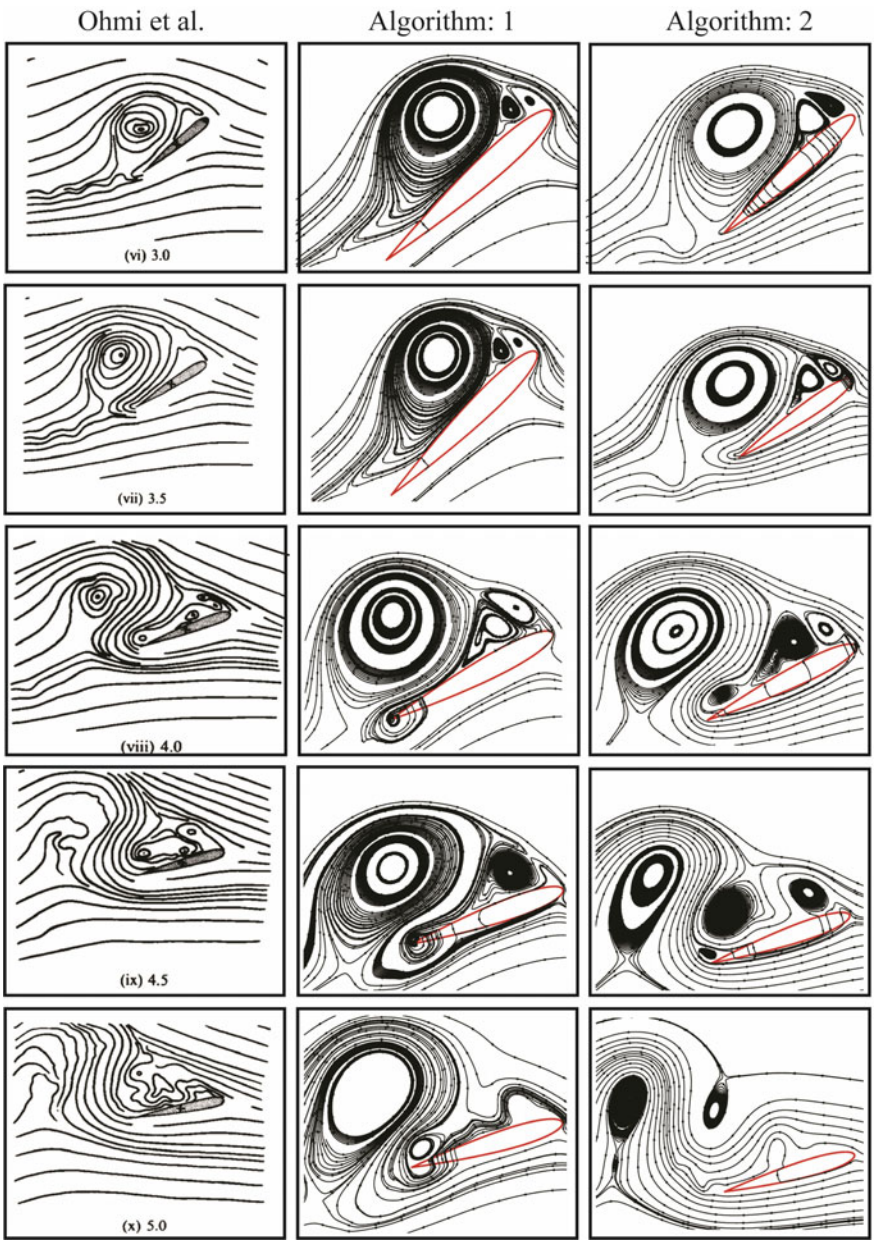


Fig. 5.6 (continued)

Comparing the results from Algorithm 1 and Algorithm 2 with that of Ohmi et al. (1991), one can notice the dynamic stall phenomenon captured by our present algorithm (Algorithm 2) is in excellent agreement with the experimental results. Case 1 algorithm on the other hand fails from the first instance depicted in Fig. 5.6. The flow separates at the trailing edge at $t^* = 0.5$ itself, before it is expected to separate at the leading edge after $t^* = 1$. The flow separation slowly spreads towards upper regions of trailing edge, and from $t^* = 1.5$ onwards, one can notice leading edge vortex which grows till $t^* = 2.5$. As the downstroke begins, the leading edge vortex stops growing and starts shedding. While all the leading edge phenomena are captured well throughout the cycle, one can clearly notice that Algorithm 1 fails to handle the sharp-cornered trailing edge region.

The following sections of the article try to explore the answers to two major questions: where does the Algorithm 1 fail? How does our present algorithm successfully address those issues? The objective is not just to highlight possible sources of errors arising from the limitations of algorithm but also to point out sources of errors arising at the level of implementation.

Immersed Boundary Operations in Interblock Boundaries

Consider earlier observation of Algorithm 1 in Fig. 5.7 where the flow separation at the trailing edge is shown in the instance corresponding to $t^* = 0.5$. A genuine reason for the flow separation could be erroneous handling of sharp corner. But a more careful analysing of the flow field results reveals the flow which separates exactly at the block boundary (as shown in Fig. 5.7a). The U and V contours shown in Fig. 5.7c, d show kinks formed exactly at the block boundary interface. The corresponding plots for the same time instance from Algorithm 2 show that the flow exactly re-attaches at the trailing edge tip. The corresponding U and V contours show smooth distribution without any kinks or disturbances near the trailing edge.

Figure 5.8a shows a streamline plot of pitching airfoil at $t^* = 0.5$ but with fewer block structure. Note that the flow here does not separate at the trailing edge but at the mid-chord as shown in Fig. 5.8b. This observation confirms that the immersed boundary treatment at the block boundaries is the source of the error. In parallel multiblock structured flow solvers, the block boundaries contain layers of dummy cells that exchange and retain information regarding the variables from the adjacent block. This becomes necessary for maintaining order of accuracy of discretization schemes near block boundaries. The number of dummy cell layers depends on the order of discretization schemes involved.

In case of parallel immersed boundary treatment, apart from the information regarding flow variables, additional information regarding tagging, distance function needs to be supplied to these dummy layers. Note the fact that these informations are calculated in every processor that contains blocks that overlap with immersed boundary surface. Each block has access to the information about entire geometry. Although the solution reconstruction is performed only for the physical domain, information from dummy layers are sought for interpolation/extrapolation operations. Thus for an accurate solution reconstruction procedure, the tagging and distance function information provided to these dummy layers should strictly correspond to the values from

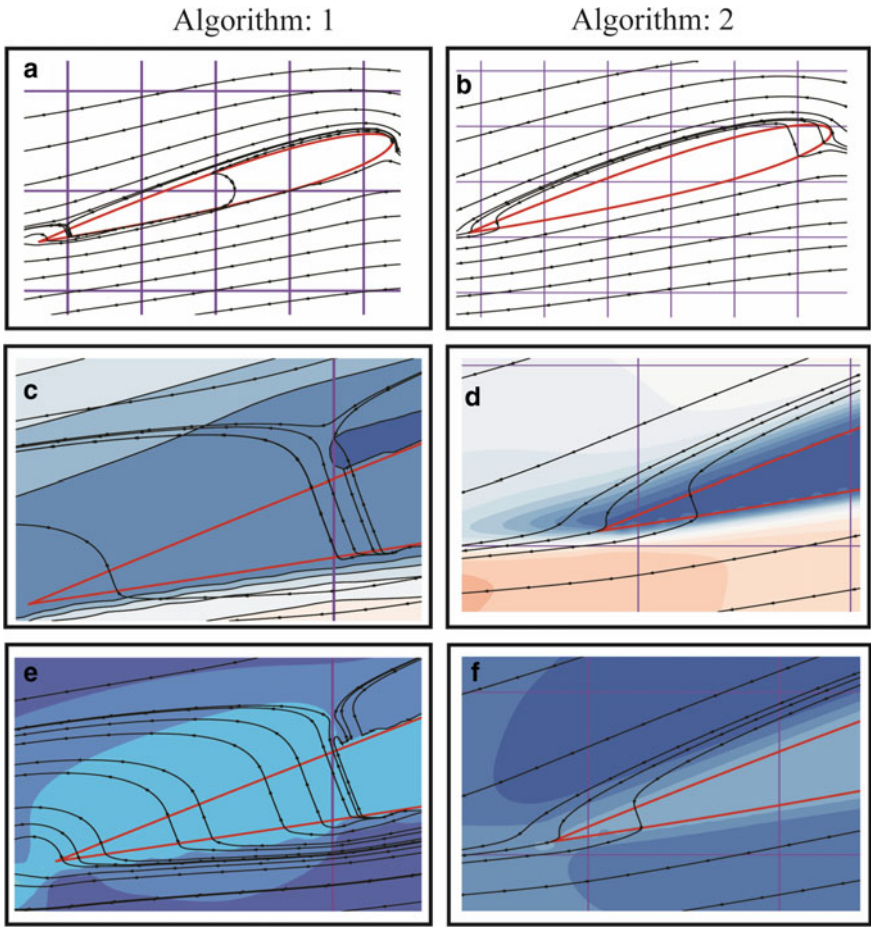


Fig. 5.7 Time evolution of vortices in a pitching NACA 0012 airfoil: Algorithm 1: Gilmanov et al. formulation; Algorithm 2: present formulation

adjacent block whose layer of cells overlaps with that of the dummy layers. Errors from these block boundaries can arise because of two main reasons

1. Number of dummy layers is inadequate for immersed boundary operations which reduce the accuracy of solution near block boundaries.
2. There is inconsistency in tagging, distance function information provided to the dummy cell layers.

In present solver, five layers of dummy cells are shared between the block boundaries which are adequate for implementing fifth-order discretization schemes. Figure 5.8c shows that there is inconsistency in the classification of nodes in dummy layer regions. The four encircled region shows that wherever there is an overlap of adjacent block, there is inconsistency in tagging. Figure 5.9 provides a clearer view of

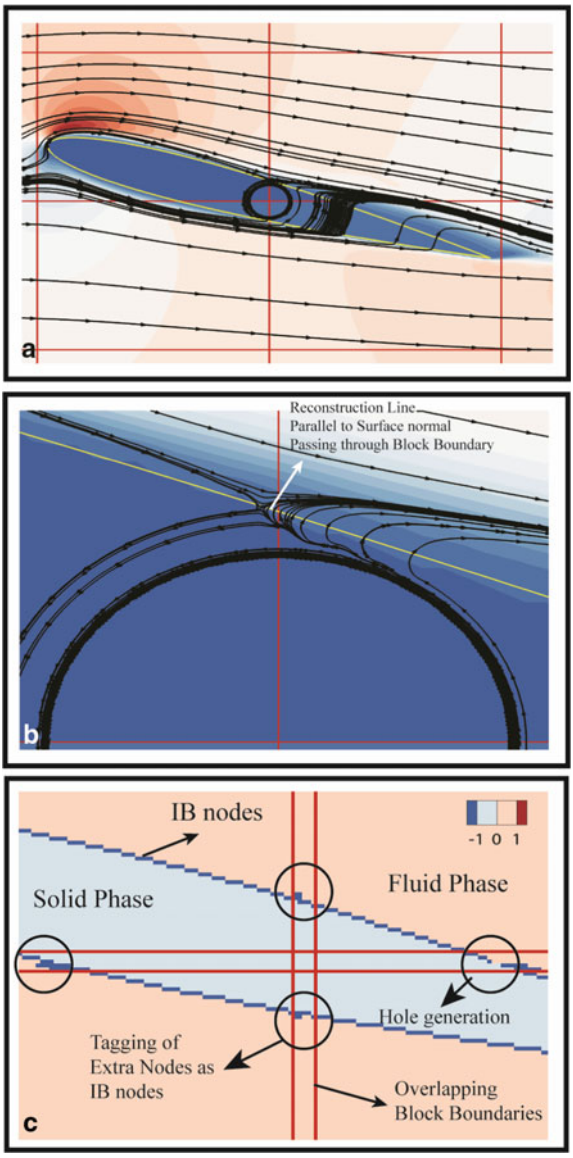


Fig. 5.8 Pitching airfoil at $t^* = 0.5$; **a** fluid Domain with fewer block structure, **b** contour plot showing distribution of nodes, **c** the encircled regions show inconsistent tagging

the region. Note that Fig. 5.9b where Algorithm 1 has tagged fluid nodes as IB nodes and Fig. 5.9c shows the region where the algorithm fails to tag IB node, leading to the hole generation. There could be two possibilities as to why is tagging inconsistent near block boundaries. One, the classification algorithm fails at the block boundaries or the search radius definition used is leading to erroneous tagging.

Figure 5.9d shows that the distribution of search radius near block boundaries is inconsistent with the rest of the physical domain. Figure 5.9e shows 3D contour of

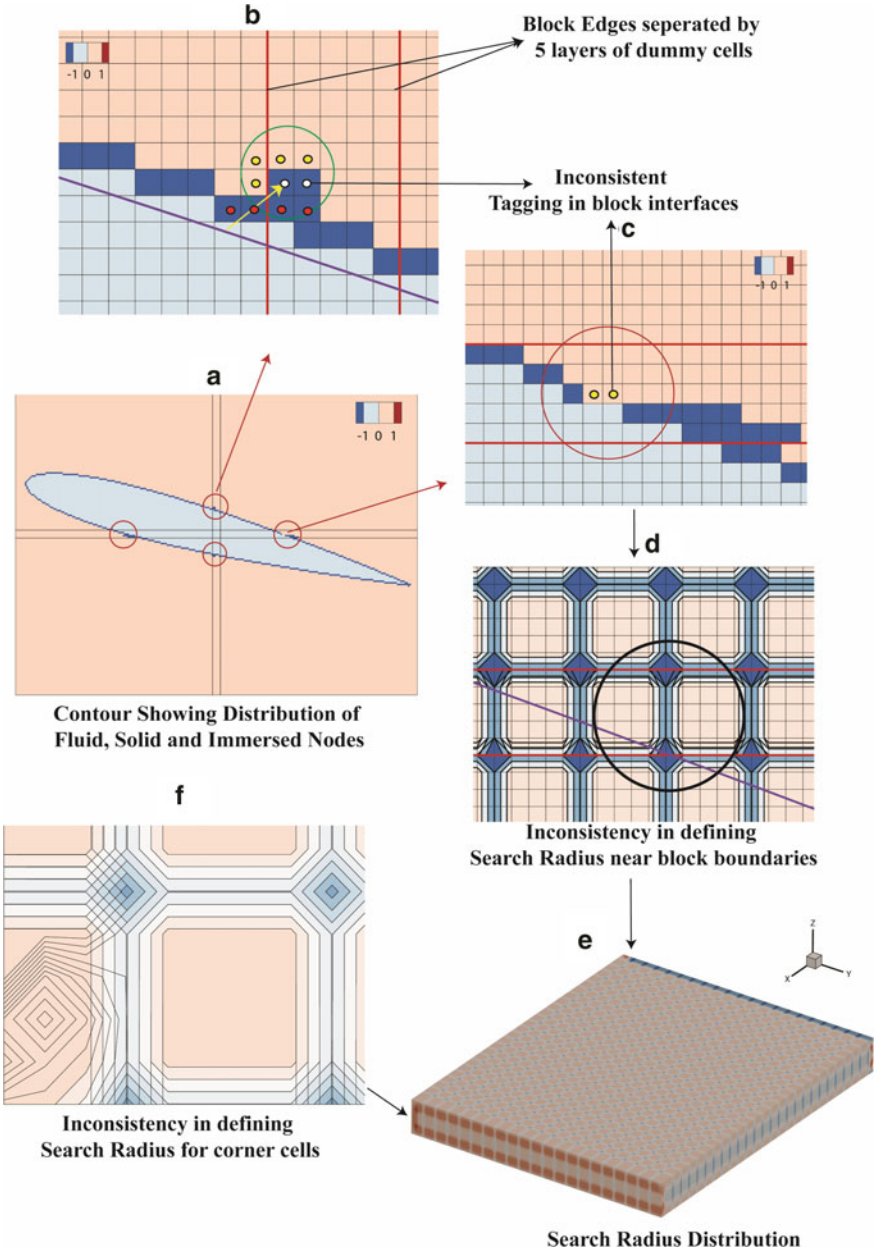


Fig. 5.9 Tagging and search radius distribution in block boundaries

search radius distribution. While distribution is uniform in the rest of the physical domain, near the block boundaries it becomes inconsistent. This is in fact due to the definition of the search radius adopted in Sect. 5.3.1. The number of adjacent cells near block boundaries is less than the interior nodes. Hence, the maximum grid spacing picked by the dummy layers as the search radius would be different from that of the adjacent block layers from the physical domain that overlaps with them. This suggests that search radius distribution in block corners would be more chaotic as there are still lesser adjacent cells. This indeed is shown in Fig. 5.9f. The discussion regarding the classification algorithm is taken up as a separate subsection below. The accurate results from Fig. 5.7b, d, f show that Algorithm 2 which uses minimum bounding sphere strategy to determine the search radius provides consistent definition.

Periodic Boundary Condition

The periodic boundary condition is employed in the solver by rebuilding the connectivity information of the grid such that the periodic faces are considered to be neighbour faces making the periodic boundary treatment implicit. When the block containing periodic face contains the immersed body, as it happens in simulating 2D flows by considering unit span in Z-direction, the immersed boundary treatment on the periodic face becomes not so straight forward. This is mainly because the dummy cell layers populating the periodic faces do not have overlapping immersed body as shown in Fig. 5.10. This is unlike the dummy cell layers that overlap the physical domain where the information regarding the immersed body is available (DC-2 and 3 in Fig. 5.10). A 2D flow demands that information on every section in Z-plane is the same. Thus, the tagging and distance function information calculated for physical boundaries can be copied to the dummy cell layers of periodic faces.

Node Classification Algorithm and Hole Generation

Figure 5.11a shows the flow field of a pitching airfoil at $t^* = 1.5$. Apart from early flow separation, the solution accuracy away from the trailing edge too is deteriorated. Figure 5.11c shows that fluid nodes near trailing edge are being tagged as solid nodes incorrectly. Figure 5.11e provides enlarged view of the tail section. Remember that Algorithm 1 suggests that if for at least one node within the search radius, the computed scalar dot product between surface normal and line drawn from itself to immersed surface is greater than zero, then the node is outside the surface. But this logic fails near sharp edges as discussed in Sect. 5.2.2. The line drawn from a given point to the triangular mesh surface can fall on edge or vertex where the surface normal is not continuous. Especially near sharp corners, one can have the same distance to two triangles from a given point, but the direction of surface normal need not be the same. This makes the sign of dot product, which is calculated with surface normal, ambiguous. On the other hand, Algorithm 2 with its Moller–Trumbore ray-casting algorithm along with winding algorithm provides a robust node classification. Figure 5.11b shows a smooth flow field. The tagging distribution corresponding to that time instance shown in Fig. 5.11d, f is clean without any holes or gaps.

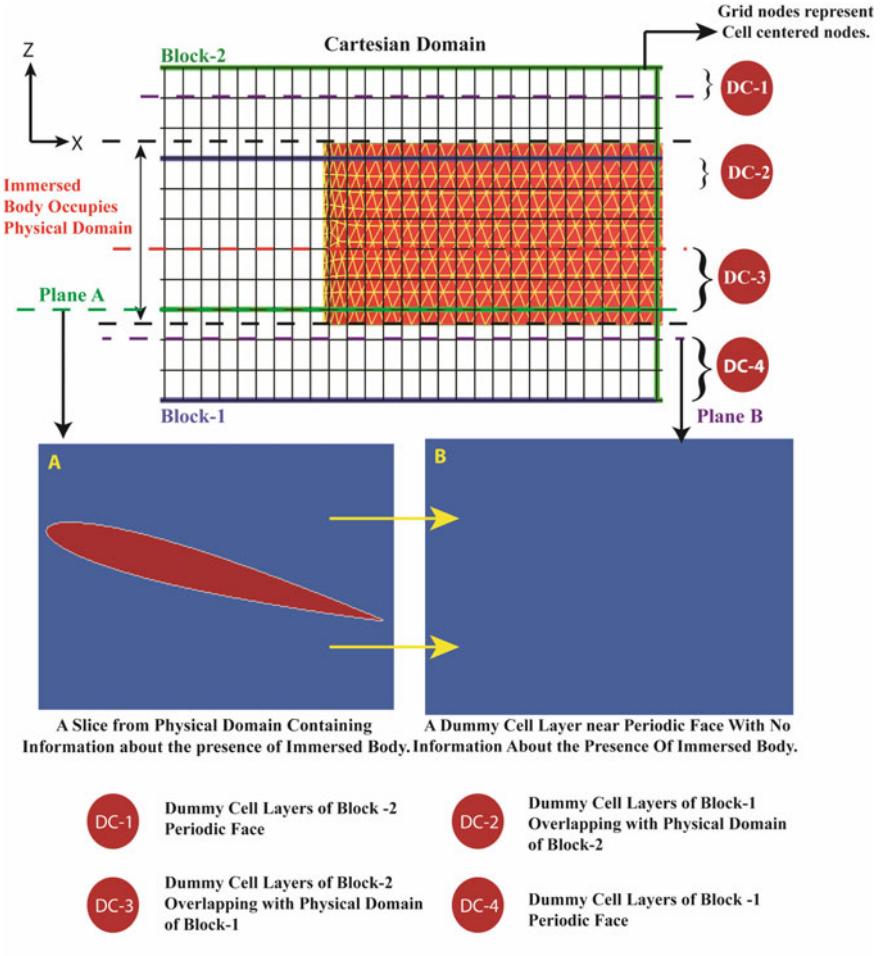


Fig. 5.10 Schemetics showing implementation of periodic boundary conditions

Direction adopted for Solution Reconstruction

To emphasize on the importance of the direction adopted for imposing reconstruction stencil, the pitching airfoil test case is simulated again now with an improvised Algorithm 1. Improvisation is done on two aspects.

1. Without changing the definition of the search radius, the variable is shared across its block boundary from physical domain of adjacent block to the dummy layers of a given block like any other flow variable which is shared in a parallel environment.
2. A strict conservative bounding box is imposed to avoid any hole or gap formation near its trailing edge.

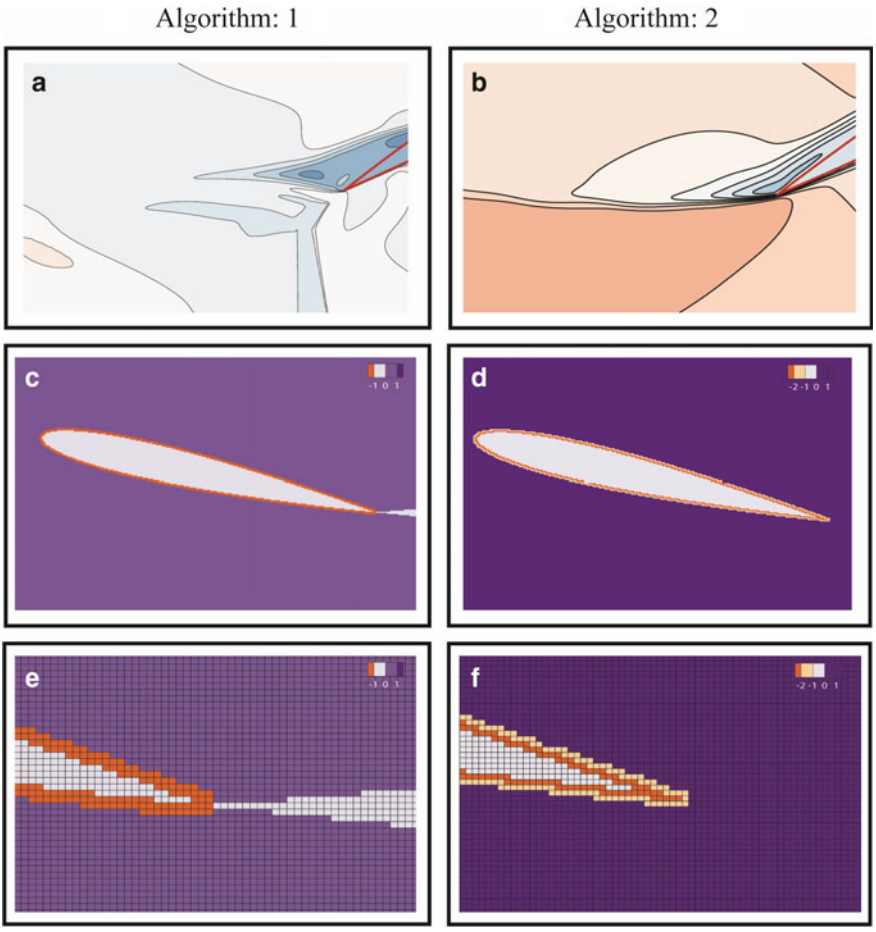


Fig. 5.11 Contrasting signed distance node classification (Algorithm 1) with ray-casting approach (Algorithm 2)

The results from the improvised algorithm are compared with the literature at three instances $t^* = 1.5$, 2.5 and 4.0. Figure 5.12 corresponding to time instance $t^* = 1.5$ shows that the improvised Algorithm 1 provides much better results compared to previous results. It is able to predict the formation of trailing edge vortex just below the mid-chord region. This result is almost identical with Algorithm 2 and Ohmi et al.'s (1991) experimental results. Also, these improvised results are better in its capturing of trailing edge vortex formation than Kumar and Roy (2016) whose results are based on sharp interface IB approach using incompressible flow solver or Akbari and Price (2003) work which is based on boundary confirming approach.

At time instance $t^* = 2.5$ (shown in Fig. 5.13), it is expected from Ohmi et al.'s (1991) results that trailing edge vortex and leading edge vortex almost coalesces into a big vortex when it reaches its peak amplitude. The results from improvised

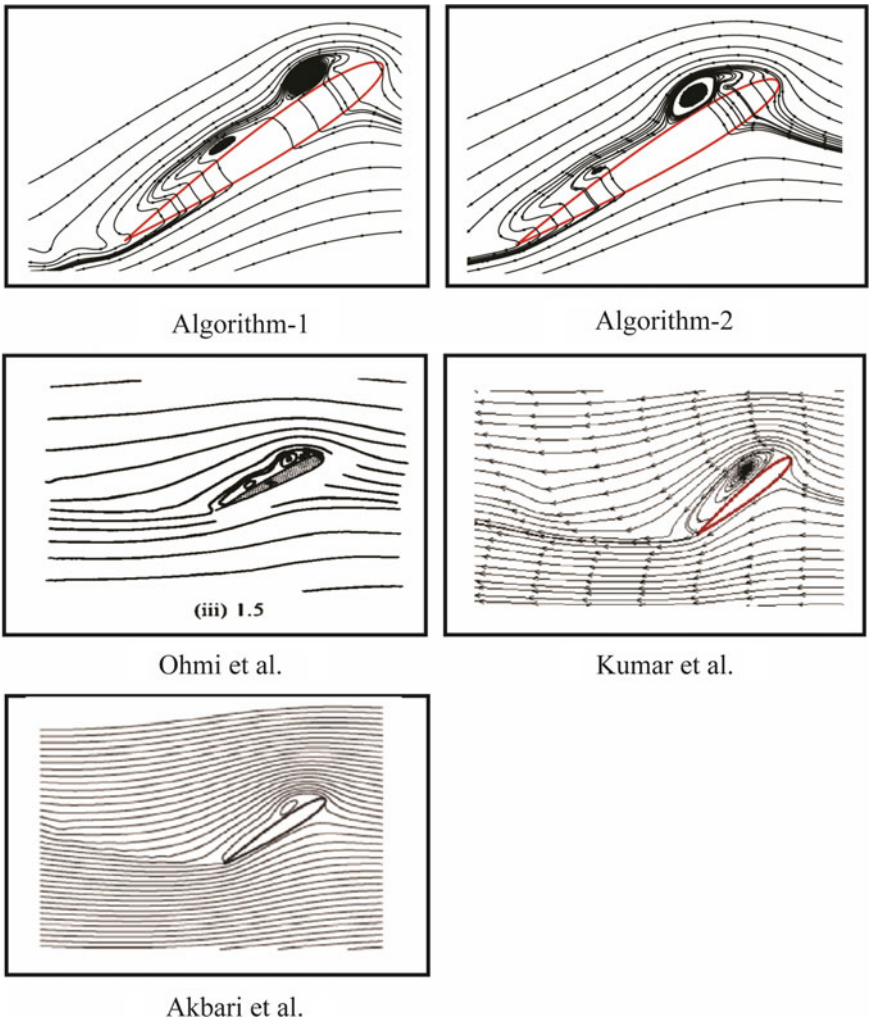


Fig. 5.12 Streamline plots of pitching airfoil at $t^* = 1.5$

Algorithm 1 show the trailing edge vortex which is still strong and has not started interacting with the leading edge vortex. Algorithm 2 shows that the trailing edge vortex is almost engulfed by the leading edge vortex. Since no trailing edge vortex is formed in Kumar and Roy (2016), the leading edge vortex formed grows to occupy the entire chord. Akbari and Price (2003) results show the presence of trailing edge vortex.

At the time instance $t^* = 4.0$ (shown in Fig. 5.14), the downstroke motion has led to the shedding of leading edge vortex. From Ohmi et al.'s (1991) results, one can notice the presence of leading edge vortex, a triangular vortex at mid-chord and

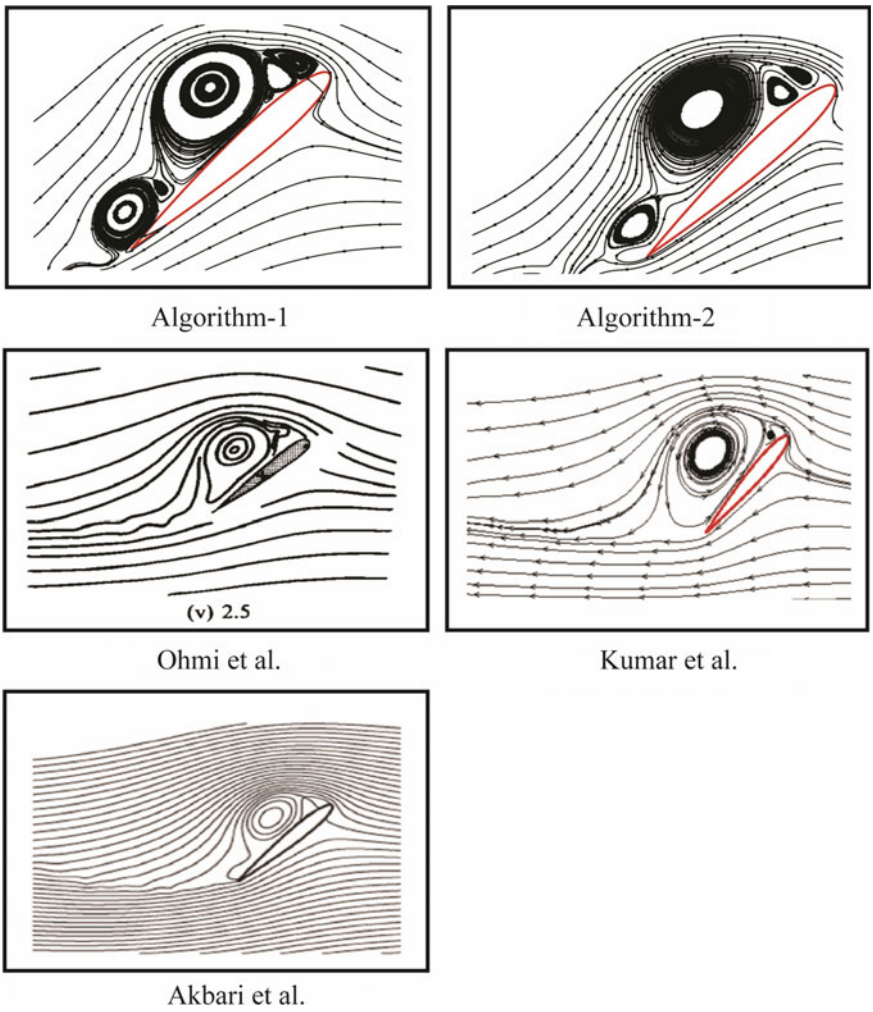


Fig. 5.13 Streamline plots of pitching airfoil at $t^* = 2.5$

a trailing edge vortex. Algorithm 2 and Akbari and Price (2003) results agree well with the experimental observations. But the results of improvised Algorithm 1 do not. The trailing edge vortex is too small. The mid-chord vortex has moved away from the surface and is coalescing with the shed vortex. The leading edge vortex is much broader. Kumar and Roy (2016) results on the other hand show the presence of mid-chord vortex and leading edge vortex. But the vortex near the trailing edge is not on the surface but in the wake.

Figure 5.15 shows the velocity contour plot corresponding to the three time instances we discussed above. At $t^* = 1.5$ and $t^* = 2.5$, the results from improvised Algorithm 1 show that the wake region is chaotic even though the flow field

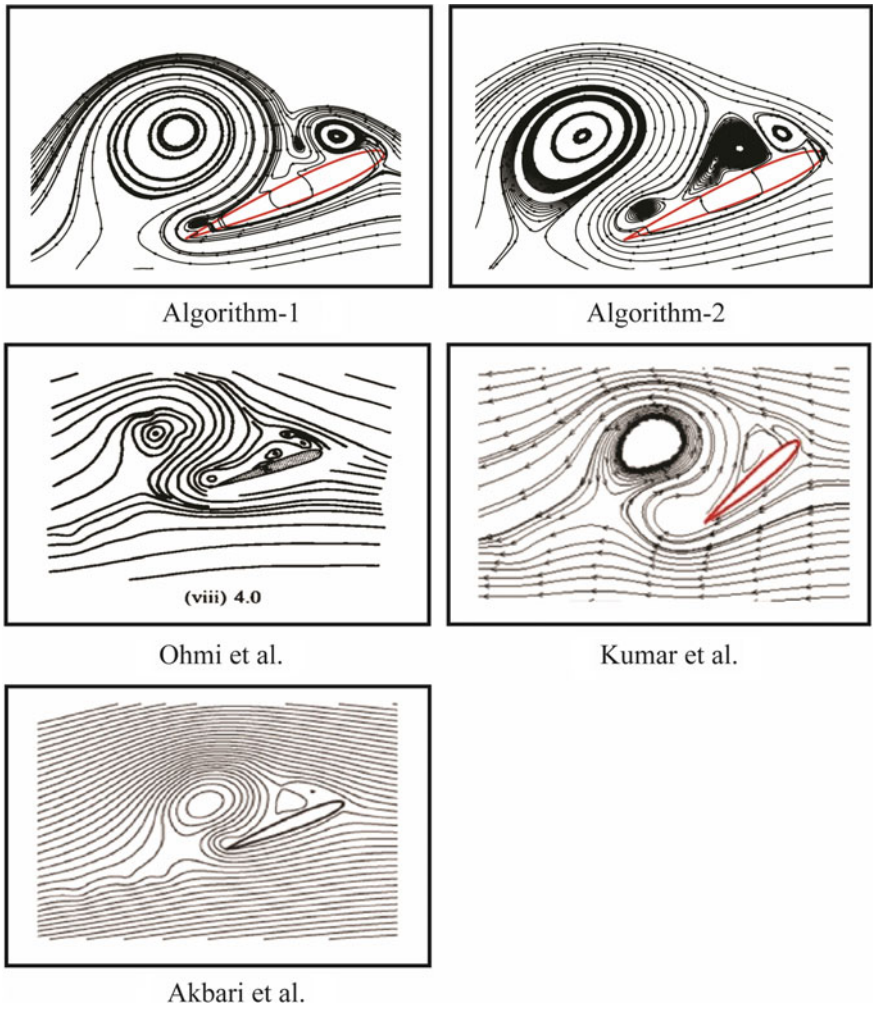


Fig. 5.14 Streamline plots of pitching airfoil at $t^* = 4.0$

around the body is captured accurately. At $t^* = 1.5$, Fig. 5.15a shows an unphysical patch of low-velocity region at the trailing edge tip. Flow field corresponding to $t^* = 4.0$ (Fig. 5.15e) is comparable with the results from Algorithm 2 (Fig. 5.15f).

Figure 5.16 shows the pressure co-efficient distribution around the leading edge of the airfoil at $t^* = 2.5$. Results of Algorithm 1 show that airfoil surface is not sharply represented just like its trailing edge part. The immersed body is actually larger than the actual geometry of the airfoil. In case of Algorithm 2, the geometry is sharply represented. Though the solution reconstruction stencil adopted is the same for both the algorithm, there is a drastic difference in the quality of solutions obtained between these two algorithms. This is true not just in the near-body region but also in wake

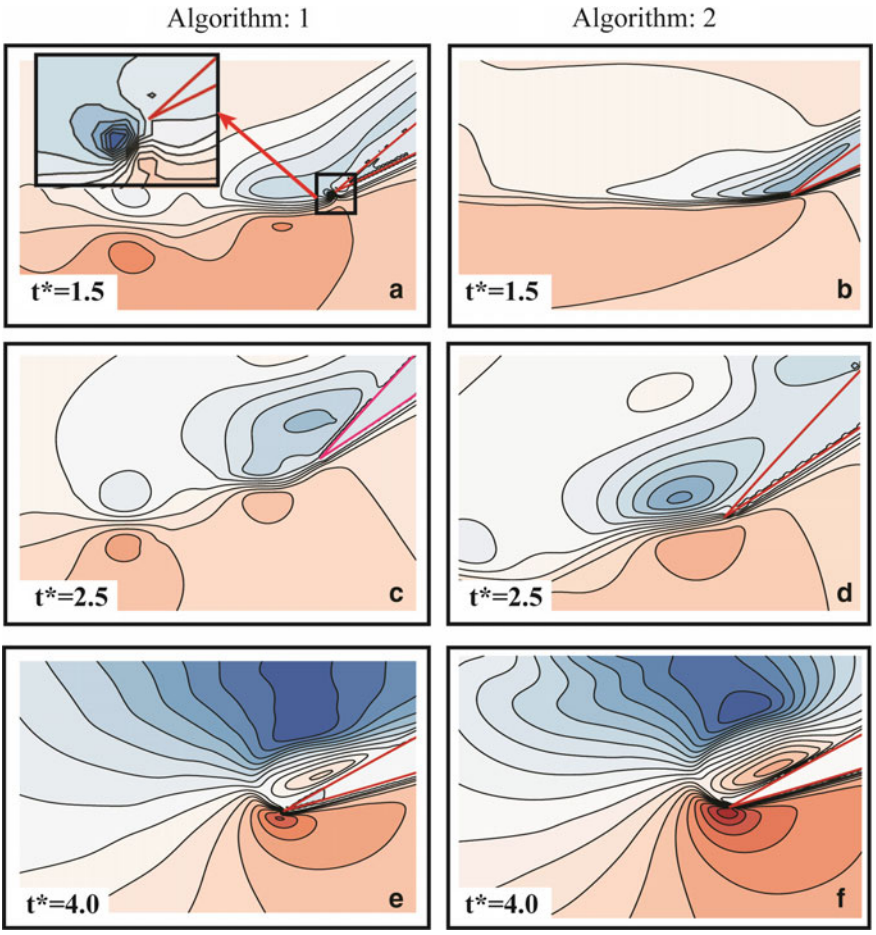


Fig. 5.15 U-V contour plot at time instances $t^* = 1.5$, 2.5 and 4.0

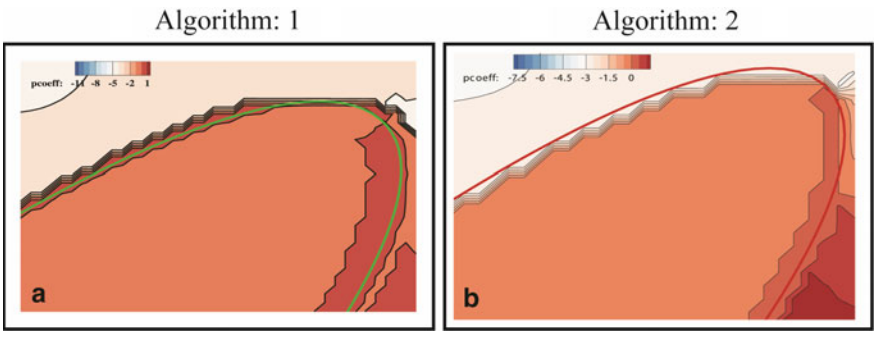


Fig. 5.16 Pressure contour plot corresponding to $t^* = 2.5$

region. This difference is attributed to the direction along which those reconstruction stencils are adopted. In case of Algorithm 1, it is parallel to the direction of surface normal. In case of Algorithm 2, it can be parallel to the direction of surface or vertex or edge normal depending on where the closest point in triangle lie. This enhances the accuracy of geometry representation especially in modelling sharp edges.

5.4 Conclusion

In this study, we have presented our simple and robust algorithm to handle the representation of sharp edges while adopting sharp interface immersed boundary approach. By appreciating the nature of sharp-edged geometries as well as the inherent limitations of representing immersed body with triangular meshes, we have adopted a set of computational geometry procedures that takes care of even the extreme situations when dealing with node classification or exact close point in the triangle, without any ambiguity. With the help of a pitching airfoil test case, we have systematically presented the important role played by such geometry processing algorithms. Their role is not just restricted to geometry pre-processing step as is the case in most of the immersed boundary approach. They play a crucial role in the solution reconstruction procedure as well. The versatile nature of our algorithm is successfully demonstrated by comparing the test case results with the algorithm adopted by Gilmanov et al.

Acknowledgements The authors would like to acknowledge the IITK computer centre (www.iitk.ac.in/cc) for providing support to perform the computation work, data analysis and article preparation.

References

Akbari MH, Price SJ (2003) Simulation of dynamic stall for a NACA 0012 airfoil using a vortex method. J Fluids Struct 17:855–874

Alumbaugh TJ, Jiao X (2005) Compact array-based mesh data structures. In: Proceedings of the 14th international meshing roundtable, Springer, pp 485–503

Bærentzen JA, Aanaes H (2005) Signed distance computation using the angle weighted pseudonormal. IEEE Trans Vis Comput Graph 11:243–253

Balaras E, Vanella M (2009) Adaptive mesh refinement strategies for immersed boundary methods. In: 47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, pp 162

Borazjani I, Ge L, Sotiropoulos F (2008) Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies. J Comput Phys 227:7587–7620

Choi J-I, Oberoi RC, Edwards JR, Rosati JA (2007) An immersed boundary method for complex incompressible flows. J Comput Phys 224:757–784

Das P, De A (2015) Numerical investigation of flow structures around a cylindrical afterbody under supersonic condition. Aerosp Sci Technol 47:195–209

Das S, Panda A, Deen NG, Kuipers JA (2018) A sharp-interface immersed boundary method to simulate convective and conjugate heat transfer through highly complex periodic porous structures. Chem Eng Sci 191:1–18

De Tullio M, Christallo A, Balaras E, Pascazio G, Iaccarino G, Napolitano M (2006) Recent advances in the immersed boundary method

Eberly D (1999) Distance between point and triangle in 3D. Magic Softw http://www.magic-software.com/Documentation/pt3tri3.pdf

Ghias R, Mittal R, Dong H (2007) A sharp interface immersed boundary method for compressible viscous flows. J Comput Phys 225:528–553

Gilmanov A, Sotiropoulos F (2005) A hybrid cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. J Comput Phys 207:457–492

Gilmanov A, Sotiropoulos F, Balaras E (2003) A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids. J Comput Phys 191:660–669

Ji H, Lien FS, Yee E (2008) A robust and efficient hybrid cut-cell/ghost-cell method with adaptive mesh refinement for moving boundaries on irregular domains. Comput Methods Appl Mech Eng 198:432–448

Kang M, Fedkiw RP, Liu XD (2000) A boundary condition capturing method for multiphase incompressible flow. J Sci Comput 15:323–360

Kumar M, Roy S (2016) A sharp interface immersed boundary method for moving geometries with mass conservation and smooth pressure variation. Comput Fluids 137:15–35

Kumar SP, De A, Das D (2015) Investigation of flow field of clap and fling motion using immersed boundary coupled lattice Boltzmann method. J Fluids Struct 57:247–263

Liu C, Hu C (2018) An adaptive multi-moment FVM approach for incompressible flows. J Comput Phys 359:239–262

Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261

Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, Von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227:4825–4852

Möller T, Trumbore B (2005) Fast, minimum storage ray/triangle intersection. In: ACM SIGGRAPH 2005 courses, ACM, pp 7

Muralidharan B, Menon S (2018) Simulation of moving boundaries interacting with compressible reacting flows using a second-order adaptive cartesian cut-cell method. J Comput Phys 357:230–262

Ohmi K, Coutanceau M, Daube O, Loc TP (1991) Further experiments on vortex formation around an oscillating and translating airfoil at large incidences. J Fluid Mech 225:607–630

Onishi K, Obayashi S, Nakahashi K, Tsubokura M (2013) Use of the immersed boundary method within the building cube method and its application to real vehicle cad data. In: 21st AIAA computational fluid dynamics conference, pp 2713

Peskin CS (2002) The immersed boundary method. Acta Numer 11:479–517

Senocak I, Sandusky M, DeLeon R, Wade D, Felzien K, Budnikova M (2015) An immersed boundary geometric preprocessor for arbitrarily complex terrain and geometry. J Atmos Ocean Technol 32:2075–2087

Seo JH, Mittal R (2011) A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. J Comput Phys 230:7347–7363

Seshadri PK, De A (2018) Assessment of pressure reconstruction schemes in sharp interface immersed boundary method. In: AIP conference proceedings, AIP Publishing, pp 030002

Udaykumar H, Mittal R, Rampunggoon P, Khanna A (2001) A sharp interface cartesian grid method for simulating flows with complex moving boundaries. J Comput Phys 174:345–380

Yang J, Stern F (2013) Robust and efficient setup procedure for complex triangulations in immersed boundary simulations. J Fluids Eng 135:101107

Zhu L, Peskin CS (2002) Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method. J Comput Phys 179:452–468

Chapter 6
Ghost Fluid Lattice Boltzmann Methods for Complex Geometries

Arpit Tiwari, Daniel D. Marsh, and Surya P. Vanka

6.1 Lattice Boltzmann Method

Lattice Boltzmann method (LBM) (Chen and Doolen 1998; Luo 2000) has emerged as a powerful alternate computational tool for simulating microscopic and macroscopic flows in complex configurations. In conventional computational fluid dynamics (CFD) methods, the Navier–Stokes equations describing the continuum behavior of fluid flows are solved numerically. The equations describing the conservation of mass, momentum and energy are solved to determine the macroscopic variables (velocity, pressure and temperature). On the other hand, LBM is a meso-scale method which solves reduced versions of the microscopic Boltzmann kinetic equations for particle distribution functions. Simplified kinetic models are developed that retain only specific details of the molecular motion sufficient to recover macroscopic hydrodynamic behavior. LBM is, therefore, an intermediate approach between the continuum and the more fundamental approach of molecular dynamics (MD) simulations.

LBM evolved from lattice gas automata (LGA), in which a simplified kinetic model is constructed for simulating fictitious particles in discrete lattice space and time. The LGA model proposed by Frisch et al. (1986) consists of a two-dimensional equilateral triangular lattice space with hexagonal symmetry. Particles point toward the nearest lattice site; the kinetic model consists of collision and streaming based on certain rules. LGA is based on Boolean operation, thus suffers from statistical noise. This problem was cured by replacing Boolean particle distribution variables with ensemble-averaged particle distribution functions (McNamara and Zanetti 1988), which formed the basis of LBM. However, the primitive formulations of LBM were computationally inefficient because of the complexity of the collision operator in

A. Tiwari (✉) · D. D. Marsh
Gamma Technologies LLC, Westmont, IL 60559, USA
e-mail: arpit01@gmail.com

S. P. Vanka
University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_6

the ensemble form. The major breakthrough in efficiency was achieved through the linearization of the collision operator (Higuera and Jiménez 1989) assuming local equilibrium. This was further simplified using the Bhatnagar–Gross–Krook (BGK) approximation (Bhatnagar et al. 1954) of single relaxation time toward equilibrium leading to the lattice Bhatnagar–Gross–Krook (LBGK) model; the local equilibrium functions are chosen such that macroscopic equations are recovered (Qian et al. 1992; Chen et al. 1992).

Since LBM is an intermediate approach between the macroscopic and microscopic methods, the Navier–Stokes equations can be obtained by carrying out multi-scale expansion of the LB equations (Chen and Doolen 1998; Luo 2000). Similarly, LB equations can be derived from the continuum Boltzmann BGK equations, in which the equilibrium distribution is described by the Boltzmann–Maxwellian function. Low Mach number reduction of Boltzmann BGK equations leads to LB equations (Chen and Doolen 1998; Luo 2000). There is an extensive literature on analysis and advancement of LBM for various applications.

6.1.1 Basic Formulation of LBM

In LBM, particle distribution functions are advanced in time via two processes: collision and streaming. Various advanced formulations have been developed over the years for these processes (e.g., multi-relaxation-time LBM and entropic LBM). However, since the focus of this book in on boundary conditions, for conciseness, we present a widely used basic formulation of LBM here—the single relaxation-time D2Q9 model. As the name suggests, it is a two-dimensional model, in which the collision process employs a single relaxation-time parameter, and particles are restricted to move along nine velocity vectors during streaming, as shown in Fig. 6.1

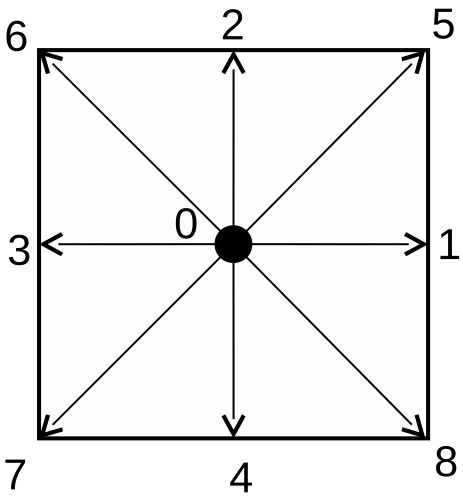


Fig. 6.1 D2Q9 model of LBM

(see Sect. 6.4.3 for a three-dimensional LBM implementation). The discrete equation governing these processes is

$$f_i(\mathbf{x} + c\,\mathbf{v}_i\Delta t, t + \Delta t) - f_i(\mathbf{x}, t) = -\Omega_i, \quad i = 0, 1, \ldots, 8, \tag{6.1}$$

in which, the left- and right-hand sides represent streaming and collision processes, respectively. f_i is the discrete particle distribution function, $\mathbf{x}$ is the spatial location vector, $\mathbf{v}_i$ is the particle velocity, t is time and Δt is the time step. $c = \Delta x/\Delta t$ is the lattice speed, where Δx is the lattice spacing. Ω_i denotes discrete collision operation. The nine velocities are given by

$$\mathbf{v}_i = \begin{cases} (0,0) & \text{for } i = 0, \\ (\cos((i-1)\pi/2),\ \sin((i-1)\pi/2)) & \text{for } i = 1 \text{ to } 4, \\ \sqrt{2}(\cos((i-5)\pi/2 + \pi/4),\ \sin((i-5)\pi/2 + \pi/4)) & \text{for } i = 5 \text{ to } 8. \end{cases} \tag{6.2}$$

The collision term Ω_i can take various forms provided the conservation laws are obeyed; the linearized collision function based on BGK approximation takes the form

$$\Omega_i = \frac{f_i - f_i^{\text{eq}}}{\tau}, \tag{6.3}$$

where τ is a relaxation-time parameter, and f_i^{eq} is the equilibrium particle distribution function:

$$f_i^{\text{eq}} = w_i \rho \left(1 + \frac{c\,\mathbf{v}_i \cdot \mathbf{u}}{c_s^2} + \frac{(c\,\mathbf{v}_i \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u}^2}{2c_s^2}\right), \tag{6.4}$$

where $c_s = c/\sqrt{3}$ is the lattice speed of sound and w_i is the weighing function:

$$w_i = \begin{cases} 4/9 & \text{for } i = 0, \\ 1/9 & \text{for } i = 1 \text{ to } 4, \\ 1/36 & \text{for } i = 5 \text{ to } 8. \end{cases} \tag{6.5}$$

ρ and $\mathbf{u}$ are the density and flow velocity, respectively, obtained from the particle distribution functions and velocities using

$$\rho = \sum f_i \text{ and } \rho\,\mathbf{u} = \sum c\, f_i\, \mathbf{v}_i. \tag{6.6}$$

Applying a Chapman–Enskog procedure on the LB equations, macroscopic continuity and momentum equations can be derived in the low Mach number limit, with pressure (p) and kinematic viscosity (ν) given by $p = \rho\, c_s^2$ and $\nu = (\tau - 1/2)\, c_s^2\, \Delta t$.

6.1.2 Advantages of LBM

LBM offers a number of advantages compared to the conventional CFD methods. The simplified kinetic model utilized in LBM has a linear streaming operator, while Navier–Stokes equations have a nonlinear convection term. The nonlinearity associated with the collision operator in LBM is localized, which makes it highly suited for parallelization (see Sect. 6.4.4 for a discussion on parallelization using GPUs). In LBM, pressure variations are implicitly expressed as a function of density variations, thus eliminating the well-known pressure–velocity coupling issue that needs special treatment in conventional incompressible CFD solvers. Furthermore, since LB equations are obtained by reducing Boltzmann equations, (1) micro-scale multiphase physics are easier to incorporate in LBM and (2) coupling with molecular dynamics (MD) simulations is straightforward (discussed in Sect. 6.4.5). It is thus widely used in the analysis of complex fluids and multi-phase flows. Conventional solvers need special care in dealing with multi-phase flows (Shukla et al. 2010; Tiwari et al. 2013). LBM also offers advantages compared to molecular simulations. In LBM, the simplified kinetic model utilizes a small set of velocities in the phase space. This makes computations significantly faster compared to solving the Boltzmann equations of molecular motion derived from the kinetic theory utilizing the Boltzmann–Maxwellian equilibrium distribution function, where the phase space is continuous and infinite.

6.2 Boundary Conditions in LBM

Accurate implementation of boundary conditions is challenging in LBM due to the difficulty in obtaining particle distribution functions from the prescribed hydrodynamic conditions at the boundaries. This is because the number of unknown particle distribution functions is typically more than the number of hydrodynamic boundary conditions. For stationary walls, the basic implementation of the well-known bounce-back condition simply inverts the particle velocities at the wall. However, it is only first-order accurate, and not applicable to moving walls. Several advancements have been proposed to improve its applicability and accuracy (Ziegler 1993; Ladd 1994; Filippova and Hänel 1998; Mei et al. 1999; Ginzburg and d'Humieres 2003; Lallemand and Luo 2003; Yu et al. 2003).

In the so-called hydrodynamic boundary implementation (Noble et al. 1995), the unknown (incoming) particle distribution functions are obtained from the prescribed hydrodynamic conditions at the boundaries using Eq. (6.6). The original implementation was only limited to those LBM models in which the number of unknown distribution functions is equal to the number of hydrodynamic boundary conditions. This limitation was addressed later by proposing additional rules to handle LBM models in which the number of missing functions is more than the prescribed boundary conditions (Maier et al. 1996; Zou and He 1997). Chen et al. (1996) proposed an

extrapolation approach, in which the incoming distribution functions are extrapolated from the interior distribution functions, and the equilibrium distribution functions at the boundaries are computed from the prescribed hydrodynamic boundary conditions using Eq. (6.4). Guo et al. (2002) developed an extension of this method by splitting the incoming distribution functions into equilibrium and non-equilibrium parts. Equilibrium parts are computed using Eq. (6.4), while the non-equilibrium parts are extrapolated from the interior functions.

All the boundary approaches mentioned above need special care when dealing with complex geometries. For conventional CFD methods, immersed boundary methods (IBM) are now widely used to deal with curved boundaries. Since its first introduction by Peskin (1972), there has been an extensive amount of research on IBM. Among the approaches developed are forcing via deformation of elastic region tracked by Lagrangian points (Peskin 1972; Goldstein et al. 1993; Lai and Peskin 2000; Lee and LeVeque 2003), forcing via Lagrange multipliers (Glowinski et al. 1999; Taira and Colonius 2007), direct forcing by modifying discrete momentum equations (Mohd-Yusof 1997; Fadlun et al. 2000; Balaras 2004; Gilmanov and Sotiropoulos 2005; Uhlmann 2005), direct boundary implementation using Cartesian grid method (Ye et al. 1999) and direct boundary implementation using ghost cells (Majumdar et al. 2001; Tseng and Ferziger 2003). In the ghost-cell technique, hypothetical (ghost) cells are placed outside the fluid domain such that each cell has at least one neighbor inside the domain. Various options have been developed to extrapolate values to these cells to enforce boundary conditions. One widely used implementation obtains values via locating image points inside the fluid domain along the boundary normal (Majumdar et al. 2001).

IBM has been implemented into LBM as well. The original IB formulation of forcing via deformation of elastic region was coupled with LBM by Feng and Michaelides (2004). The same authors later employed direct-forcing IB formulation in LBM (Feng and Michaelides 2005). An alternative way of calculating the forcing term was developed by Niu et al. (2006) via the momentum exchange method of Ladd (1994). Several researchers have extended/improved forcing function-based IBM for various applications (Peng et al. 2006; Zhang et al. 2007; Dupuis et al. 2008; Tian et al. 2011; Kang and Hassan 2011). A drawback of this approach is that the no-slip condition is not strictly enforced at the walls. Wu and Shu (2009) proposed a velocity correction method to solve this issue. Another way of implementing strict boundary conditions is using the ghost cells-based IB method. Tiwari and Vanka (2012) first coupled this method with LBM and demonstrated its efficiency, accuracy, generality and ease of implementation. This approach and the subsequent research works toward its applications and improvements are discussed in the following sections.

6.3 Ghost Fluid LBM

Here, we describe a ghost fluid immersed boundary lattice Boltzmann method (GF-IB-LBM) developed by Tiwari and Vanka (2012), which imposes hydrodynamic boundary conditions via ghost nodes, rather than using the forcing concept. A gen-

eral approach using extrapolation along boundary normal is employed to obtain hydrodynamic values at the ghost nodes, which are then used to obtain equilibrium particle distribution functions. The non-equilibrium particle distribution functions are simply extrapolated from the fluid domain. The two contributions are then added to obtain particle distribution functions at the ghost nodes.

6.3.1 Algorithm and Implementation

For conciseness, we restrict our focus to wall (stationary as well as moving) boundary conditions in curved geometries. (The method detailed below can be extended to other types of boundary conditions in a straightforward fashion). The implementation in two dimensions (see Sect. 6.4.3 for three-dimensional extension) briefly involves the following steps.

1. Ghost node and corresponding image point identification: Before streaming operation, ghost nodes adjacent to a boundary are identified such that each ghost node has at least one neighboring node in the fluid domain. For each ghost node, an image point is located inside the fluid domain along the boundary normal. This is shown in Fig. 6.2.
2. Density and velocity determination at image points: A special bilinear interpolation procedure is developed by Tiwari and Vanka (2012) to obtain hydrodynamic values at the image points. A four-point interpolation is used when all the surrounding nodes are interior (Fig. 6.2a). If a surrounding node is not interior (Fig. 6.2b), then it is replaced by the point of intersection of the normal from that node with the boundary curve. For velocity, the values at the wall intersection points are simply the prescribed boundary values. These points are used along with the interior nodes to obtain velocities at image points. This approach to obtain bilinear coefficients can be expressed using this general formula:

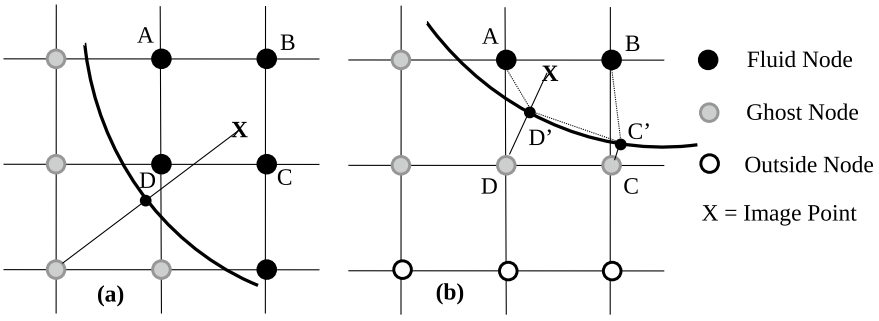


Fig. 6.2 Interpolation at image points: **a** all neighboring nodes inside, and **b** two inside and two outside nodes (Tiwari and Vanka 2012)

$$\begin{aligned} a\,x_j + b\,y_j + c\,x_j\,y_j + d &= \mathbf{u}_j \quad \text{if node } j \text{ is inside,} \\ a\,x'_j + b\,y'_j + c\,x'_j\,y'_j + d &= \mathbf{u}'_j \quad \text{otherwise,} \end{aligned}$$

where $j = 1{:}4$ denote the four surrounding nodes, a to d are the bilinear coefficients, $(x_j,\ y_j)$ and $\mathbf{u}_j$ are the spatial Cartesian coordinates and velocity, respectively, at node j and $(x'_j,\ y'_j)$ and $\mathbf{u}'_j$ are the spatial Cartesian coordinates and velocity, respectively, at the wall intersection point of node j. For density, the interpolation formula is modified such that it utilizes the zero normal gradient condition at the wall intersection points. This can also be expressed using this general formula:

$$\begin{aligned} a\,x_j + b\,y_j + c\,x_j\,y_j + d &= \rho_j \quad \text{if node } j \text{ is inside,} \\ a\,n_{xj} + b\,n_{yj} + c\,(x_j\,n_{yj} + y_j\,n_{xj}) &= 0 \quad \text{otherwise,} \end{aligned}$$

where $(n_{xj},\ n_{yj})$ denotes the boundary normal from node j, and ρ_j is the density at node j.

3. Density and velocity determination at ghost nodes: Velocity and density values at the image points are extrapolated to the ghost points along the normal direction such that the prescribed velocity and zero density gradient conditions are satisfied at the wall.
4. Particle distribution function determination at ghost nodes: The equilibrium part (f_i^{eq}) is obtained from density and velocity values at the ghost nodes using Eq. (6.4). The non-equilibrium part ($f_i^{\mathrm{neq}} = f_i - f_i^{\mathrm{eq}}$) is obtained analogous to density computation using the aforementioned special interpolation procedure. The two contributions are then summed to obtain particle distribution functions at the ghost nodes, which are then streamed inside the fluid domain during streaming operation. Note that second-order accuracy of the equilibrium part is ensured by the second-order accurate bilinear interpolation of density and velocity. However, the simple extrapolation of non-equilibrium part is only first-order accurate. Overall, second-order accuracy is attained because the non-equilibrium part corresponds to the first-order term in the asymptotic expansion of the particle distribution functions (Tiwari and Vanka 2012; demonstrated in Sects. 6.4.1 and 6.4.2).

6.3.2 Advantages

The overall approach is simple and efficient and preserves second-order accuracy for curved boundaries. A major advantage is its generality—applicable to inflow/outflow, moving wall, symmetric and periodic boundary conditions (including Dirichlet as well as Neumann conditions; Tiwari and Vanka 2012). An illustration is presented in Sect. 6.4.2. Furthermore, the method by design imposes hydrodynamics conditions strictly at the boundaries. Boundary enforcement is local, hence preserves high parallelism of LBM (discussed in Sect. 6.4.4). It also enables straightforward coupling with molecular dynamics simulations via ghost nodes, as demonstrated in Sect. 6.4.5.

6.4 Application of GF-IB-LBM

We first demonstrate the accuracy of GF-IB-LBM on four test problems involving curved and moving boundaries. Then, we discuss its coupling with molecular dynamics in a GPU-parallelized framework. For conciseness, only key results are presented here; we refer to Tiwari and Vanka (2012), Tiwari et al. (2009) and Marsh (2010) for more details.

6.4.1 One-Dimensional Problem

We first consider cylindrical Couette flow problem to (1) compare results with analytical solution and (2) demonstrate the importance of extrapolation of non-equilibrium distribution function in achieving second-order accuracy. The flow is assumed laminar, which makes this problem inherently one dimensional, which we solve using a two-dimensional lattice for demonstration (Tiwari and Vanka 2012). We consider an inner cylinder of radius r_1 rotating with an angular velocity ω and a stationary outer cylinder of radius r_2. Angular velocity is chosen such that Reynolds number based on the inner cylinder's diameter and tangential speed is 50. Figure 6.3 shows good agreement of velocity (u) variation along the radial (r) direction with the analytical solution. In Fig. 6.4a, the L_2 error norm ($\|e\|_N$) of velocity along the radial direction is plotted against grid spacing to demonstrate second-order accuracy of the method. We also plot in Fig. 6.4b, the results obtained without extrapolating the non-equilibrium part (f_i^{neq}), which shows a slope ≈ 1.4. This demonstrates the importance of extrapolation of the non-equilibrium part.

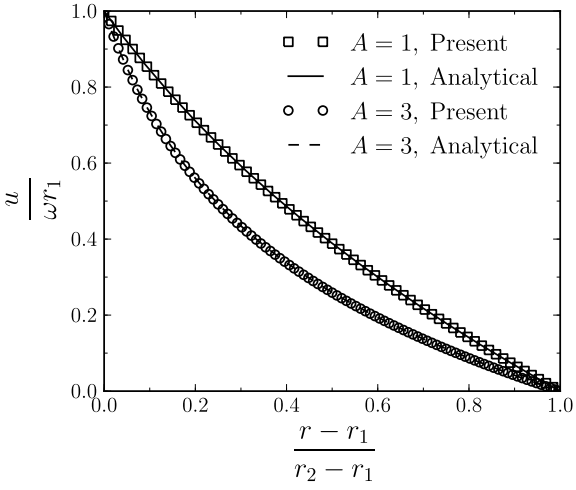
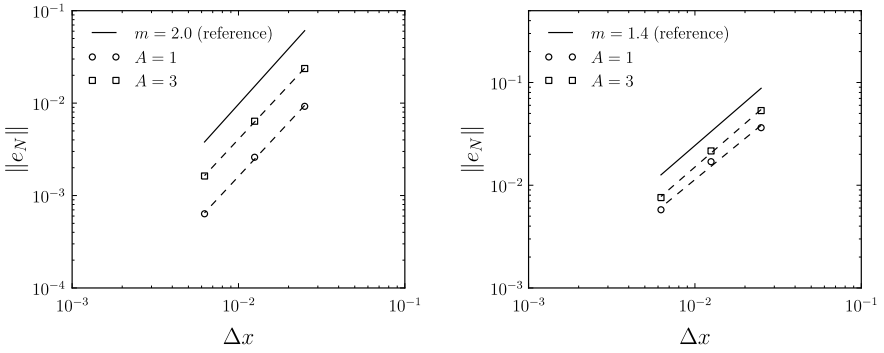


Fig. 6.3 Comparison of radial velocity profile for cylindrical Couette flow using a 321×321 grid in a $2.5r_2 \times 2.5r_2$ square domain with analytical solution for two aspect ratios ($A = (r_2 - r_1)/r_1$) (Tiwari and Vanka 2012)

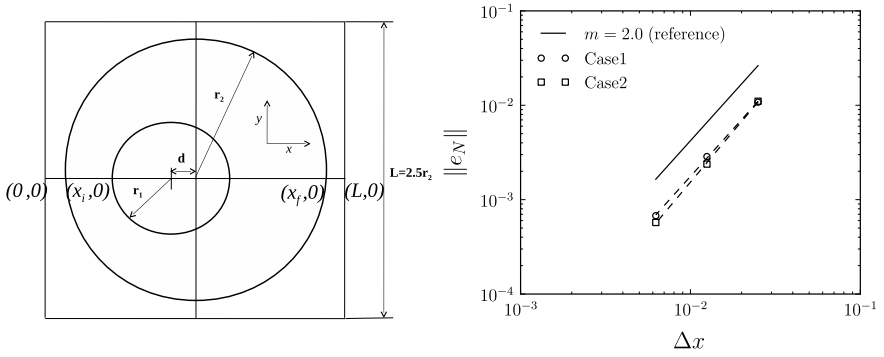


(a) with f_i^{neq} extrapolation (GF-IB-LBM) (b) without f_i^{neq} extrapolation

Fig. 6.4 Variation of L_2 error norm ($\|e\|_N$) of radial velocity with grid spacing (Δx) for cylindrical Couette flow for two aspect ratios ($A = (r_2 - r_1)/r_1$); m denotes slope (Tiwari and Vanka 2012)

6.4.2 *Two-Dimensional Problems*

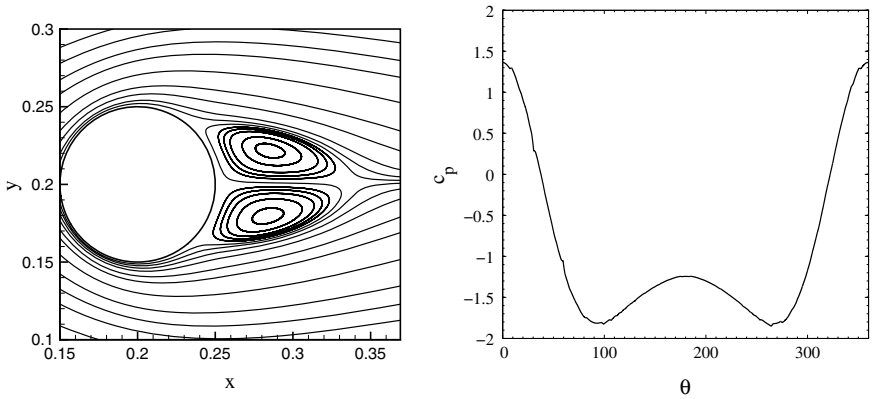
We next consider flow between two rotating eccentric cylinders (Fig. 6.5a). Due to a misalignment in their rotation axes, the flow between them is two dimensional. This configuration is included here to show second-order accuracy of GF-IB-LBM for a two-dimensional problem. Tiwari and Vanka (2012) considered four different combinations of eccentricity, radius ratio and rotational speeds. For conciseness, we present here grid convergence of two cases: (1) inner cylinder rotating and (2) outer cylinder rotating. Figure 6.5b demonstrates second-order accuracy of the method.



(a) schematic (b) error norm

Fig. 6.5 Schematic of the domain considered and variation of L_2 error norm ($\|e\|_N$) of radial velocity with grid spacing (Δx) for two cases of flow between rotating eccentric cylinders: case 1 has inner cylinder rotating and case 2 has outer cylinder rotating; m denotes slope (Tiwari and Vanka 2012)

We next consider flow over a cylinder in a channel. The previous two problems had only wall boundaries, therefore, this problem is chosen to demonstrate generality of the method for other types of boundary conditions. In this problem, parabolic velocity boundary condition is applied at the inlet, and constant pressure as well as fully developed conditions are separately considered at the outlet (Tiwari and Vanka 2012). This is a widely used verification problem; extensive benchmarking data exists (Schäfer et al. 1996). Reynolds number based on average inlet velocity and cylinder diameter is 20 for the simulated configuration. We consider three uniformly spaced lattices with $n = 16$, 32 and 64, where n denotes the number of nodes across the diameter. Figure 6.6a shows streamlines in the recirculation zone, and Fig. 6.6b shows variation of pressure coefficient c_p with cylinder angle θ using GF-IB-LBM ($n = 64$) with constant pressure boundary condition at the outlet. The drag and lift coefficients compare well with those reported by the high-resolution study of Schäfer et al. (1996) in Table 6.1.



(a) streamlines (b) pressure coefficient

Fig. 6.6 Streamlines in the recirculation zone and variation of pressure coefficient with cylinder angle for flow over a cylinder using GF-IB-LBM with $n = 64$ (Tiwari and Vanka 2012)

Table 6.1 Comparison of drag and lift coefficients (Tiwari and Vanka 2012) with Schäfer et al. (1996)

Coefficient	$n = 16$	$n = 32$	$n = 64$	Schäfer et al. (1996)
Drag	5.3203	5.4772	5.5799	5.5700–5.5900
Lift	0.0488	0.0141	0.0101	0.0104–0.0110

6.4.3 Three-Dimensional Problem

We demonstrate the extension of the approach to three dimensions in this section. Single relaxation-time D3Q27 LBM model is used here. As the name suggests, it consists of a three-dimensional lattice with particles restricted to move along 27 directions. The ghost fluid technique described in Sect. 6.3 is extended to three dimensions via tri-linear interpolation. In this case, an image point is surrounded by eight neighboring nodes; during interpolation, the outside nodes are replaced with the intersection of normal from those nodes with the boundary, analogous to the two-dimensional implementation. For demonstration, we simulate Taylor–Couette flow between cylinders (Fig. 6.7) using this approach (Tiwari et al. 2009). We consider the following configuration: radius ratio (inner radius/outer radius) is 0.5 and aspect ratio (height/inner radius) is 3.8. Rotational speed is chosen such that the Reynolds number based on the gap between the annulus and the tangential speed is 100. The generation of toroidal vortices due to flow instability at high Reynolds number is well known and widely studied (Wereley and Lueptow 1998) for the Taylor–Couette problem. Tiwari et al. (2009) simulated this configuration using GF-IB-LBM; Fig. 6.7 shows generation of vortices with a $125 \times 125 \times 95$ lattice.

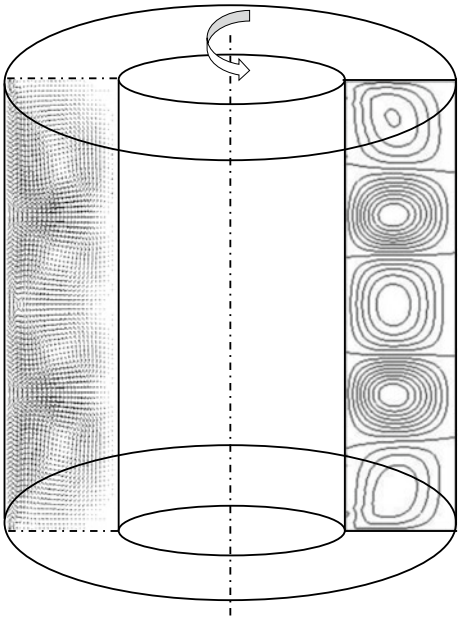


Fig. 6.7 Schematic, velocity vectors and contours of horizontal velocity component obtained by simulating Taylor–Couette problem using three-dimensional GF-IB-LBM (Tiwari et al. 2009)

6.4.4 Parallelization Using Graphical Processors

Lattice Boltzmann method is inherently highly parallelizable, hence significant speed-up can be obtained by utilizing graphical processing units (GPUs). One of the advantages of GF-IB-LBM is that it preserves the locality of the underlying LBM model—only needs information of the nodes surrounding the image points to enforce boundary conditions. Marsh (2010) developed a parallel implementation of LBM on GPUs and highlighted that parallelization of the ghost fluid technique was straightforward. They used Compute Unified Device Architecture (CUDA) (NVIDIA: https://developer.nvidia.com/cuda-zone), which is the melding of hardware and software that NVIDIA has provided to allow scientific applications to be more easily written and executed on an NVIDIA GPU. CUDA operates by executing threads on multiprocessors contained within the GPU. They reported 50–75 times speed-up on a modern GPU compared to a modern central processing unit (CPU) for their test problems using GF-IB-LBM.

6.4.5 Coupling with Molecular Dynamics

We next discuss the applicability of ghost fluid method in coupling LBM with molecular dynamics (MD) simulations. MD is an atomistic method, which has been used in computational studies for a long time, but has only recently become feasible for many applications due to its high computational cost. One of the target areas of MD simulations is nano- and micro-scale flows, where the flow behavior is not well described by the Navier–Stokes equations. MD examines physical phenomena on the atomistic scale by considering individual molecules and their interactions with each other. Extensive literature exists on interaction potentials (Allen et al. 2004); the Lennard-Jones potential is often used:

$$V_{ij}(r_{ij}) = 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right], \tag{6.7}$$

where V_{ij} and r_{ij} are the potential and distance, respectively, between molecules i and j. The two parameters ϵ and σ are used to characterize the interaction strength and length scale, respectively.

Coupling molecular dynamics with a non-atomistic CFD code such as lattice Boltzmann is advantageous when boundary wall effects are important to capture at the atomistic resolution, but the computational penalty of such high fidelity in the bulk flow is not desired. For example, consider a simple two-dimensional, planar channel flow as shown in Fig. 6.8. Marsh (2010) decomposed it into three domains for coupling: an MD domain near walls, an LBM domain for bulk flow and an overlap domain. They used Schwarz alternating method (SAM) (Dolean et al. 2015) to couple the two methods. This approach decouples both time and length scales

Fig. 6.8 Schematic of LBM-MD coupling in a planar channel (Marsh 2010)

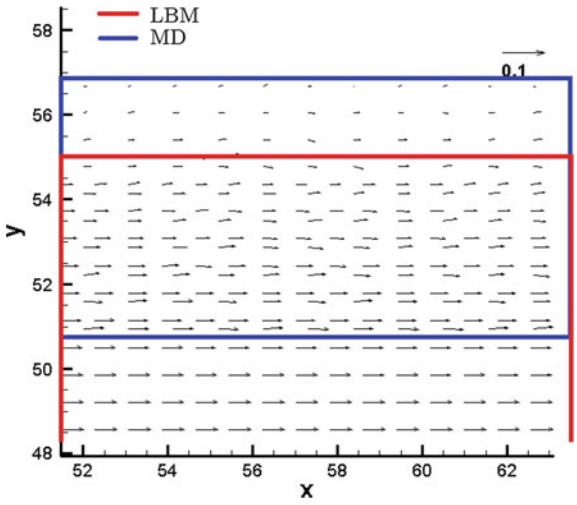


Fig. 6.9 Velocity halfway through channel flow in the overlap region (Marsh 2010)

allowing for a fully hybrid scheme. LBM and MD individually solve their respective domains including the overlap region. SAM operates by first advancing LBM by one time step (Δt). Boundary conditions are then applied to the atomistic region via the overlap region. In this step, the velocity of a molecule in the overlap region is set to the fluid velocity at the nearest lattice node. MD is then advanced via multiple (p) smaller time steps (δt) such that $\Delta t = p\,\delta t$. Boundary conditions are then applied to the bulk region. GF-IB-LBM offers straightforward enforcement of boundary conditions from MD to LBM in this step. It just requires using the averaged values from MD and imposing them on the appropriate ghost nodes. Marsh (2010) developed a parallel implementation of the coupling approach on GPUs (as discussed in Sect. 6.4.4), and simulated the aforementioned channel flow configuration. For demonstration, we show in Fig. 6.9 the velocity vectors near wall obtained from the coupled simulation; we refer to Marsh (2010) for more details.

6.5 Recent Advances

Khazaeli et al. (2013) implemented ghost fluid technique on thermal LBM. Their approach uses interpolation–extrapolation methodology based on image points similar to Tiwari and Vanka's (2012) GFM. Thermal LBM contains an additional distribution function for internal energy, whose values they obtain at ghost nodes analogous to the particle distribution function calculation via extrapolation from image points. They used an inverse distance-weighing approach for interpolation and demonstrated second-order accuracy for several problems with curvilinear boundaries.

Chen et al. (2013) investigated pressure oscillations that appear due to boundary implementation in LBM. Their investigation focused on the ghost fluid IB method (Tiwari and Vanka 2012), for which, they implemented a cut-cell-based weighting strategy to enforce geometric conservation to suppress these oscillations. They tested this method on four problems and demonstrated that the modified GFM reduces pressure oscillations while preserving its accuracy. Chen et al. (2014) compared different bounce-back and IB schemes and concluded that the unified-interpolation bounce-back of Yu et al. (2003), the direct-forcing approach of Kang and Hassan (2011) and the ghost fluid approach of Tiwari and Vanka (2012) are best suited for the acoustic problems they considered.

Kaneda et al. (2014) developed a multi-relaxation-time extension of the single relaxation-time GFM (Tiwari and Vanka 2012). They compared their results with the standard bounce-back scheme and confirmed that GFM has better accuracy, but found a defect in density calculation at image points. They attributed this to a larger predicted pressure and proposed an improvement by implementing a normal moment relation (the balance of centrifugal force and pressure) for the estimation of density distribution functions at the boundaries. Jahanshaloo et al.'s (2016) review provides an overview of several approaches developed to impose boundary conditions in LBM (including thermal LBM).

Mozafari-Shamsi et al. (2016a) developed an extension of GFM (Tiwari and Vanka 2012) for thermal LBM and implemented it for Dirichlet as well as Neumann thermal boundary conditions. As mentioned earlier, thermal LBM contains an additional equation to evolve internal energy distribution functions. Internal energy distribution functions at the ghost nodes are obtained analogous to the calculation of particle distribution functions as described in Sect. 6.3; they employed GFM's inherent feature of computing gradient of the macroscopic variables normal to the curved boundaries to formulate heat flux (Neumann) boundary conditions. In a later study, they (Mozafari-Shamsi et al. 2016b) used this GFM approach to formulate conjugate heat transfer boundary conditions at curved interfaces of two materials having different thermal properties. Boundary conditions for conjugate heat transfer are difficult to impose because heat fluxes must match in addition to imposing a common temperature value at the boundary points. Here, again, GFM's (Tiwari and Vanka 2012) normal gradient calculation makes it suitable to enforce such interface conditions. They verified the accuracy and stability of GFM computations on three test problems and confirmed its second-order accuracy.

Li et al. (2016) developed a quadratic interpolation (QGFM) variant of bilinear interpolation GFM (BGFM) (Tiwari and Vanka 2012). They compared them with Guo et al.'s (2002) extrapolation method, linear interpolation bounce-back (LIBB) method and quadratic interpolation bounce-back (QIBB) method. They found that LIBB, QIBB, Guo et al.'s scheme and BGFM are comparable in efficiency for the problems simulated, but QGFM takes about 10% more computation time due to a larger stencil construction. As expected, they observed that quadratic interpolation schemes (QGFM and QIBB) are more accurate compared to their linear counterparts (BGFM and LIBB). However, they found that the conventional bounce-back schemes are more accurate than ghost fluid interpolation schemes for the problems studied. They also compared these techniques for boundary pressure oscillations in LBM and found that oscillations are best suppressed by Guo et al.'s scheme. The authors also studied the influence of different collision models, refilling techniques and force evaluation methods in suppressing pressure oscillations.

Xu et al. (2018) recently proposed a forcing-based IB-LBM scheme for fluid–structure interaction problems. To improve numerical stability, their scheme approximates the feedback coefficient explicitly and splits Lagrangian force into traction from surrounding flow and inertial force from boundary acceleration. They also developed a dynamic geometry-adaptive grid refinement strategy, which improves the efficiency of the coupled solution by having fine resolution only near the fluid–structure interfaces.

References

Allen MP et al (2004) Introduction to molecular dynamics simulation. Computational soft matter: from synthetic polymers to proteins, vol 23, pp 1–28

Balaras E (2004) Modeling complex boundaries using an external force field on fixed Cartesian grids in large-eddy simulations. Comput Fluids 33(3):375–404

Bhatnagar PL, Gross EP, Krook M (1954) A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. Phys Rev 94 (3):511

Chen S, Doolen GD (1998) Lattice Boltzmann method for fluid flows. Annu Rev Fluid Mech 30(1):329–364

Chen H, Chen S, Matthaeus WH (1992) Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method. Phys Rev A 45(8):R5339

Chen S, Martinez D, Mei R (1996) On boundary conditions in lattice Boltzmann methods. Phys Fluids 8(9):2527–2536

Chen L, Yu Y, Hou G (2013) Sharp-interface immersed boundary lattice Boltzmann method with reduced spurious-pressure oscillations for moving boundaries. Phys Rev E 87(5):053306

Chen L, Yu Y, Lu J, Hou G (2014) A comparative study of lattice Boltzmann methods using bounce-back schemes and immersed boundary ones for flow acoustic problems. Int J Numer Methods Fluids 74(6):439–467

Dolean V, Jolivet P, Nataf F (2015) An introduction to domain decomposition methods: algorithms, theory, and parallel implementation, vol 144. SIAM

Dupuis A, Chatelain P, Koumoutsakos P (2008) An immersed boundary-lattice-Boltzmann method for the simulation of the flow past an impulsively started cylinder. J Comput Phys 227(9):4486–4498

Fadlun EA, Verzicco R, Orlandi P, Mohd-Yusof J (2000) Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161(1):35–60

Feng Z-G, Michaelides EE (2004) The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems. J Comput Phys 195(2):602–628

Feng Z-G, Michaelides EE (2005) Proteus: a direct forcing method in the simulations of particulate flows. J Comput Phys 202(1):20–51

Filippova O, Hänel D (1998) Grid refinement for lattice-BGK models. J Comput Phys 147(1):219–228

Frisch U, Hasslacher B, Pomeau Y (1986) Lattice-gas automata for the Navier-Stokes equation. Phys Rev Lett 56(14):1505

Gilmanov A, Sotiropoulos F (2005) A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. J Comput Phys 207(2):457–492

Ginzburg I, d'Humieres D (2003) Multireflection boundary conditions for lattice Boltzmann models. Phys Rev E 68(6):066614

Glowinski R, Pan T-W, Hesla TI, Joseph DD (1999) A distributed Lagrange multiplier/fictitious domain method for particulate flows. Int J Multiph Flow 25(5):755–794

Goldstein D, Handler R, Sirovich L (1993) Modeling a no-slip flow boundary with an external force field. J Comput Phys 105(2):354–366

Guo Z, Zheng C, Shi B (2002) An extrapolation method for boundary conditions in lattice Boltzmann method. Phys Fluids 14(6):2007–2010

Higuera FJ, Jiménez J (1989) Boltzmann approach to lattice gas simulations. EPL (Europhys Lett) 9(7):663

Jahanshaloo L, Sidik NAC, Fazeli A, Mahmoud Pesaran HA (2016) An overview of boundary implementation in lattice Boltzmann method for computational heat and mass transfer. Int Commun Heat Mass Transfer 78:1–12

Kaneda M, Haruna T, Suga K (2014) Ghost-fluid-based boundary treatment in lattice Boltzmann method and its extension to advancing boundary. Appl Therm Eng 72(1):126–134

Kang SK, Hassan YA (2011) A comparative study of direct-forcing immersed boundary-lattice Boltzmann methods for stationary complex boundaries. Int J Numer Methods Fluids 66(9):1132–1158

Khazaeli R, Mortazavi S, Ashrafizaadeh M (2013) Application of a ghost fluid approach for a thermal lattice Boltzmann method. J Comput Phys 250:126–140

Ladd AJC (1994) Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation. J Fluid Mech 271:285–309

Lai M-C, Peskin CS (2000) An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. J Comput Phys 160(2):705–719

Lallemand P, Luo L-S (2003) Lattice Boltzmann method for moving boundaries. J Comput Phys 184(2):406–421

Lee L, LeVeque RJ (2003) An immersed interface method for incompressible Navier-Stokes equations. SIAM J Sci Comput 25(3):832–856

Li X, Jiang F, Hu C (2016) Analysis of the accuracy and pressure oscillation of the lattice Boltzmann method for fluid-solid interactions. Comput Fluids 129:33–52

Luo L-S (2000) Theory of the lattice Boltzmann method: lattice Boltzmann models for nonideal gases. Phys Rev E 62(4):4982

Maier RS, Bernard RS, Grunau DW (1996) Boundary conditions for the lattice Boltzmann method. Phys Fluids 8(7):1788–1801

Majumdar S, Iaccarino G, Durbin P (2001) RANS solvers with adaptive structured boundary non-conforming grids. Annual research briefs, Center for Turbulence Research, pp 353–366

Marsh DD (2010) Molecular dynamics-lattice Boltzmann hybrid method on graphics processors. University of Illinois at Urbana-Champaign

McNamara GR, Zanetti G (1988) Use of the Boltzmann equation to simulate lattice-gas automata. Phys Rev Lett 61(20):2332

Mei R, Luo L-S, Shyy W (1999) An accurate curved boundary treatment in the lattice Boltzmann method. J Comput Phys 155(2):307–330
Mohd-Yusof J (1997) Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries. Annual research briefs, Center for Turbulence Research, pp 317–327
Mozafari-Shamsi M, Sefid M, Imani G (2016a) Developing a ghost fluid lattice Boltzmann method for simulation of thermal Dirichlet and Neumann conditions at curved boundaries. Numer Heat Transfer Part B: Fundam 70(3):251–266
Mozafari-Shamsi M, Sefid M, Imani G (2016b) New formulation for the simulation of the conjugate heat transfer at the curved interfaces based on the ghost fluid lattice Boltzmann method. Numer Heat Transfer Part B: Fundam 70(6):559–576
Niu XD, Shu C, Chew YT, Peng Y (2006) A momentum exchange-based immersed boundary-lattice Boltzmann method for simulating incompressible viscous flows. Phys Lett A 354(3):173–182
Noble DR, Chen S, Georgiadis JG, Buckius RO (1995) A consistent hydrodynamic boundary condition for the lattice Boltzmann method. Phys Fluids 7(1):203–209
NVIDIA. CUDA. https://developer.nvidia.com/cuda-zone
Peng Y, Shu C, Chew Y-T, Niu XD, Lu X-Y (2006) Application of multi-block approach in the immersed boundary-lattice Boltzmann method for viscous fluid flows. J Comput Phys 218(2):460–478
Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10(2):252–271
Qian Y-H, d'Humières D, Lallemand P (1992) Lattice BGK models for Navier-Stokes equation. EPL (Europhys Lett) 17(6):479
Schäfer M, Turek S, Durst F, Krause E, Rannacher R (1996) Benchmark computations of laminar flow around a cylinder. In: Flow simulation with high-performance computers II. Springer, German, pp 547–566
Shukla RK, Pantano C, Freund JB (2010) An interface capturing method for the simulation of multi-phase compressible flows. J Comput Phys 229(19):7411–7439
Taira K, Colonius T (2007) The immersed boundary method: a projection approach. J Comput Phys 225(2):2118–2137
Tian F-B, Luo H, Zhu L, Liao JC, Lu X-Y (2011) An efficient immersed boundary-lattice Boltzmann method for the hydrodynamic interaction of elastic filaments. J Comput Phys 230(19):7266–7283
Tiwari A, Vanka SP (2012) A ghost fluid Lattice Boltzmann method for complex geometries. Int J Numer Methods Fluids 69(2):481–498
Tiwari A, Freund JB, Pantano C (2013) A diffuse interface model with immiscibility preservation. J Comput Phys 252:290–309
Tiwari A, Samala R, Vanka SP (2009) Ghost fluid based immersed boundary treatment for lattice Boltzmann method. In: ASME 2009 international mechanical engineering congress and exposition. American Society of Mechanical Engineers, pp 309–313
Tseng Y-H, Ferziger JH (2003) A ghost-cell immersed boundary method for flow in complex geometry. J Comput Phys 192(2):593–623
Uhlmann M (2005) An immersed boundary method with direct forcing for the simulation of particulate flows. J Comput Phys 209(2):448–476
Wereley ST, Lueptow RM (1998) Spatio-temporal character of non-wavy and wavy Taylor-Couette flow. J Fluid Mech 364:59–80
Wu J, Shu C (2009) Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications. J Comput Phys 228(6):1963–1979
Xu L, Tian F-B, Young J, Lai JCS (2018) A novel geometry-adaptive Cartesian grid based immersed boundary-lattice Boltzmann method for fluid-structure interactions at moderate and high Reynolds numbers. J Comput Phys 375:22–56
Ye T, Mittal R, Udaykumar HS, Shyy W (1999) An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. J Comput Phys 156(2):209–240
Yu D, Mei R, Shyy W (2003) A unified boundary treatment in lattice Boltzmann method. In: 41st aerospace sciences meeting and exhibit, p 953

Zhang J, Johnson PC, Popel AS (2007) An immersed boundary lattice Boltzmann approach to simulate deformable liquid capsules and its application to microscopic blood flows. Phys Biol 4(4):285

Ziegler DP (1993) Boundary conditions for lattice Boltzmann simulations. J Stat Phys 71(5–6):1171–1177

Zou Q, He X (1997) On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. Phys Fluids 9(6):1591–1598

Part II
Compressible Flow Modeling

Chapter 7
A Levelset-Based Sharp-Interface Modified Ghost Fluid Method for High-Speed Multiphase Flows and Multi-Material Hypervelocity Impact

Pratik Das, Nirmal K. Rai, and H. S. Udaykumar

7.1 Introduction

The dynamic response of multi-material interfaces under high-speed flow conditions is important in a wide variety of engineering applications. For example, the interaction between gas–liquid interfaces and high-speed flows plays an important role in underwater explosions and droplet combustion in gas-turbine engines and rocket motors (Powell et al. 2001; Mayer and Tamura 1996). Shock interaction with gas–solid interfaces is important in shock-induced dispersal of granular material after explosions (Boiko et al. 1997), high-speed coating technologies (Dongmo et al. 2008), shock processing of powders (Thadhani 1988), shock wave lithotripsy (Jamaluddin et al. 2011), etc. The hypervelocity interaction between two solid interfaces is important in the high-speed impact penetration scenarios seen during high-velocity machining processes (Marusich and Ortiz 1995), high-velocity geological impacts (Artemieva and Shuvalov 2008), and munition–target interaction (Bürger et al. 2012). In such high-speed multi-material flow problems, severe topological change of the multi-material interfaces can occur. The interfaces may suffer extreme deformation (high-speed machining), fragmentation (droplet break-up), and collapse (shock-induced bubble or void collapse in solid or liquid); new interfaces can be created (cavitation in liquid or damage in solid material). The situation is further complicated by the interaction of high-speed nonlinear waves (e.g., shocks, tensile waves, or detonation fronts) in the material with the interfaces. The complex physics associated with the interfacial dynamics makes compressible multi-material flow problems numerically challenging. In this chapter, we describe a generic numerical framework for solving problems involving the interaction of multi-material interfaces with high-speed flows.

P. Das · N. K. Rai · H. S. Udaykumar (✉)
Department of Mechanical Engineering, The University of Iowa, Iowa City, IA, USA
e-mail: hs-kumar@uiowa.edu

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_7

Numerical methods for solving high-speed multi-material flow problems are broadly classified under two approaches: Lagrangian and Eulerian. Lagrangian approaches are popular for solving high-speed multi-material flow problems, especially in solid mechanics. The computational mesh in Lagrangian methods follows the material points. In high-strain rate problems, extreme deformation of the material may cause entanglement of the initial Lagrangian mesh and frequent re-meshing may be required, which renders the numerical solution of such problems computationally challenging. The numerical challenges of mesh entanglement with large deformation in the Lagrangian methods are ameliorated to some extent in the arbitrary Lagrangian–Eulerian (ALE) methods. In the ALE method, the mesh conforms to the contours of the deforming object, but the mesh is not attached to the material points. Nonetheless, re-meshing is still required in ALE to handle large deformation of interfaces and objects in high-speed flow problems. An alternative approach for such problems is the Eulerian method. In the Eulerian frameworks, the mesh is fixed, and the material is allowed to "flow through" the mesh. Eulerian formulations are preferred for solving problems in fluid mechanics, but the Eulerian framework can also be used to solve the high-speed problems in solid mechanics. In the Eulerian formulations for solid mechanics, spurious elastic dissipation may occur as the elastic part of strain is not fully recovered because of nonintegrability in the elasticity model. Nevertheless, under high-strain rate conditions, the elastic strains are negligible compared to the plastic strain. Furthermore, a unifying feature of broad spectrum of problems under high-speed and/or high-strain rate conditions is the hyperbolic nature of the governing equations, which can be cast under the umbrella of a general Eulerian framework.

In the fixed-grid Eulerian methods, there is no explicit definition of the interfaces between different materials or phases. The interfaces do not align with the fixed background mesh; instead, the interfaces are embedded in the fixed-grid. The interfaces are tracked implicitly either through a progress variable/field variable or Lagrangian marker point. For example, the levelset methods (Osher and Sethian 1988; Sethian and Smereka 2003; Sussman et al. 1998) uses a signed distance field on the Eulerian grid to track the evolution of the interface. Similarly, in the volume of fluid method (VOF) (Hirt and Nichols 1981), a marker function is defined as the volume of a certain phase at a given computational cell of the Eulerian grid to keep track of the interfaces. The front tracking methods (Unverdi and Tryggvason 1992) use Lagrangian marker points to trace the location of the interfaces embedded within the Eulerian grid. Among these methods, levelset-based sharp-interface tracking methods are attractive for solving high-speed multi-material flow problems. In the levelset-based approach, zero-levelset contours sharply define the interfaces embedded in the background mesh. Therefore, in the levelset-based approach, the definition of the sharp interface is readily available from the levelset field, whereas, in VOF or front tracking methods, the sharp interface is reconstructed from the volume fraction field or the Lagrangian marker points, respectively. Also, with the levelset-based approach, the extreme deformation, collision, merging, and fragmentation of the interface is naturally incorporated through the advection of the levelset

field. Therefore, levelset-based sharp-interface tracking is attractive for high-speed multi-material flows where extreme deformation of the interfaces is commonplace.

The major challenges of Eulerian sharp-interface methods lie in applying boundary conditions at the interfaces because in such methods the embedded interface does not align with the background mesh. The ghost fluid method (GFM), originally developed by Fedkiw et al. (1999), has been successfully used to prescribe appropriate boundary conditions at the embedded interfaces. In the GFM approach (Sambasivan and UdayKumar 2009; Shiv Kumar and UdayKumar 2009), a band of computational cells around the interface is defined as ghost points corresponding to each phase of the interacting media. The ghost band, when supplied with appropriate flow conditions, together with the respective real fluid, constitutes a single flow field. The success of the GFM approach largely depends on the accuracy with which the ghost states are populated. The ghost states, in turn, are derived based on the material enclosed by the embedded interfaces. Thus, in the GFM framework, the treatment of embedded interfaces essentially boils down to suitably defining the ghost states such that the material properties and the interface conditions are represented accurately. The interfacial conditions imposed at the interface through the GFM depends on the materials/phases separated by the interfaces. The numerical implementations of the interfacial conditions for different multi-material interfaces are discussed in this chapter.

In the following sections of this chapter, first, the unified governing equations for multi-material flows along with the strategies for material modeling cast in a Cartesian grid-based Eulerian framework is presented in Sects. 7.2.1 and 7.2.2. Following that the numerical methods for solving the governing equations are discussed in Sect. 7.2.3. The interfacial treatment through the levelsets and GFM are described in Sects. 7.2.4, 7.2.5, and 7.2.6. Results obtained from several different multi-material flow problems are presented in Sect. 7.3. At the end, the concluding remarks and the scopes for future work are discussed in Sect. 7.4.

7.2 Methods

A Eulerian sharp-interface multiphase framework to perform reactive mesoscale simulations involving different phases, i.e., solid, liquid, or gas under shock loading is presented. A detailed description of the governing equations, constitutive models and numerical algorithms are discussed in this section.

7.2.1 Governing Equation

The governing equations for compressible multiphase systems are solved in the following form:

$$\frac{\partial}{\partial t}(\rho Y_k) + \frac{\partial}{\partial x_j}\left(\rho u_j Y_k\right) = \frac{\partial}{\partial x_j}\left(-J_{j,k}\right) + \dot{\omega}_k \tag{7.1}$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j - \sigma_{ij}\right) = \mathrm{M}_i \tag{7.2}$$

$$\frac{\partial}{\partial t}(\rho E) + \frac{\partial}{\partial x_j}\left[u_j\left(\rho E - \sigma_{ij}\right)\right] = \frac{\partial}{\partial x_j}\left(-q_j\right) + S_E \tag{7.3}$$

where ρ, u_i, σ_{ij}, q_j, and E are the density, velocity components, Cauchy stress tensor, heat flux, and the specific total energy (kinetic and internal), respectively. The subscript k is an index for identifying species in the multicomponent system. Y_k, $J_{j,k}$, and $\dot{\omega}_k$ are the mass fraction, diffusion mass flux, and the rate of production or destruction of the mass of the kth species. The source terms M_i and S_E account for the exchange of momentum and energy between the different phases due to phase change at the sharp interface. The Cauchy stress tensor σ_{ij} is decomposed into the deviatoric τ_{ij} and dilatational part $p\delta_{ij}$ as,

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij} \tag{7.4}$$

The definition of the deviatoric τ_{ij} and dilatational part $p\delta_{ij}$ changes with the phase description. A detailed description of the constitutive models for different phases is presented next.

7.2.2 Constitutive Models

Constitutive models for Solids

For the compressible flow of deformable solid materials, the dilatational part of the stress, i.e., pressure in Eq. (7.4) is described using Mie–Gruneisen equation of state (Meyers 1994) form as:

$$p(e, V) = p_c(V) + \Gamma(V)\frac{(e - e_c(V))}{V} \tag{7.5}$$

where $V = 1/\rho$ is the specific volume, p_c is the cold curve, and e_c is the energy along the isotherm and Γ is the Gruneisen parameter. For metals which feature in the applications presented in the results section, such as nickel and aluminum, p_c is expressed as,

$$p_c(V) = \frac{\rho_0 c_0^2 \eta}{1 - s\eta^2} \tag{7.6}$$

where $\eta = 1 - \frac{V}{V_0}$, ρ_0 is the speed of sound, and s is the material parameter. The values for the material parameters used in the current analysis are provided in the

previous work (Sambasivan et al. 2013). The energy on the isotherm e_c is obtained as,

$$e_c(V) = e_0 - \int_{V_0}^{V} p_c(V)\mathrm{d}V \tag{7.7}$$

where e_0 is the reference energy at 0 K (usually set to 0).

The deviatoric response of the solid materials (exhibiting elastoplastic behavior) τ_{ij} is modeled using a hypo-elastic formulation where the rate of deviatoric stress tensor $\dot{\tau}_{ij}$ is related to the rate of change of strain rate tensor D_{ij} which is expressed in terms of velocity components as,

$$D_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{7.8}$$

The response of elastoplastic materials to high intensity (shock/impact) loading conditions are modeled by assuming the additive decomposition:

$$D_{ij} = D^e_{ij} + D^p_{ij} \tag{7.9}$$

where D^e_{ij} and D^p_{ij} are the elastic and plastic strain rate components, respectively, u_i and u_j are the velocity components. Assuming isochoric plastic flow ($\mathrm{tr}(D^p_{ij}) = 0$), the volumetric or dilatational response is governed by an equation of state (Eq. 7.5) while the deviatoric response follows the conventional theory of plasticity. Using Eq. (7.10), the rate of change of deviatoric stress component can be modeled using a hypo-elastic stress-strain relation:

$$\hat{\tau}_{ij} = 2G\left(\overline{D}_{ij} - D^p_{ij}\right) \tag{7.10}$$

where G is the modulus of rigidity, $\hat{\tau}_{ij}$ is the Jaumann derivative, and $\overline{D}_{ij}$ is the deviatoric strain rate component.

The Jaumann derivative is used to ensure the objectivity of the stress tensor with respect to rotation and expressed as,

$$\hat{\tau}_{ij} = \dot{\tau}_{ij} + \tau_{ik}\Omega_{kj} - \Omega_{ik}\tau_{kj} \tag{7.11}$$

where Ω_{ij} is the spin tensor:

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) \tag{7.12}$$

The deviatoric strain rate component in Eq. (7.10) is given by:

$$\overline{D}_{ij} = D_{ij} - \frac{1}{3} D_{kk} \delta_{ij} \tag{7.13}$$

The isochoric plastic strain rate component $\left(D_{ij}^{p} = \overline{D}_{ij}^{p}\right)$ in Eq. (7.10) is modeled assuming a coaxial flow theory (Drucker's postulate) for strain hardening material (Khan and Huang 1995):

$$D_{ij}^{p} = \Lambda N_{ij} \tag{7.14}$$

where $N_{ij} = \tau_{ij} / \sqrt{\tau_{kl} \tau_{kl}}$ is the outward normal to the yield surface and Λ is a positive scalar factor called the consistency parameter (Ponthot 2002).

To update the stress state of the elastoplastic solid materials, first, an elastic predictor step is performed by solving the stress equation,

$$\frac{\partial}{\partial t}\left(\rho \tau_{ij}\right) + \frac{\partial}{\partial x_j}\left(u_j \tau_{ij}\right) = 2G\left(\overline{D}_{ij}\right) + \Omega_{ik} \tau_{kj} - \tau_{ik} \Omega_{kj} \tag{7.15}$$

along with the conservation of mass, momentum, and energy equations (Eqs. 7.1–7.3). After the elastic update, the plastic deformation of the material is incorporated using the radial return algorithm given by Ponthot (2002), where the elastic stress is brought back to the yield surface. The Johnson–Cook model defined the yield surface as,

$$\sigma_Y = \left[A + B\left(\bar{\varepsilon}^{p}\right)^{n}\right]\left[1 + C \ln\left(\frac{\dot{\bar{\varepsilon}}^{p}}{\dot{\bar{\varepsilon}}^{p}}\right)\right]\left[1 - \theta^{m}\right] \tag{7.16}$$

where σ_Y is the yield stress, $\bar{\varepsilon}^{p}$ is the effective plastic strain, $\dot{\bar{\varepsilon}}^{p}$ is the effective plastic strain rate, A, B, C, n, m, and $\dot{\bar{\varepsilon}}_0^{p}$ are the model constants and $\theta = \frac{T - T_0}{T_{\mathrm{m}} - T_0}$ (T is the temperature, T_0 and T_{m} are the reference and melting temperature).

The details regarding the calculation of $\bar{\varepsilon}^{p}$ and $\dot{\bar{\varepsilon}}^{p}$ from the radial return algorithm is presented in the previous work (Sambasivan et al. 2013). The temperature T is calculated as,

$$T = T_0 + \frac{e - e_0}{C_V} \tag{7.17}$$

where e is the specific internal energy obtained from the energy equation (Eq. 7.3), e_0 is the reference energy, and C_V is the specific heat at constant volume for the solid material.

Constitutive models—for Liquids and Gas

For the liquid and gaseous phase, the dilatational part, i.e., pressure in Eq. (7.4) is obtained based on the different equation of state models available in the literature. Tait equation of state [ref] is used for water and liquid aluminum. Air and vapor are modeled using the ideal gas equation of state. The JWL equation of state (Massoni

Table 7.1 Initial conditions for the simulation of shock-induced combustion of an aluminum droplet

	ρ (kg/m^3)	P (Pa)	u (m/s)	T (K)
Pre-shocked air ($x \geq 8.64$ m)	1.2	101,325.0	0.0	291.5
Post-shocked air ($x < 8.64$ m)	5.13	1431,216.0	225.91	966.5
Droplet	2030.0	181,582.8	0.0	2743.0

et al. 1999) is used for the gaseous mixture formed after decomposition of energetic materials such as HMX. The expressions for the different equation of states are briefly discussed below.

Tait equation of state for liquids:
The Tait EOS in the following form is used to obtain p in the liquid phase:

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^N - 1\right] + A \tag{7.18}$$

where A, B, N, and ρ_0 are physical constants and depend on the material (Liu et al. 2005; Houim and Kuo 2013). The values of physical constants used in this work for water and liquid aluminum are shown in Table 7.1.

JWL equation of state for reaction products for HMX:
The JWL equation of state is used for the reaction products obtained from the decomposition of solid HMX:

$$p = A\left[1 - \frac{\omega V_0}{V R_1}\right]\exp(-R_1 V/V_0) + B\left[1 - \frac{\omega V_0}{V R_2}\right]\exp(-R_2 V/V_0) \tag{7.19}$$

where $V = 1/\rho$ is the specific volume, ω is the Gruneisen parameter, and A, B, R_1, and R_2 are material parameters obtained from the work of Massoni et al. (1999).

For the liquid phase, the deviatoric or the viscous stress tensor in Eq. (7.4) is given by:

$$\tau_{ij} = \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_j}{\partial x_j}\delta_{ij} \tag{7.20}$$

where μ is the viscosity of the liquid.

It is important to note that for the liquid and gas phases, the stress update equation is not solved (unlike solids) and the deviatoric stress is obtained after solving the mass, momentum, and the energy equations (Eqs. 7.1–7.3).

Constitutive models—Thermal and Species Diffusion
The heat fluxes due to thermal diffusion and species diffusion effects are obtained from:

$$q_{\mathrm{j}} = \sum_{k=1}^{N} J_{j,k} h_k - \frac{\partial (kT)}{\partial x_{\mathrm{j}}} \tag{7.21}$$

where N is the total number of species in the gaseous phase. h_k is the specific enthalpy of kth, μ and k are the mixture averaged viscosity and thermal conductivity.

The diffusion mass flux $\left(J_{j,k}\right)$ of the kth species is obtained from:

$$J_{j,k} = \rho Y_k v_{j,k} \tag{7.22}$$

where $v_{j,k}$ is the diffusion velocity of the kth species along the jth direction. The diffusion velocities are first calculated from:

$$\widehat{v_{j,k}} = -\frac{D_{k,\mathrm{mix}}}{X_k}\left(\frac{\partial X_k}{\partial x_j} + (X_k - Y_k)\frac{\partial (\ln p)}{\partial x_j}\right) \tag{7.23}$$

where X_k is the mole fraction of the kth species. The mixture averaged diffusion coefficient $D_{k,\mathrm{mix}}$ is obtained from binary diffusion coefficients D_{kl} using:

$$D_{k,\mathrm{mix}} = \frac{1 - Y_k}{\sum_{l=1,k\neq l}^{N} X_l / D_{kl}} \tag{7.24}$$

The diffusion velocities of the kth species are then corrected to ensure mass conservation (Powell et al. 2001):

$$v_{j,k} = \widehat{v_{j,k}} - \sum_{k=1}^{N} Y_k \widehat{v_{j,k}} \tag{7.25}$$

The source term in the species transport Eq. (7.1), $\dot{\omega_k}$ accounts for the vapor added to the gaseous phase at the interface and the change in species concentration:

$$\dot{\omega_k} = \dot{\omega}_{k,\mathrm{evap}} + \dot{\omega}_{k,\mathrm{react}} \tag{7.26}$$

$\dot{\omega}_{k,\mathrm{evap}}$ accounts the change in vapor mass fraction at the grid points near the droplet surface due to evaporation and $\dot{\omega}_{k,\mathrm{react}}$ accounts for the change in the mass fraction of all the species involved in the chemical reaction.

$\dot{\omega}_{k,\mathrm{evap}}$ is computed from the following equation:

$$\dot{\omega}_{k,\mathrm{evap}} = \begin{cases} 0, & \text{for } k = 1 \\ \frac{\dot{m}'' A_{\mathrm{int}}}{V}, & \text{for } k = 2 \end{cases} \tag{7.27}$$

where A_{int} is the area of the interface within a computational cell. V is the volume occupied by the gaseous phase in a cell. A_{int} and V are computed using algorithms described in Mousel (2012), Scardovelli and Zaleski (2000). $\dot{m}''$ is the evaporation

mass flux at the gas–liquid interface and is computed from the Scharge-Knudsen equation (Houim and Kuo 2013):

$$\dot{m}'' = \frac{2C}{2-C}\sqrt{\frac{\mathrm{Mw}_k}{2\pi R_u}}\left(\frac{P_{\mathrm{sat}}}{\sqrt{T_l}} - \frac{P_v}{\sqrt{T_g}}\right) \tag{7.28}$$

where

$$C = \left\{1 - \left(\frac{\rho_g}{\rho_l}\right)^{\frac{1}{3}}\right\}\exp\left(-\frac{1}{2\left(\rho_l/\rho_g\right)^{1/3} - 2}\right)$$

where R_u is the universal gas constant and Mw_k is the molecular weight of the kth species.

For problems involving shock-induced chemical reactions, the source term $\dot{\omega}_{k,\mathrm{react}}$ in Eq. (7.26) is obtained using the Arrhenius-based chemical kinetic model. For instance, energetic materials such as HMX can undergo decomposition depending on the temperature rise. To model the chemical decomposition of HMX, a three steps Arrhenius model given by Tarver et al. (1996) defines the $\dot{\omega}_{k,\mathrm{react}}$. Gas-phase combustion of aluminum vapor in the air is modeled using a 11 equation reaction model (Huang et al. 2009).

The source terms M_i represent the momentum exchange between the gas and the liquid phase due to the phase change at the interface. M_i is calculated from:

$$M_i = \frac{\dot{m}'' A_{\mathrm{int}}}{V} u_i \tag{7.29}$$

The source term in the energy equation, S_E, represents the total energy associated with the phase changes and is calculated from:

$$S_E = \sum_{k=1}^{N} \frac{\dot{m}'' A_{\mathrm{int}}}{V}\left[\left(h_{f,k} + \int_{T^o}^{T} C_{p,k}(\tau)\mathrm{d}\tau\right) - \frac{R_u}{\mathrm{Mw}_k}T\right] \tag{7.30}$$

where $h_{f,k}$ is the specific enthalpy of formation of the kth species at the reference state ($T^0 = 298$ K). $C_p^k(T)$ is the specific heat capacity at a constant pressure of the kth species, at a temperature T. $C_{p,k}(T)$ is a polynomial function of temperature T in the absolute scale and taken from (Burcat's Thermodynamic Data).

In the problems involving chemical reactions or phase changes, each of the Eulerian computation grid cells defines the mixture average pressure of the grid point, i.e.:

$$p = \sum_{k=1}^{n} p_k \tag{7.31}$$

where p_k is the partial pressure of the kth component of the gaseous mixture.

The average temperature (T) of the mixture is obtained by solving the following equation for total specific energy (E) of the system using the Newton-Raphson method:

$$E(T) = \sum_{k=1}^{n} \left[Y_k \left(h_{f,k} + \int_{T^o}^{T} C_{p,k}(\tau) \mathrm{d}\tau \right) - \frac{R_u}{\mathrm{Mw}_k} T \right] + \frac{u^2 + v^2 + w^2}{2} \tag{7.32}$$

7.2.3 Numerical Schemes

The numerical schemes to solve the system of equations described in Sects. 7.2.1 and 7.2.2 are briefly discussed in this section. Since there are different timescales involved in the governing equations for convection, thermal, and species diffusion and reaction, the numerical scheme for the governing equations is based on an operator splitting algorithm. The splitting of the operators is decided based on the relative timescales of the physical process which can be determined *a priori* using the material parameters relevant for the physical processes. For instance, the species, momentum and thermal diffusion coefficients can inform about the relative time scales. Depending on the operators, the numerical schemes can vary.

The hyperbolic terms in the governing equations are first integrated using a third-order Runge–Kutta (TVD-RK) (Gottlieb and Shu 1998) scheme to obtain an intermediate solution state $\boldsymbol{U}^*$ at the nth timestep:

$$\boldsymbol{U}^* = H^{\Delta t}\left(\mathbf{U}^n\right) \tag{7.33}$$

where $\boldsymbol{U}^n$ is the solution state at the end of the nth timestep. $H^{\Delta t}()$ is the linearized operator for integrating the hyperbolic terms in the governing equations. The parabolic terms in the governing equations are integrated using the Runge–Kutta–Chebyshev (RKC) explicit time integration scheme (Verwer et al. 2004) to obtain a second intermediate state $\boldsymbol{U}^{**}$ from $\boldsymbol{U}^*$:

$$\boldsymbol{U}^{**} = P^{\Delta t}\left(\mathbf{U}^*\right) \tag{7.34}$$

where $P^{\Delta t}()$ is the operator for integrating the parabolic terms.

Finally, the source terms are integrated using a fifth-order explicit Runge–Kutta–Fehlberg scheme to obtain the solution at the $n + 1$th timestep:

$$\boldsymbol{U}^{n+1} = S^{\Delta t}\left(\mathbf{U}^{**}\right) \tag{7.35}$$

The timestep size Δt is from the CFL number:

$$\Delta t = \mathrm{CFL}\left[\frac{\Delta x}{u+a}\right]_{\min}, \text{ where CFL} \leq 1 \text{ and a is the wave speed} \tag{7.36}$$

A third-order accurate ENO-LLF (Shu and Osher 1989) scheme is used for spatial discretization of the hyperbolic terms in the governing equations. A fourth-order accurate finite difference scheme (Das 2017) is used to discretize the parabolic terms.

7.2.4 Interface Tracking Using Levelsets

The levelset method (Osher and Sethian 1988; Sethian and Smereka 2003) is used in this work to define the interface between the gaseous and the liquid phases. The zero-levelset contour defines the location of the sharp interface between the liquid and the gaseous phases. A narrow-band levelset field provides the signed normal distance to the nominal interface from any point in a band around the sharp interface. The levelset field is advected to capture the evolution of the interface as the flow evolves in time:

$$\frac{\partial \phi}{\partial t} + \mathbf{u_n} \cdot \nabla \phi = 0 \tag{7.37}$$

where ϕ represents the levelset field. $\mathbf{u_n}$ is the normal velocity of the interface. The levelset field is advected at the end of each flow timestep to capture the evolution of the gas–liquid interface. The third-order TVD-Runge–Kutta method is used to perform the time integration. The fifth-order WENO scheme (Jiang and Shu 1996) is used for spacial discretization of Eq. (7.37). The high-order discretization scheme maintains the accuracy of the levelset advection and mitigates the mass-conservation error caused by numerical diffusion. The levelset field is reinitialized (Sussman et al. 1994) every five timesteps to ensure that it remains a signed distance function. The different materials separated by the zero-levelset contours are coupled using a modified ghost fluid method (GFM), which is described next.

7.2.5 Boundary Conditions at the Interface

The flow calculations in the different materials or phases separated by the sharp interfaces are coupled through the appropriate boundary conditions. The boundary conditions or the interfacial jump conditions depend on the materials separated by the interface. The appropriate boundary conditions for different types of interfaces are described in the following sub-sections.

Boundary treatment of solid–solid interfaces:
In problems where deformable solids interact with each other on parts of the interface, continuity of normal stress components and the continuity of normal velocity

components are enforced. No constraint is applied to the tangential components of stress and velocity fields.

The velocity field and stress states are transformed in the local normal and tangential directions at each grid points as,

$$u_n = |\vec{u}_n| = \vec{u}.\hat{n} \tag{7.38}$$

$$u_s = |\vec{u}_s| = \vec{u}.\hat{s} \tag{7.39}$$

where $\vec{u}$ is the velocity vector in the Cartesian coordinates, $\vec{u}_n$ and $\vec{u}_s$ are the normal and tangential velocity vectors.

The total stress tensor in the normal and tangential coordinates is given by

$$\widetilde{\sigma} = J\sigma J^T \tag{7.40}$$

where $J = \begin{pmatrix} n_x & n_y \\ s_x & s_y \end{pmatrix}$ is the Jacobian matrix and $\hat{n}$ and $\hat{s}$ are local normal and tangential vectors defined at the interface.

The coupling of the normal component of stress and velocity and decoupling of the tangential components ensures frictionless sliding between the materials. Thus, at the interfaces:

$$\left[\vec{u}.\hat{n}\right] = 0 \tag{7.41}$$

$$[\widetilde{\sigma}_{nn}] = 0 \tag{7.42}$$

$$[\widetilde{\sigma}_{ns}] = 0 \tag{7.43}$$

$$[P] = 0 \tag{7.44}$$

where $\widetilde{\sigma}_{nn}$ and $\widetilde{\sigma}_{ns}$ are the normal components of the stress tensor, P is pressure, and $\vec{u}$ is the velocity vector.

Boundary treatment of solid–void interfaces:
This type of interfacial condition arises whenever the deformable solid interface interacts with a surrounding void, i.e., at a free surface. Conditions corresponding to physically consistent wave reflection phenomena are enforced at all free surfaces. Therefore, zero-traction conditions on the normal stress components are enforced on those portions of the interface that are free surfaces, viz.:

$$\widetilde{\sigma}_{nn} = 0 \tag{7.45}$$

$$\widetilde{\sigma}_{ns} = 0 \tag{7.46}$$

Boundary treatment of fluid–rigid solid interfaces:
Interfaces between fluid and rigid solids are encountered in the simulations of particle-laden flows. In such calculations, the particles are assumed as rigid solids, i.e., particle deformation is neglected, and a no-slip and no-penetration boundary conditions are enforced at the fluid–rigid solid interfaces. A Dirichlet boundary condition is applied for the velocity components in the fluid. The velocity of the fluid at the interface is set to the velocity of the solid–fluid interface representing the embedded rigid object ($\boldsymbol{u}_\mathrm{I}$).

For pressure and density, Neumann boundary conditions are enforced at the solid–fluid interface. The density boundary condition is as follows:

$$\frac{\partial \rho}{\partial n} = 0 \tag{7.47}$$

A normal force balance at the interface provides the pressure boundary condition:

$$\frac{\partial p}{\partial n} = \frac{\rho_s u_{I,t}^2}{R} - \rho_s a_n \tag{7.48}$$

where $u_{I,t}$ is the magnitude of the tangential component of velocity of the fluid at the interface and a_n is the magnitude of the normal component of acceleration of the solid–fluid interface ($\boldsymbol{a}_\Gamma$).
Boundary treatment of fluid-fluid interfaces:
Fluid–fluid interfaces are encountered in problems involving bubbles or droplets suspended in a gas. The interactions of the gas and liquid at the interface are further complicated by the effects of surface tension and phase change. The interaction of the two fluids at the interface is described by the following jump conditions:

$$[u_n] = \dot{m}'' \left[\frac{1}{\rho}\right] \tag{7.49}$$

$$[p] = -\gamma\kappa - \dot{m}''[u_n] - [\tau_{nn}] \tag{7.50}$$

$$[\tau_{ns}] = -\frac{\mathrm{d}\gamma}{\mathrm{d}s} \tag{7.51}$$

$$\left[\dot{q}''_{\mathrm{cond}}\right] = -\dot{m}''[h] + [\tau_{nn} u_n] + [\tau_{ns} u_s] \tag{7.52}$$

where the operator [ ] represents:

$$[\chi] = \chi_\mathrm{g} - \chi_\mathrm{l}$$

χ is any flow variable of interest. The subscripts g and l represent the flow variables at the interface in the gaseous and the liquid phase, respectively. The subscripts n and s represent the directions normal and tangential to the interface. γ is the local

surface tension at the gas–liquid interface. κ is the local curvature at the interface and is calculated from the levelset field (Sussman et al. 1994).

Equation (7.49) accounts for the jump in the normal velocity of the two phases at the interface caused by vaporization or condensation. Equation (7.50) describes the jump in pressure at the interface due to surface tension ($-\gamma\kappa$), vaporization ($\dot{m}[u_n]$), and jump in the normal component of viscous stress (τ_{nn}). The jump in the tangential components of the deviatoric stress tensor ($[\tau_{ns}]$) in Eq. (7.51) represents the effect of Marangoni stresses at the interface. The jump in the heat flux $\left(\left[\dot{q}''_{\text{cond}}\right]\right)$ is given by Eq. (7.52). It accounts for the latent heat of evaporation ($\dot{m}[h]$) and the work done by the viscous stresses ($[\sigma_{nn}u_n]$, $[\sigma_{ns}u_s]$).

The above-mentioned jump conditions for the different types of interfaces are implemented through the ghost fluid method to couple the flow fields of different materials at the sharp interface. The implementation of the GFM for these different types of boundary conditions is described in the following section.

7.2.6 The Ghost Fluid Method

The ghost fluid method is used to supply appropriate boundary conditions at the sharp interface. The ghost fluid method was originally proposed by Fedkiw et al. (1999). In GFM, extra few layers of computational cells, defined as ghost layers, are added beyond the sharp-interface boundary for each phase. Figure 7.1 shows a schematic of a computational grid with an embedded interface to demonstrate the categorization of the computational points into the "bulk points" in the material and the "ghost points" within the ghost layer around the interface. The number of ghost layers depends on the stencil size and order of the discretization scheme. The ghost points provide the boundary conditions for the flow calculations at the computational cells of their corresponding phase near the interface. The "ghost layer" is populated such that the appropriate boundary condition at the sharp interface is imposed. There are two steps in populating the ghost points with the ghost values. In the first step, first, the flow field near the interface is reconstructed from the data available at the bulk points in the vicinity of the interface to estimate the flow variables near the interface. In the second step, the flow variables near the interface obtained from the reconstructed flow field are used to estimate the ghost value for any flow variables at the ghost point such that the boundary condition/interfacial jump condition for that variable at the interface is satisfied.

Estimation of the flow variables near the interface:
Flow variables near the interface are estimated to extend the flow field from the bulk points to the ghost points. However, in the Cartesian grid-based sharp-interface methods, the grid points often do not align with the location of the interface. Therefore, to obtain the values of the flow variables near the interface, the flow field is reconstructed along the interface normal direction.

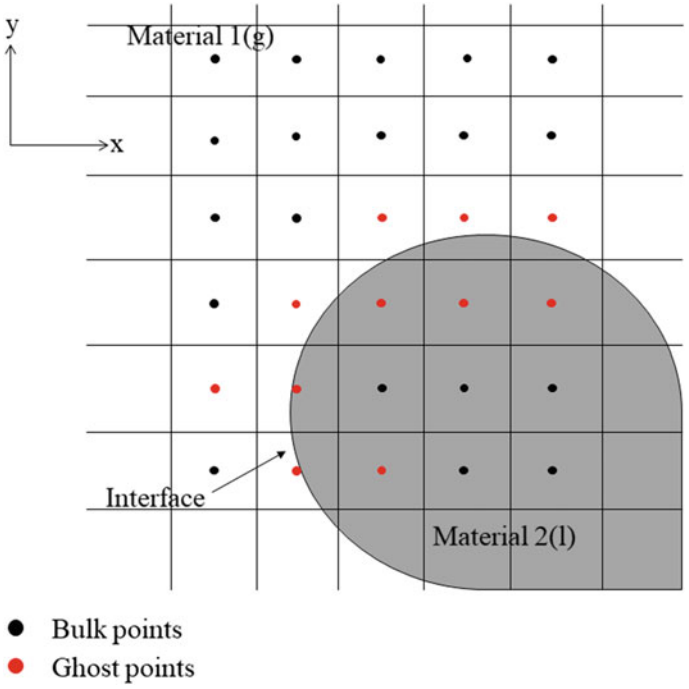


Fig. 7.1 Categorization of the computational cells

The numerical methods for obtaining the flow variables near the interface in the current levelset-based framework is explained through the following example of a typical ghost point with respect to material 1 IG in Fig. 7.2. The flow variables near the interface are obtained by probing the material 1 at a distance $1.5\Delta x$ from the interface in the gaseous phase. To probe for values of the field variables, first, the normal projection of IG on the interface, (point labeled I in Fig. 7.2) is obtained. The location of I is obtained from the following equation:

$$\mathbf{X}_I = \mathbf{X}_{IG} - \phi_{IG} * \mathbf{n}_{IG} \tag{7.53}$$

where ϕ_{IG} is the magnitude of the levelset field at the point G and $\mathbf{n}_{IG}$ is the unit vector normal to the interface computed at G from the levelset field (Sussman et al. 1998). $\boldsymbol{X}_I$ and $\mathbf{X}_{IG}$ are the locations of the points I and IG, respectively. Following this, a probe is inserted in the material 1. The probe is $1.5\Delta x$ away from the point I on the interface. The location of the probe is given by the following equations:

$$\mathbf{X}_F = \mathbf{X}_I + 1.5\Delta x * \mathbf{n}_{IG} \tag{7.54}$$

where $\mathbf{X}_F$ is the position of the endpoint of the probe F. A convex hull is formed around F using neighboring grid points in the vicinity, as shown in Fig. 7.2. The

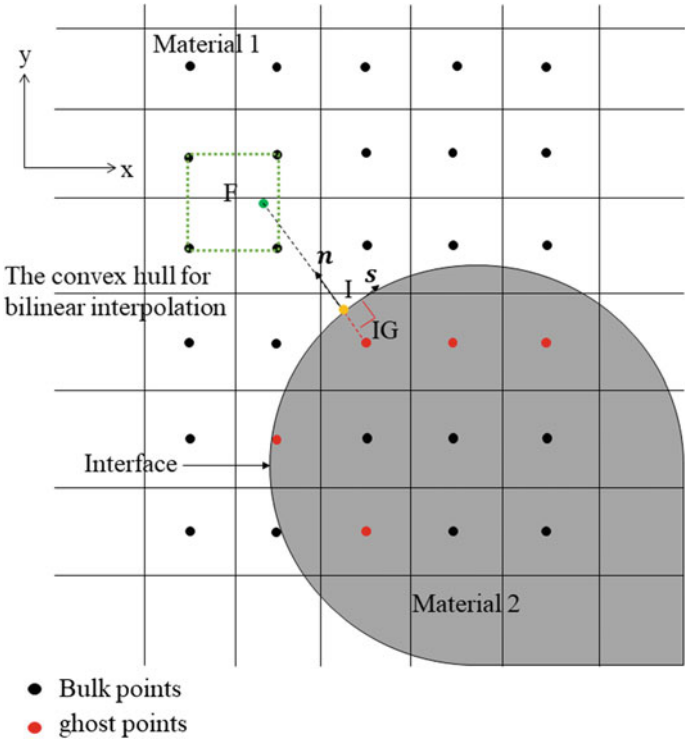


Fig. 7.2 Numerical method for reconstructing the flow field near the interface

flow variables, e.g., pressure (p_F), density (ρ_F), and velocity ($\boldsymbol{u}_F$) at F are obtained using bilinear interpolation from values at the grid points forming the convex hull. The field variables at F interpolated from the computational grid are extended to the ghost point *IG* while imposing appropriate boundary conditions at the interface.

Calculation of the ghost values from the flow variables estimated near the interface
The ghost values of the flow variables at the ghost points are calculated from the reconstructed flow field near the interface. The ghost values are computed such that the appropriate boundary conditions/interfacial jump conditions are satisfied. The boundary conditions depend on the type of interface. The numerical method for implementing the appropriate boundary conditions for solid–solid, solid–void, fluid–rigid solid, fluid–fluid, and solid–solid interfaces are described in the following sub-sections.

Solid– solid interface
The interaction between two deformable solids is modeled by populating ghost values that satisfy the continuity of normal velocity, pressure, and normal component of the stress tensor as described in Sect. 7.2.5. The ghost point for material 1 in material 2, i.e., the point *IG*, is populated with the following conditions for the field variables,

The density field is supplied using a zero gradient, i.e., Neumann condition:

$$\rho_{\mathrm{IG}} = \rho_F \tag{7.55}$$

The continuity of pressure is enforced at the ghost node *IG* in material 2 by simply injecting the node with the real value of the pressure in material 2 as,

$$p_{IG} = p_{IG}^{\mathrm{real}} \tag{7.56}$$

Similar to the pressure, the continuity of normal velocity is applied by initializing the normal component of the velocity vector with the real value (from material 2) of the normal velocity component at node IG:

$$u_{n,IG} = u_{n,IG}^{\mathrm{real}} \tag{7.57}$$

The tangential velocity component at *IG* is extended using the zero-gradient condition,

$$u_{s,IG} = u_{t,F} \tag{7.58}$$

The stress tensor at the ghost node *IG* is reconstructed by enforcing the zero-gradient condition for the tangential components and continuity of normal stress components,

$$\widetilde{\sigma}_{IG} = \begin{pmatrix} \widetilde{\sigma}_{ns}^{\mathrm{real}} & \widetilde{\sigma}_{ns}^{\mathrm{real}} \\ \widetilde{\sigma}_{ns}^{\mathrm{real}} & \widetilde{\sigma}_{ss,F} \end{pmatrix} \tag{7.59}$$

Solid–void interface:
For the solid–void interface, conditions corresponding to the physically correct wave reflections are enforced. For instance, at the free surface, a compressive wave is reflected back as a tensile wave and vice versa. Therefore, zero-traction conditions for the normal stress components are enforced. The other field variables, i.e., density and velocity components are initialized with the zero-gradient conditions.

The field variables at the ghost node, *IG* for the solid–void interface is populated as,

$$\rho_{IG} = \rho_F \tag{7.60}$$

$$u_{n,IG} = u_{n,F} \tag{7.61}$$

$$u_{s,IG} = u_{s,F} \tag{7.62}$$

$$p_{IG} = p_F \tag{7.63}$$

The zero-traction conditions, i.e., zero value of the normal stress components at the interface are applied at the ghost node *IG*,

$$\widetilde{\sigma}_{IG} = \begin{pmatrix} -\widetilde{\sigma}_{nn,F} & -\widetilde{\sigma}_{nn,F} \\ -\widetilde{\sigma}_{nn,F} & \widetilde{\sigma}_{ss,F} \end{pmatrix} \tag{7.64}$$

Fluid–rigid solid interface:
The no-slip boundary condition is applied at the fluid–rigid solid interface. The ghost values for the flow variables at a typical ghost point *IG* (Fig. 7.2) for a fluid–rigid solid interface are described in this section. A Neumann boundary condition for the pressure and density are imposed at the interface by setting

$$p_{IG} = p_F \tag{7.65}$$

and

$$\rho_{IG} = \rho_F \tag{7.66}$$

where ρ_{IG} and ρ_{IG} are the ghost values of pressure and density at the ghost point *IG*. A no-slip boundary conditions for velocity is used. For that $\boldsymbol{u}_F$ is first decomposed into the components normal and tangent to the interface, as given below:

$$\begin{pmatrix} \mathrm{u_{n,F}} \\ u_{s,F} \end{pmatrix} = \begin{bmatrix} n_x & n_y \\ n_y & -n_x \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{7.67}$$

where $u_{n,F}$ and $u_{s,F}$ are the components of $\boldsymbol{u}_F$ along the normal and the tangential direction of the interface. u and v are the components of $\boldsymbol{u}_F$ along the x- and y-axis, respectively. n_x and n_y are the x and y components of $\boldsymbol{n_{IG}}$, respectively. The ghost values of the velocity components at *IG* $\left(u_{n,IG}, u_{s,IG}\right)$ are calculated from the velocity of the gaseous phase and the velocity of the solid–gas interface using linear interpolation, as follows:

$$u_{n,IG} = \frac{u_{n,I}(\phi_{IG} + 1.5\Delta x) - \phi_{IG} u_{n,F}}{1.5\Delta x} \tag{7.68}$$

$$u_{s,IG} = \frac{u_{s,I}(\phi_{IG} + 1.5\Delta x) - \phi_{IG} u_{s,F}}{1.5\Delta x} \tag{7.69}$$

where $u_{n,I}$ and $u_{s,I}$ are the components of velocity of the interface along the normal and the tangential direction of the interface.

GFM treatment of the fluid–fluid interface:
Ghost values for the fluid–fluid interfaces are obtained such that Eqs. (7.49)–(7.52) are satisfied. However, only satisfying Eqs. (7.49)–(7.52) while computing the ghost values is not sufficient to ensure the coupling of the two phases at the interfaces. Equations (7.49)–(7.52) are coupled with a local 1D Riemann problem at the interface to

allow the characteristics waves in the bulk material to travel across the interfaces. However, solving Eqs. (7.49), (7.50), and (7.51) simultaneously in conjunction with an interfacial Riemann problem is computationally expensive. To avoid this problem, the interfacial jump conditions are decoupled and solved separately during the hyperbolic step and the parabolic step within an overall single flow timestep. The following 1D local Riemann problem along with the following interfacial jump conditions are solved to populate the ghost points before integrating the hyperbolic terms in the governing equations:

$$[u_n] = \dot{m}''\left[\frac{1}{\rho}\right] \tag{7.70}$$

$$[p] = -\gamma\kappa - \dot{m}''[u_n] \tag{7.71}$$

The contributions from the parabolic terms for the two phases at the interface are coupled by populating the ghost points such that the following interfacial jump conditions are imposed at the interface:

$$[\tau_{nn}] = 0 \tag{7.72}$$

$$[\tau_{ns}] = -\frac{\mathrm{d}\gamma}{\mathrm{d}s} \tag{7.73}$$

$$\left[\dot{q}''_{\mathrm{cond}}\right] = -\dot{m}''[h] + [\tau_{nn}u_n] + [\tau_{ns}u_s] \tag{7.74}$$

The methods adopted to obtain the ghost values for the hyperbolic and the parabolic steps are described in the following two sub-sections.

Treatment of interface for hyperbolic terms
An interfacial Riemann problem is solved to obtain p, ρ and u_n at the interfacial ghost points. Figure 7.3 shows a schematic to illustrate the numerical method for constructing a local 1D Riemann problem at a typical interfacial ghost point labeled *IG*. A local Riemann problem normal to the interface is constructed at the ghost point *G*. The initial conditions for the local Riemann problem are obtained from the flow variables (ρ, u_n, p) in the gaseous $\left(\rho_g, u_{n,g}, p_g\right)$ and the liquid $\left(\rho_l, u_{n,l}, p_l\right)$ phases near the interface.

A 1D Riemann problem is solved to obtain the intermediate (*) states from the flow variables in the gaseous phase $\left(\rho_g, u_{n,g}, p_g\right)$ and the liquid phase $\left(\rho_l, u_{n,l}, p_l\right)$ at the interface. The intermediate (*) states are used as the ghost values at the ghost points for each phase. In this formulation, the jumps in pressure and normal velocity of the intermediate (*) states across the contact discontinuity given by the following equations:

$$\left[u_n^*\right] = u_{n,g}^* - u_{n,l}^* = \dot{m}''\left[\frac{1}{\rho}\right] \tag{7.75}$$

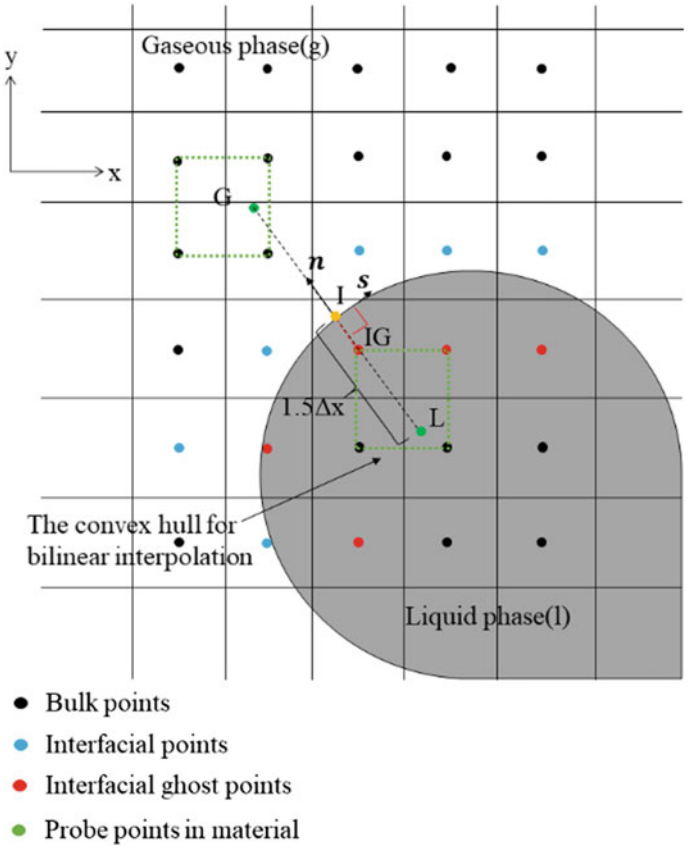


Fig. 7.3 Schematic for computing the ghost values during the hyperbolic step of the multi-step integration process

$$\left[p^*\right] = p_g^* - p_l^* = -\gamma\kappa - \dot{m}''[u_n] \tag{7.76}$$

are incorporated in the 1D Riemann problem. The Riemann problem takes the following algebraic form:

$$f_l\left(p_l^*, p_l, \rho_l, u_{n,l}\right) + f_g\left(p_g^*, p_g, \rho_g, u_{n,g}\right) + u_{n,g} - u_{n,l} + [u_n] = 0 \tag{7.77}$$

where

$$f_g = \begin{cases} \left(p_g^* - p_g\right)\sqrt{\frac{A_g}{p_g^* - B_g}}, & \text{when } p_g^* > p_g \\ \frac{2a_g}{\gamma - 1}\left[\left(\frac{p_g^*}{p_g}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right], & \text{when } p_g^* < p_g \end{cases}$$

$$A_g = \frac{2}{(\gamma+1)\rho_g}$$

$$B_g = \frac{\gamma-1}{\gamma+1}\left(P_g + B\right)$$

$$a_g = \sqrt{\frac{\gamma P_g}{\rho_g}}$$

and

$$f_l = \begin{cases} \left(p_l^* - p_l\right)\sqrt{\frac{A_l}{\left(p_l^*+\overline{B}\right)-B_l}}, & \text{when } p_l^* > p_l \\ \frac{2a_l}{N-1}\left[\left(\frac{p_l^*+\overline{B}}{p_l+\overline{B}}\right)^{\frac{N-1}{2N}} - 1\right], & \text{when } p_l^* < p_l \end{cases}$$

$$A_l = \frac{2}{(N+1)\rho_l}$$

$$B_l = \frac{N-1}{N+1}\left(p_l + \overline{B}\right)$$

$$a_l = \sqrt{\frac{N * \left(p_l + \overline{B}\right)}{\rho_l}}$$

$$\overline{B} = B - A$$

The Newton-Raphson method is used to solve Eq. (7.77). ρ_g^*, ρ_l^*, $u_{n,g}^*$, and $u_{n,l}^*$ are obtained from the following equations:

$$\rho_g^* = \begin{cases} \rho_g\sqrt{\frac{\frac{p_g^*}{p_g}+\frac{\gamma-1}{\gamma+1}}{\frac{\gamma-1}{\gamma+1}\frac{p_g^*}{p_g}+1}}, & \text{when } p_g^* > p_g \\ \rho_g\left(\frac{p_g^*}{p_g}\right)^{\frac{1}{\gamma}}, & \text{when } p_g^* < p_g \end{cases} \tag{7.78}$$

$$\rho_l^* = \begin{cases} \rho_l\sqrt{\frac{\frac{p_l^*+\overline{B}}{p_l+\overline{B}}+\frac{N-1}{N+1}}{\frac{\gamma-1}{\gamma+1}\frac{p_l^*+\overline{B}}{p_l+\overline{B}}+1}}, & \text{when } p_l^* > p_l \\ \rho_l\left(\frac{p_l^*+\overline{B}}{p_l+\overline{B}}\right)^{\frac{1}{N}}, & \text{when } p_l^* < p_l \end{cases} \tag{7.79}$$

$$u_g^* = \frac{u_g + u_l}{2} + \frac{f_g - f_l}{2} + \frac{\dot{m}''\left[\frac{1}{\rho}\right]}{2} \tag{7.80}$$

$$u_l^* = \frac{u_g + u_l}{2} + \frac{f_g - f_l}{2} - \frac{\dot{m}''\left[\frac{1}{\rho}\right]}{2} \tag{7.81}$$

ρ_g^*, $u_{n,g}^*$, and p_g^* are obtained by solving the 1D interfacial Riemann problem at I and are used as the ghost values for the gaseous phase. Similarly, ρ_l^*, $u_{n,l}^*$, and p_l^*, computed from the 1D interfacial Riemann problem are used as ghost values at the interfacial ghost points with respect to the fluid phase. These ghost values for density, velocity, and pressure at the interfacial ghost points are extrapolated to the interior ghost points using a PDE-based multidimensional extrapolation approach (Meyers 1994).

The GFM for the parabolic terms

The GFM treatment at the interface for the parabolic terms in the governing equations is different from the hyperbolic terms. The ghost values for velocity and temperature are calculated separately before integrating the parabolic terms in the governing equation such that Eqs. (7.72)–(7.74) are satisfied.

Calculation of the velocity field in the ghost fluid region

The numerical method for computing the ghost values of velocity components for coupling the parabolic terms at the interface is described in this section in the context of a typical ghost point *IG* in Fig. 7.3. The ghost values of the velocity at *IG* are obtained by solving Eqs. (7.72) and (7.73). which can be written in the following forms:

$$[\tau_{nn}] = \left[2\mu\frac{\partial u_n}{\partial n} - \frac{2}{3}\mu\left(\frac{\partial u_n}{\partial n} + \frac{\partial u_s}{\partial s}\right)\right] = 0 \tag{7.82}$$

$$[\tau_{ns}] = \left[\mu\left(\frac{\partial u_s}{\partial n} + \frac{\partial u_n}{\partial s}\right)\right] = -\frac{\mathrm{d}\gamma}{\mathrm{d}s} \tag{7.83}$$

where u_n and u_s are the components of velocities of the fluid phases along the normal and the tangential direction of the interface calculated using Eq. (7.67). The derivatives of u_n and u_s in Eqs. (7.82) and (7.83) can be approximated from the reconstructed velocity field of the corresponding phases around the interface. Now, the velocity of the two phases at the interface is not readily available because the Cartesian grid does not align with the interface. The velocity of the fluids at the interface can be obtained solving Eqs. (7.82) and (7.83) along with the jump conditions for the velocity field at the interface given by the following equations:

$$u_{n,I,g} - u_{n,I,l} = \left[u_{n,I}\right] = \dot{m}''\left[\frac{1}{\rho}\right] \tag{7.84}$$

$$u_{s,I,g} - u_{s,I,l} = \left[u_{s,I}\right] \tag{7.85}$$

where $u_{n,I}$ and $u_{s,I}$ are the components $\boldsymbol{u}_I$ along the normal and the tangential direction of the interface.$u_{n,I,g}$ and $u_{s,I,g}$ are the velocity components of the gaseous phase along $\boldsymbol{n}$ and $\boldsymbol{s}$ at the point I in Fig. 7.3. $u_{n,I,l}$ and $u_{s,I,l}$ are the velocity components of the liquid phase along $\boldsymbol{n}$ and $\boldsymbol{s}$ at the point I.

The ghost values of the velocity at IG $\left(\boldsymbol{u}_{IG}|_{\text{ghost}}\right)$ are extrapolated from the velocity of the gaseous phase at the interface $\left(\boldsymbol{u}_{I,g}\right)$ and the point G $(\boldsymbol{u}_G)$, so that:

$$\boldsymbol{u}_{IG}|_{\text{ghost}} = \frac{\boldsymbol{u}_{I,g}(\phi_{IG} + 1.5\Delta x) - \phi_{IG}\boldsymbol{u}_G}{1.5\Delta x} \tag{7.86}$$

Similarly, the velocity at the ghost points with respect to the liquid phase can also be calculated by solving Eqs. (7.82)–(7.85).

Calculation of the temperature field in the ghost fluid region:
The ghost value for the temperature at *IG* is calculated such that the jump condition in the heat flux given by Eq. (7.74) is satisfied. The jump in heat flux between the gaseous and the liquid phase is cast in the following form:

$$-k_g \left.\frac{\partial T}{\partial n}\right|_g + k_l \left.\frac{\partial T}{\partial n}\right|_l = \left[\dot{q}''_{\text{cond}}\right] \tag{7.87}$$

where k_g and k_l are the thermal conductivity of the gas and the liquid, respectively, at the interface. $\left(\frac{\partial T}{\partial n}\right)_g$ and $\left(\frac{\partial T}{\partial n}\right)_l$ are the thermal gradients in the gaseous and the liquid phase at the interface, in the direction normal to the interface. For a typical ghost point *IG*, shown in Fig. 7.3a, $\left(\frac{\partial T}{\partial n}\right)_g$ and $\left(\frac{\partial T}{\partial n}\right)_l$ are estimated from the following relations:

$$\left.\frac{\partial T}{\partial n}\right|_g = \frac{T_G - T_{I,g}}{1.5\Delta x} \tag{7.88}$$

$$\left.\frac{\partial T}{\partial n}\right|_l = -\frac{T_L - T_{I,l}}{1.5\Delta x} \tag{7.89}$$

where T_G and T_L are the temperature at the points *G* and *L* in Fig. 7.3a. Similar to the velocity components, T_G and T_L are estimated from the temperature at the nearest four grid points using bilinear interpolation. $T_{I,g}$ and $T_{I,l}$ are the temperature of the gaseous and liquid phases, respectively, at the interface. The jump in temperature at the interface is given by:

$$T_{I,g} - T_{I,l} = [\mathrm{T}_I] \tag{7.90}$$

In this work, the temperature is assumed to be continuous at the interface. Therefore, $[\mathrm{T}_I] = 0$.

Equations (7.88) and (7.89) are substituted in Eq. (7.87) to obtain:

$$-k_g \frac{T_G - T_{I,g}}{1.5\Delta x} - k_l \frac{T_L - T_{I,l}}{1.5\Delta x} = \left[\dot{q}''_{\text{cond}}\right] \tag{7.91}$$

Equations (7.90) and (7.91) are solved to obtain $T_{I,g}$ and $T_{I,l}$ as given below:

$$T_{I,g} = \frac{k_g T_G + k_l T_L + k_l [T_I] + 1.5\Delta x\left[\dot{q}''_{\text{cond}}\right]}{k_g + k_l} \tag{7.92}$$

$$T_{I,l} = \frac{k_g T_G + k_l T_L - k_g [T_I] + 1.5\Delta x \left[\dot{q}''_{\text{cond}}\right]}{k_g + k_l} \tag{7.93}$$

Once $T_{I,g}$ and $T_{I,l}$ are obtained, the ghost value of the temperature at IG, $T_G|_{\text{ghost}}$, is obtained by linear extrapolation:

$$T_{IG}|_{\text{ghost}} = \frac{T_{I,g}(\phi_{IG} + 1.5\Delta x) - \phi_{IG} T_G}{1.5\Delta x} \tag{7.94}$$

Further detail of the current GFM can be found in Das and UdayKumar (2019).

7.3 Results and Discussion

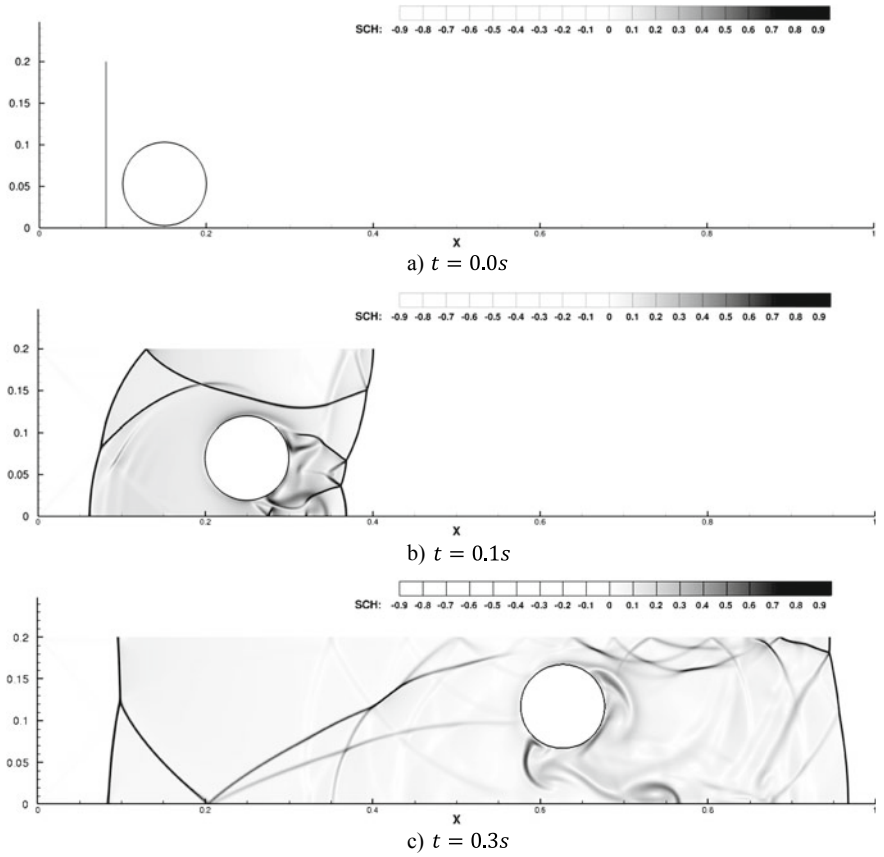
The above numerical framework has been validated against several benchmark experimental and numerical studies in the previous work (Sambasivan and UdayKumar 2009; Shiv Kumar and UdayKumar 2009; Das 2017; Das et al. 2018a, b). In this section, we demonstrate the extent of the capabilities of the current sharp-interface methods in solving high-speed multi-material flow problems through the following numerical example involving fluid–solid, fluid–fluid, solid–solid, and solid–void interfaces.

7.3.1 Fluid–Rigid Solid Interface: Shock-Induced Lift-off of a Rigid Cylinder in a Shock Tube

A numerical study of shock-induced lift-off of a rigid cylinder is performed using the current method (Das 2017). The trajectory of the center of the cylinder calculated from the current calculations is compared with a benchmark result. The length of the computational domain is selected as the reference length scale and is taken to be 1.0. The height of the domain is 0.2 and the diameter of the cylinder is 0.1 non-dimensional units. The cylinder center is initially at (0.15, 0.05), i.e., the cylinder is placed close to the bottom wall of the shock tube. The non-dimensional values of the pressure and density of the un-shocked fluid are 1.4 and 1, respectively. The non-dimensionalized density of the cylinder is 10.77. The Reynolds number calculated based on the flow conditions behind the traveling shock wave is 240. A shock wave of Ma = 3.0 is located initially at $x = 0.08$ and is allowed to evolve until time $t = 0.3$ s. A reflective boundary condition is applied at the top and the bottom edges of the computation domain. Neumann boundary condition is applied at the east and the west edges of the computation domain. In this study, a uniform Cartesian grid is used. Five different grid resolutions are considered, corresponding to 50, 100, 150, 200, and 400 points across diameter for the grid convergence study. The numerical

Schlieren plots presented in Fig. 7.4 are obtained from the mesh with a grid resolution of 200 points across the diameter of the cylinder.

The numerical Schlieren fields computed at different time instances ($t = 0.0$, 0.1, 0.3 s) are shown in Fig. 7.4. As the flow evolves, the incident shock interacts with the cylinder and reflects from the cylinder surface. The reflected shock travels outward from the cylinder surface and interacts with the bottom wall. The shock wave reflected from the bottom wall of the computational domain interacts with the cylinder again, producing a non-zero lift on the cylinder. The non-zero lift causes the cylinder to move up from the bottom edge of the computational domain. The locus of the center of the moving cylinder is compared with the benchmark results (Shiv Kumar and UdayKumar 2009; Meyers 1994) in Fig. 7.5. The trajectory of the cylinder center obtained from the current study is in good agreement with the results of previous studies. It is also observed that the lift-off height of the cylinder is somewhat lower in the current viscous flow simulation, i.e., viscous effects suppress the lift-off of the



a) $t = 0.0s$

b) $t = 0.1s$

c) $t = 0.3s$

Fig. 7.4 Shock-induced ($M_s = 3.0$) lift-off of a rigid cylinder in shock tube

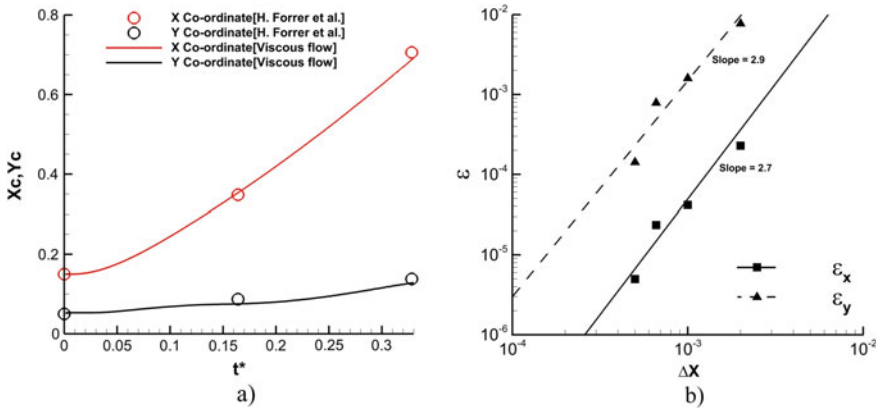


Fig. 7.5 **a** Locus of the center of the rigid cylinder during the shock-induced lift-off. **b** The decreasing L_2 error in the locus of the cylinder with grid refinement

cylinder. This effect is modest in the present case since the length over which the lift-off occurs is small. Thus, although the flow features differ noticeably between the viscous and inviscid cases, the differences in the particle motion are not significant for the current cylinder lift-off problem, at least for the duration of the simulation (Fig. 7.5).

A convergence study is performed for the above moving boundary problem; the convergence evaluation is based on the errors in tracking the locus of the cylinder center. The L_2 error in the locus of the cylinder is computed from:

$$\varepsilon = \sqrt{\frac{\int_0^T \left(x_{c_i}^{\text{fine grid}} - x_{c_i}^{\text{Coarse grid}} \right)^2 \mathrm{d}t}{\int_0^T \left(x_{c_i}^{\text{fine grid}} \right)^2 \mathrm{d}t}} \tag{7.95}$$

The error is seen to monotonically decrease with grid refinement in Fig. 7.5a.

Results obtained from the simulation of cylinder lift-off caused by shock impingement in a shock tube show that the current GFM is adequate for viscous simulations of moving boundary problems in supersonic flow.

7.3.2 *Fluid–Rigid Solid Interactions: Mach 5 Shock Interaction with a Cluster of Particles*

A resolved simulation of shock interaction with a cluster of randomly arranged cylindrical particles is demonstrated. In this simulation, the shock Mach number (M_s) is 5. A cluster of 62 randomly arranged aluminum particles of uniform diameter is used in this simulation. The volume fraction (ϕ) of the particles in the cluster is

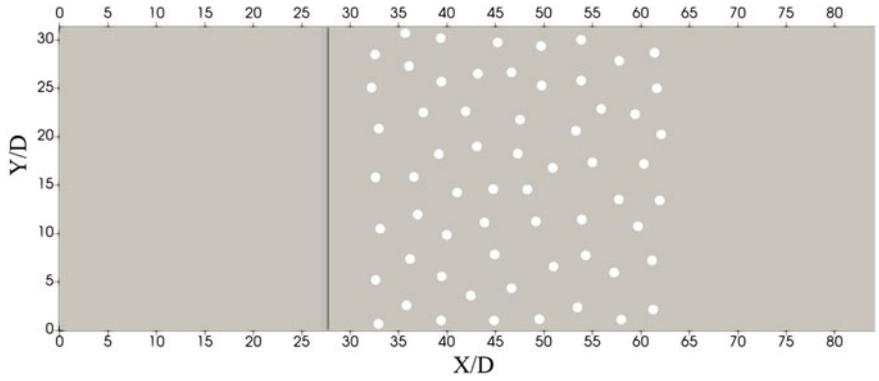
5%. Reynolds number of the post-shock incoming flow with respect to each particle (Re_D) is 1000.

A uniform Cartesian grid used in the current calculation. The solid–fluid interfaces of the particles are tracked sharply using the current levelset-based approach. No-slip boundary condition is applied at the interface. The particles in this calculation are resolved using 100 mesh points across the diameter to capture the viscous boundary layer on the particles. The diameter of the particles (D) is selected as the characteristic length scale in this calculation. The initial configuration of the particle cloud immersed in a quiescent fluid is shown in Fig. 7.6a. Outflow boundary conditions are applied on all sides of the computational domain.

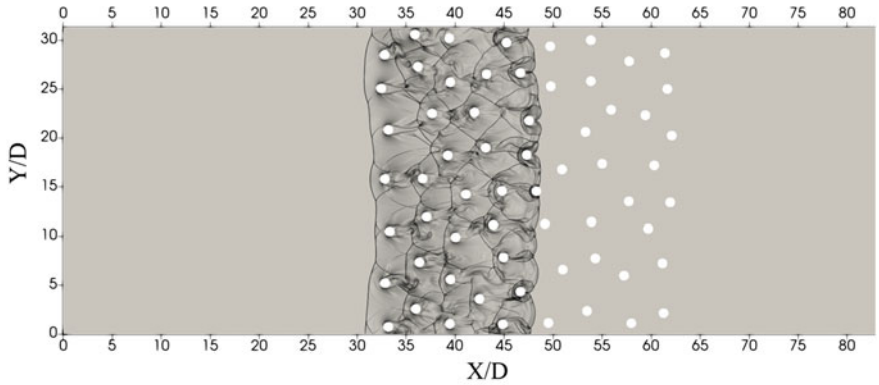
The sequence of numerical Schlieren in Fig. 7.6a–c shows the evolution of the unsteady flow field and the intricate shock structures during the interaction of incoming shock waves with the particles. As the incident shock interacts with the particle cluster, the reflections of the incident shock from the front row of the particles coalesce to form an effectively planar reflected shock, as seen in Fig. 7.6b, c. A part of the incident shock is also transmitted through the cluster of particles. The reflected and transmitted shock waves are seen in Fig. 7.6c. The transmitted shock loses its strength as it travels through the cluster of particles. The attenuation of the strength of the transmitted shock wave has been observed previously by Chaudhury et al. (2013).

As the transmitted shock wave propagates through the cluster, multiple internal reflections of the shocks lead to an unsteady flow field. The vorticity contour plot of the shocked flow field at $t * \frac{U_s}{D} = 57.0$ in Fig. 7.7 exhibits this unsteadiness. Baroclinic vortices generated in the slip lines combine with wake vortices caused by separated shear layers form the coherent structures observed in the vorticity contour plot. Vorticity concentrations are also seen in inviscid flow calculations of shocks traversing particle clusters (Das 2017; Regele et al. 2014). In the present case, viscous effects augment the inviscid vorticity generation mechanisms leading to increased magnitudes of vorticity in the cluster (Das 2017).

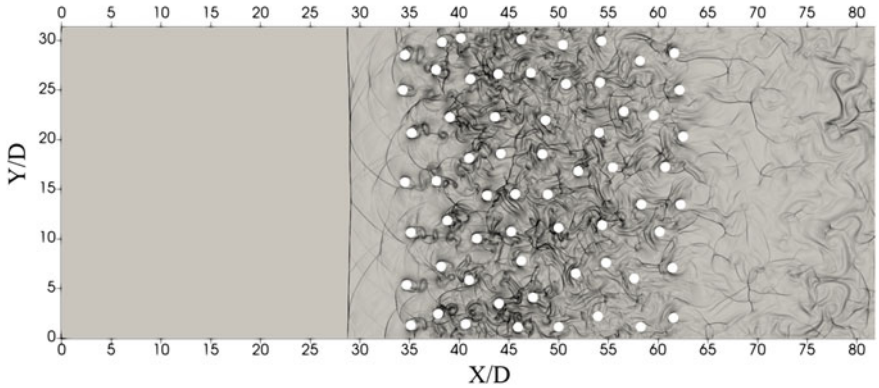
Significant movement of the particles during the interaction with the incoming shock wave is not observed in the current simulation. This is because the timescale of the current simulation is significantly smaller than the timescale of the movement of the particles (Das 2017; Mehta et al. 2016). However, it is worth mentioning that the particle cluster gets compressed inhomogeneously during the interaction with the shock. The sequence of numerical Schlieren in Fig. 7.7 shows that the displacement of the particles in the downstream part of the cloud is less than the particles in the front of the cloud. As the shock passes over the cloud, the particles located at the leading edge begin to equilibrate with the flow even before the shock has reached the downstream end of the cloud. Owing to this, the particles at the front-end start moving before the shock reaches the trailing end of the cloud. This leads to enhanced clustering at the leading edge of the cloud, i.e., the local volume fraction of the particles at the front-end of the cloud becomes higher relative to the rear-end. Therefore, even within the short period of the shock passage, movement of the particles changes the local solid volume fraction in the cloud.



a) $t * U_s/D = 0.0$



b) $t * \frac{U_s}{D} \sim 20.0$



c) $t = 57.0$

Fig. 7.6 Mach 5 shock interaction with a cluster of 62 rigid cylindrical particles

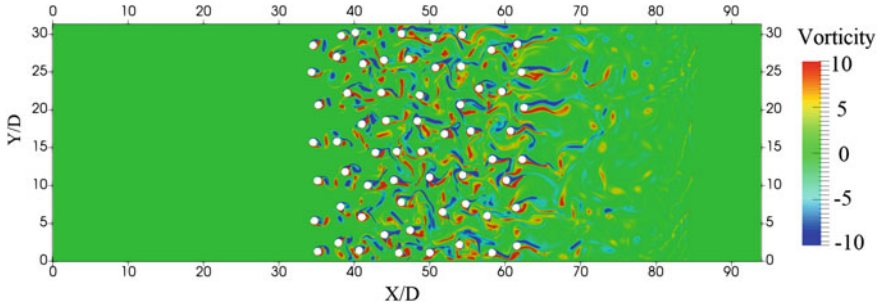


Fig. 7.7 Vorticity contours during Mach 5 shock interaction with a cluster of particles of 10% volume fraction

The results obtained from the current simulation show that the levelset-based approach in conjunction with the current GFM can capture the intricate features of the flow fields, such as the viscous boundary layer over the particles and the movement of the particles due to shock interaction. The current results and the previous studies (Khan and Huang 1995; Massoni et al. 1999; Houim and Kuo 2013) have shown that the current levelset-based approach is suitable for studying shock interaction with particle clouds through resolved simulations.

7.3.3 Gas–Liquid Interfaces: Mach 3.5 Shock Interaction with an Aluminum Droplet

The shock-induced combustion of a cylindrical droplet is studied using the current levelset-based sharp-interface method. For this study, the interaction of a Mach 3.5 shock wave with a cylindrical aluminum droplet of 4 μm in diameter is simulated. The initial computational setup for the simulation is shown in Fig. 7.8. Reflective

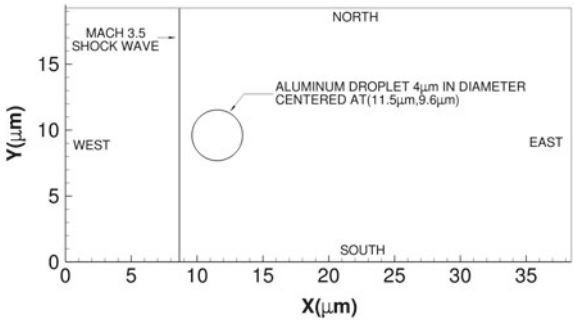


Fig. 7.8 Initial condition for 2D simulation of Mach 3.5 shock interaction with an aluminum droplet of 4 micron in diameter

boundary conditions are used at the north and the south boundaries of the computational domain. An outflow boundary condition is used at the east and the west boundaries. The initial conditions for the simulation are as in Table 7.1.

The vaporization of the aluminum droplet at the surface and the combustion of the evaporated aluminum in the air are modeled in this calculation. An 11-step reduced-order reaction model for aluminum combustion in the air is used for this current calculation (Huang et al. 2009). The material properties for the gas and the liquid phases such as viscosity, thermal conductivity, and surface tension used in the current simulation can be found in Houim (2011). With the given values for the material properties and the flow conditions, the non-dimensional numbers are $\mathrm{Re}_D = 1000$ and $\mathrm{We}_D = 31.56$. The grid resolution of 200 mesh points across the diameter is used for this study.

The interaction of Mach 3.5 shock with the aluminum droplet is shown through multiple snapshots of numerical Schlieren and pressure contours in Fig. 7.9. The initial interaction of the incident shock with the droplet is demonstrated in Fig. 7.9a–c. The high temperature and the impulsive vaporization of the molten aluminum droplet initiate a shock wave at the droplet surface as seen in Fig. 7.9a. The interaction of the incident shock with the gas–liquid interface produces a reflected shock in the gas and a transmitted shock in the liquid. The reflected and transmitted waves are seen in Fig. 7.9a. The transmitted wave travels faster through the liquid than the incident wave in the air as the speed of sound is higher in the liquid than in air. The transmitted shock wave travels further through the droplet and reaches the gas–liquid interface at the leeward side of the droplet, as shown in Fig. 7.9b. Figure 7.9c shows that the transmitted wave reflects back from the interface at the leeward end into the droplet as a strong expansion wave. Therefore, the numerical Schlieren and the pressure contours in Fig. 7.9a–c demonstrate that the physical behavior of nonlinear wave interaction with the gas–liquid interface is captured by the current sharp-interface method. The higher pressure observed within the droplet is due to the effect of surface tension. This shows that the current GFM incorporates the effects of surface tension while accurately propagating the characteristic waves from one medium to another at the interface.

Figure 7.9d, e show, as the flow evolves further, the shock wave travels past the droplet and the droplet starts to deform. The vaporized aluminum accumulated in the wake starts to react with air in the wake of the droplet. The chemical reaction induces strong unsteadiness and asymmetry in the wake even at the low Re_D. The contours of temperature and the species mass fractions are shown in Fig. 7.10. Figure 7.10a shows that an unsteady diffusion flame forms in the wake of the droplet. The highest temperature occurs in the shear layers, close to the boundary layer detachment points behind the droplets. This is because of the high concentration of the aluminum vapor in the shear layer behind the droplets, as seen in Fig. 7.10b. The aluminum vapor generated at the front side droplet surface is mostly contained within the thin boundary layer on the droplet surface. As the boundary layer detaches from the surface at the leeward side of the droplet, the aluminum vapor accumulated in the boundary layer flows into the post-detachment shear layer. However, Fig. 7.10b shows that there in a very little amount of aluminum vapor in the recirculation region

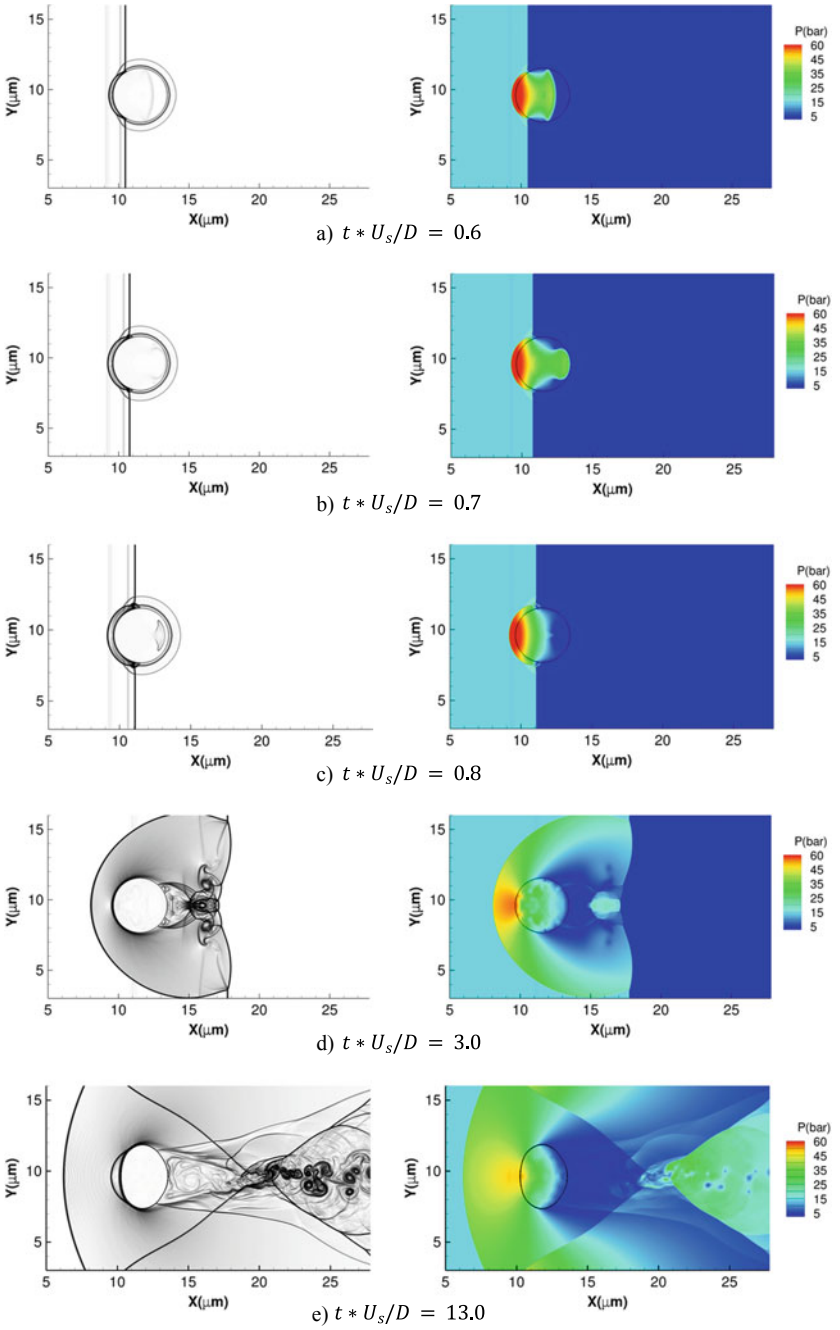


Fig. 7.9 Sequence of contour plot obtained from the simulation of Mach 3.5 shock interacting with and aluminum droplet (Red = 1000). The left column shows the numerical schlieren images. Pressure contours are shown in the right column

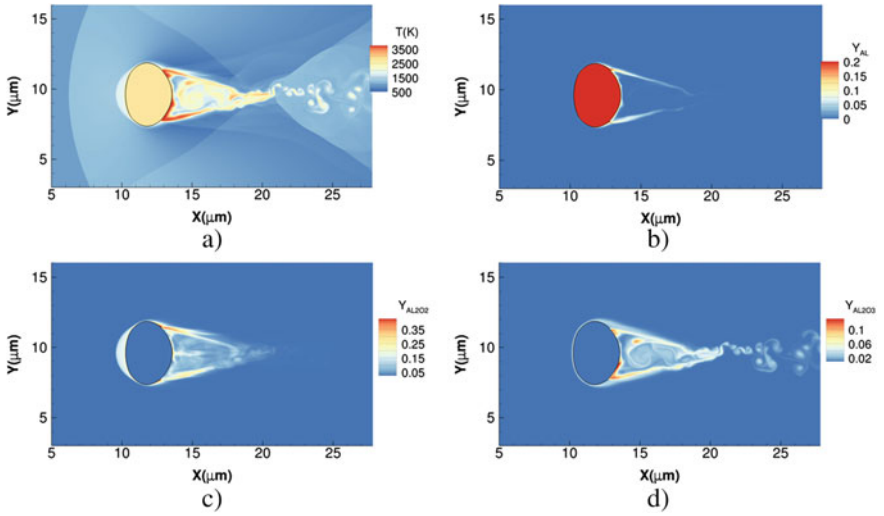


Fig. 7.10 Contours of temperature (**a**) and mass fractions of aluminum vapor (**b**), Al_2O_2 (**c**), and Al_2O_3 (**d**) during the Mach 3.5 shock-induced combustion of an aluminum droplet

behind the droplet. This is due to the fact that most of the evaporated aluminum gets oxidized in the shear layer. The evaporated aluminum in the shear layer reacts with the oxygen in the freestream and the high-temperature diffusion flame is formed. The combustion products such as Al_2O_2 and Al_2O_3 flow into the recirculation region behind the droplets. Figure 7.10c, d shows the accumulations of the combustion products within the recirculation bubble.

The results presented in this section show that the current sharp-interface method coupled with the compressible reacting flow solver successfully resolves the nuances of droplet combustion in a shocked flow. The current GFM allows the characteristic waves to travel across the gas–liquid interfaces without generating numerical artifacts. The effects of surface tension and vaporization at the liquid surface are also incorporated using the current sharp-interface method. The reacting flow simulation shows that the current method is suitable for interface-resolved simulation of droplet combustion in shocked flows.

7.3.4 Solid–Void Interactions: Reactive Pore Collapse in HMX Under 1000 m/s Shock Load

The capability of the current framework to handle shock-induced chemical reaction is analyzed by studying the initiation and growth of reaction in a porous energetic material, HMX. In porous energetic materials, the collapse of pores under shock load leads to the formation of localized heated regions called hotspots (Field John

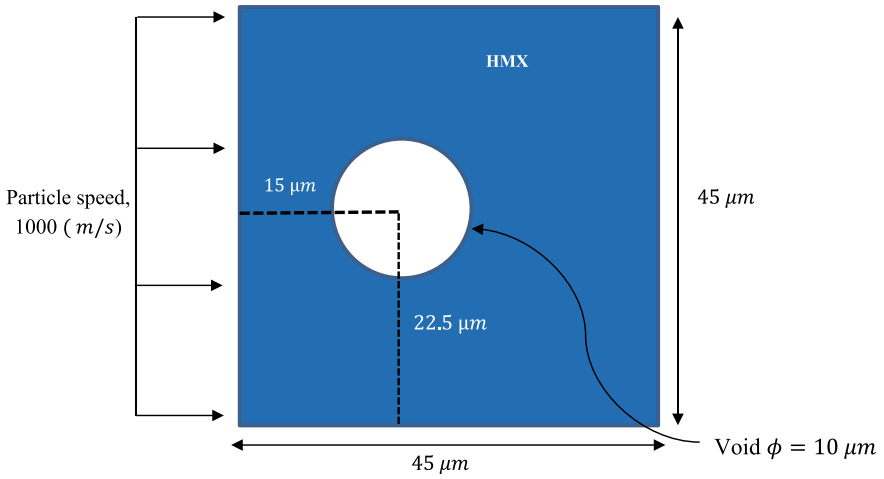
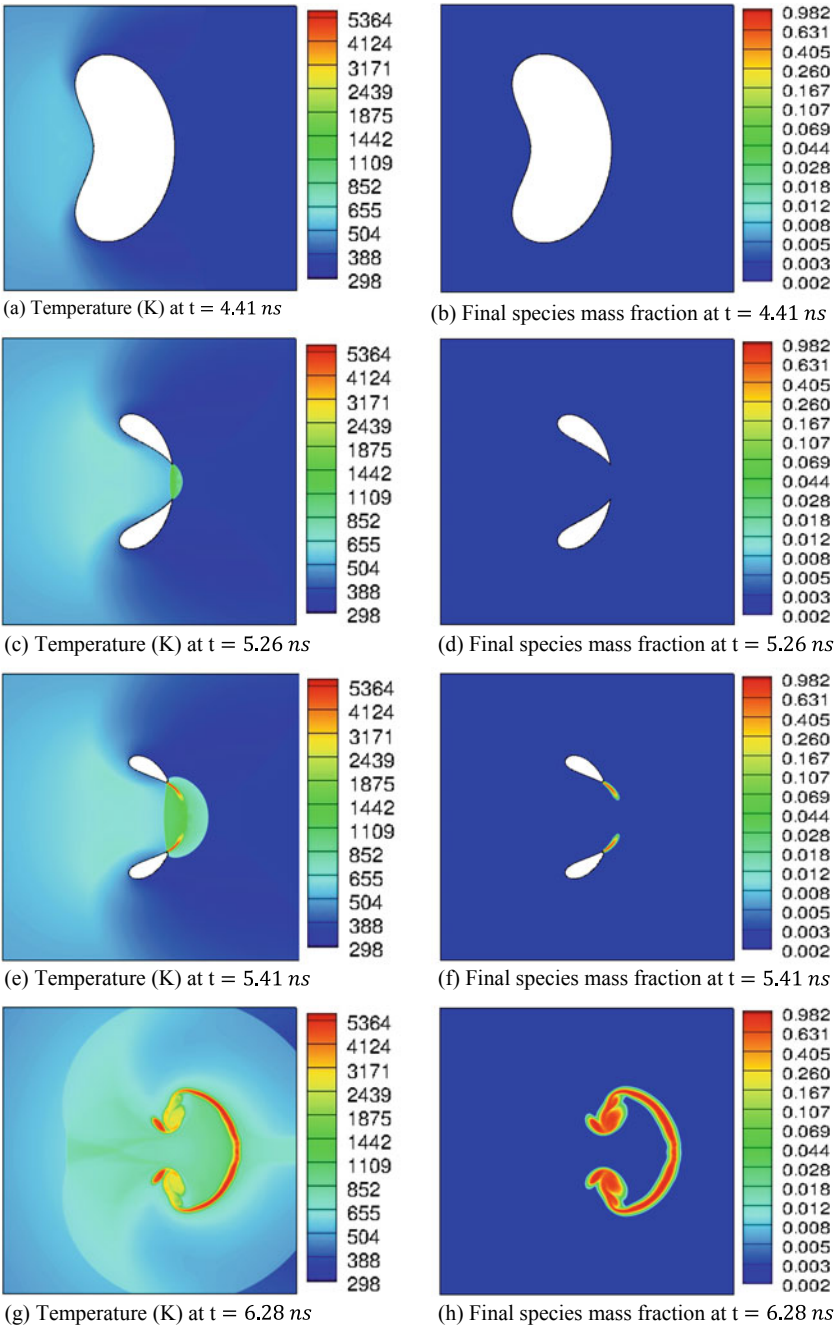


Fig. 7.11 Cylindrical void of diameter 10 μm embedded in the HMX domain of size 45 × 45 μm shock load is applied as a velocity boundary condition in the form of a pulse of duration 3 ns. The east, south, and north faces of the domain are supplied with outlet boundary condition

1992). Depending on the temperature and size of the hotspot, chemical reactions can initiate and grow in the material. To understand the behavior of hotspots in porous HMX, reactive pore collapse simulations are performed for a 10 μm diameter pore impacted under a sustained shock of particle speed, 1000 m/s as shown in Fig. 7.11. Shock load is applied from the west face of the domain boundary. The east, north, and south boundaries are supplied with zero-gradient boundary conditions. The reaction initiation in the hotspot is modeled using Arrhenius kinetics-based three steps decomposition mechanism proposed by Tarver et al. (1996). The reaction initiation in the hotspot region leads to the decomposition of solid HMX to gaseous reaction products. To define the mixture pressure, the Birch–Murnaghan equation of state is used for the HMX and JWL equation of state is implemented for the final gaseous products.

Figure 7.12 shows the temperature and mass fraction of the final gaseous species contours obtained from the reactive pore collapse analysis. Figure 7.12c, d shows that material jet impact forms near the pore interface and leads to the formation of the blast wave along with the symmetrical secondary lobes. The blast wave leads to the collapse of the secondary lobes is seen in Fig. 7.12f. The temperature at the lobe collapse locations is high enough to initiate the chemical reaction (Fig. 7.12f). As the collapse of the secondary lobes progresses, a further rise in temperature takes place which is also augmented by energy released because of chemical reactions. Eventually, the complete collapse of the secondary lobes takes place and ignites the HMX material at the secondary lobe locations. The reaction zone grows from the lobe collapse locations to its surrounding under the combined influence of convection and diffusion. It is interesting to note that for the applied shock, the rise in the temperature



(a) Temperature (K) at t = 4.41 ns

(b) Final species mass fraction at t = 4.41 ns

(c) Temperature (K) at t = 5.26 ns

(d) Final species mass fraction at t = 5.26 ns

(e) Temperature (K) at t = 5.41 ns

(f) Final species mass fraction at t = 5.41 ns

(g) Temperature (K) at t = 6.28 ns

(h) Final species mass fraction at t = 6.28 ns

Fig. 7.12 Contour plots of temperature and mass fraction of the final species at different instances of time for reactive single void collapse analysis under shock loading of 1000 m/s. The grid size for the current simulation corresponds to 700 grid points across the void diameter of 10 μm

because of the initial jet impact is not enough to initiate the reaction. Instead, reaction initiates at the offset locations where secondary lobes are collapsed.

The current analysis shows the numerical framework can predict the initiation and growth of chemical reaction in porous energetic materials.

7.3.5 *Solid–Solid Interactions: Shock Compaction of Metallic Particles Mixture*

To demonstrate the ability of the current framework to handle compaction of large clusters of particles, shock compaction of Ni/Al metallic mixtures is performed. The numerical setup consists of a collection of spherical particles (Al-Ni mixture) impacted by a copper flyer plate at the velocity of 1 km/s. The volume fraction of particles is 60%, with the rest being void space. The particles have a diameter of 20 μm and a frictionless contact is imposed at the interface. The computational domain is 200 × 200 μm. The north and south domain boundaries are frictionless walls. The copper plate is modeled as a piston that extends indefinitely to the left. Thus, the west boundary is modeled with a constant velocity of 1 km/s. The mesh size is set to have 50 grid cells across the diameter of a particle. Each particle interface is modeled by a separate levelset function to allow for full contact-separation treatment (Rai et al. 2014). Frictionless contact (sliding) condition is imposed between all particles.

Figure 7.13 shows the density and temperature profiles at time $t = 40$ ns, $t = 70$ n, and $t = 90$ ns. As seen in the figure, the Al particles undergo more deformation than the Ni particles. The nickel particles tend to form clusters. This preferential flow of Al through the Ni matrix and the clustering of Ni are observed in experiments (Eakins and Thadhani 2008) as well as in other simulations (Eakins and Thadhani 2008). As can be observed, the focused flows of aluminum pinched between other particles create jets of material with localized high temperature and velocity. The flows cause the formation of small breakaway particulate material that detaches from their original particle, as observed in previous simulations using CTH (Eakins and Thadhani 2008). The particle fragments are at high temperature and can be ejected from compacted particles with high velocity.

Vortex flows are also observed in our calculations, as in other experimental and simulation work (Nesterenko et al. 1994; Tamura and Horie 1998). A detailed view at time $t = 74$ ns is presented in Fig. 7.14. Aluminum particles form a focused flow that encounters nickel particles. The aluminum is melted and swirls around a void forming a vortex pattern. This vortex is mostly composed of aluminum; the deformability of globular Ni is too low to effectively mix with Al in this vortex, which explains the inertness of Ni-Al mixtures (Eakins and Thadhani 2009). Thus, despite the high mixing of the particles in such vortical flows, in mixtures with large differences in deformability (such as in Ni-Al systems), vortical flows do not contribute to the triggering of the chemical reaction. The compaction of spherical-shaped particles

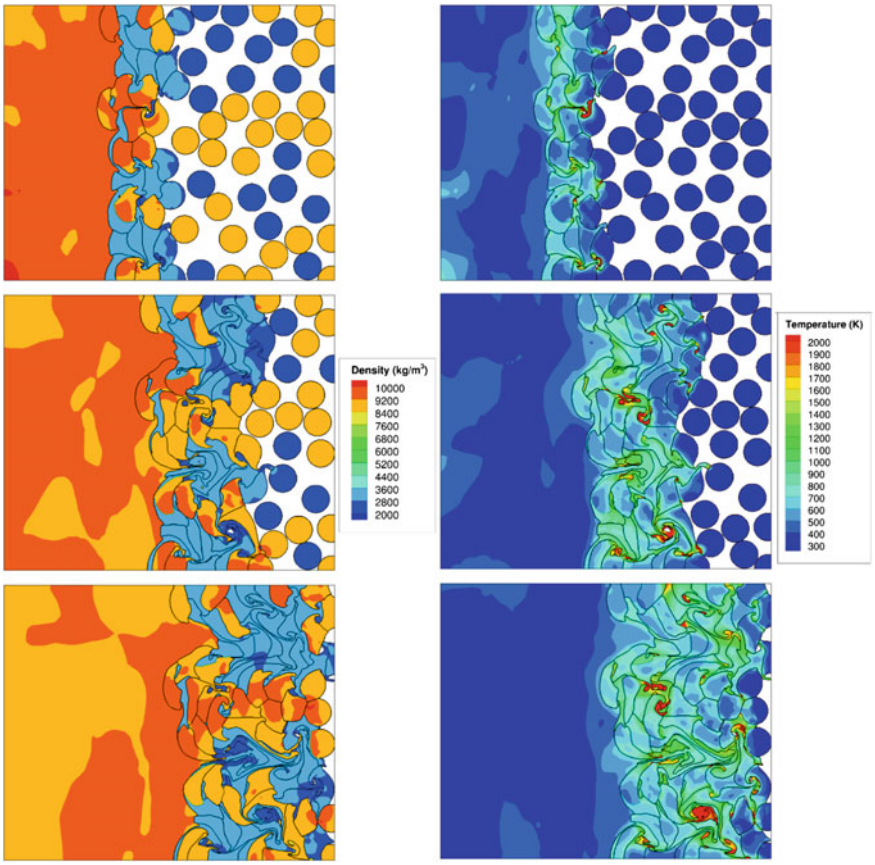


Fig. 7.13 Density and temperature fields for a Ni/Al mixture impacted by a copper flyer plate at 1 km/s at time $t = 40$ ns, $t = 70$ ns, and $t = 90$ ns

was found to present less contact area and mass mixing between aluminum and nickel particles, and thus to be less sensitive to shock-initiated chemical reaction (Eakins and Thadhani 2008). Flake Ni particles mixed with Al were found to yield more intimate mixing (Eakins and Thadhani 2006). Thus, particle morphology plays a significant role in initiating reactions.

Therefore, the current analysis shows that the current Eulerian framework can efficiently handle contact and impact situation leading to the large deformation of deformable solids that exhibit elastoplastic behavior under shock loading.

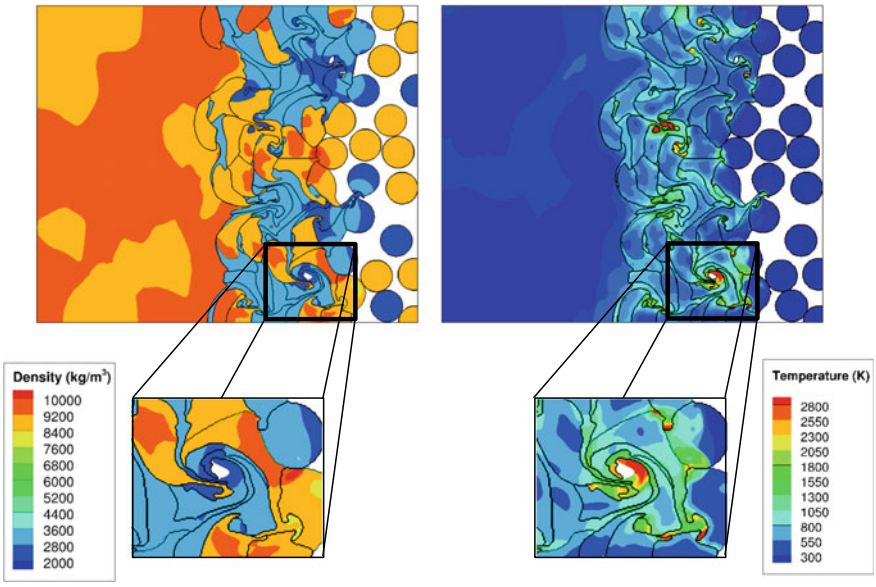


Fig. 7.14 Density and temperature fields for a Ni/Al mixture impacted by a copper flyer plate at 1 km/s at time $t = 74$ ns. Detail of the particle compaction

7.4 Conclusions

A versatile sharp-interface Eulerian method is presented for the interface-resolved simulations of high-speed multi-material flows. The levelset method is used to track the large deformation of the material interfaces. A modified ghost fluid method is used to supply the appropriate boundary conditions to the corresponding materials at the interfaces. The current method allows for a broad-range of high-speed multi-material flow problems involving interactions between solid and fluid phases in a generic Cartesian grid-based Eulerian framework.

The generality of the current framework is owing to the hyperbolic nature of the governing equations and the sharp-interface method for capturing the complex interfacial dynamics. The hypo-elastic model for stress-train relations under the high-strain rate assumptions allows us to cast the governing equations for solid materials in an Eulerian framework. Therefore, a unified Eulerian framework is used for solving the governing equation for both solid and fluid phases under high-speed conditions.

The multi-material interfaces embedded in the fixed Eulerian grid are tracked sharply using levelset methods. The levelset method allows us to track extreme topological changes in the material interfaces with ease and robustness. The simulations presented in the results section demonstrate that the levelset method can be used to track extreme deformation of multiple closed interfaces within a single simulation. The levelset-based sharp-interface tracking coupled with current GFM is used to couple the field variables of different materials at the interfaces accurately. Different

interfacial flow phenomenon such as nonlinear wave interaction with the interfaces, the effects of phase change and surface tension at the interface are modeled using the current GFM. The current GFM is suitable for imposing the appropriate boundary conditions at the interfaces separating different materials and/or phases.

The capability of the current framework to handle large deformation, phase change, and chemical reactions is demonstrated using a wide variety of problems involving high-speed multi-material interactions between the solid, liquid, and gaseous phases. The interactions of rigid solid–fluid, fluid–fluid, and deformable solid–void interfaces are demonstrated through three different problems in the results section, i.e., shock-induced lift-off a rigid cylinder and combustion of aluminum droplet under a Mach 3.5 shock, and pore collapse-induced reaction initiation in energetic materials under shock load. Shock interactions with multi-material interfaces are captured accurately in the current method. The simulation of shock interactions with an aluminum droplet demonstrates that the current Riemann solver-based GFM allows the nonlinear wave to travel across the gas–liquid interfaces without incurring any artificial numerical artifact. The sharp-interface multiphase framework is shown to efficiently handle the large deformation involved in the collapse of the pore in HMX to form hotspot. The multiphase framework allows to track the reaction initiation and expansion of the reaction products from the hotspots to the surrounding involving the decomposition of solid HMX. Therefore, the three problems demonstrate the robustness of the sharp-interface multiphase framework to handle large deformation, phase change, and chemical reactions in wide variety of materials.

The robustness and efficiency of the current framework to handle process scale simulations are demonstrated by solving two problems involving many particles, i.e., shock interaction of cluster of rigid particles and the compaction of Ni/Al metallic powder under shock load. Each particle is defined using the narrow-band levelset approach. The simulations of shock interaction of particle cluster demonstrate the capabilities of the current numerical method to perform resolved simulations of particle-laden flows. The current method is used in Das et al. (2018) to develop surrogate models for drag on particles in shocked flows from resolved 3D simulations. In the simulation of shock-induced compaction of Ni/Al powder, the definition of the particles using different levelsets allows to model the physics governing the compaction between the Ni/Al particles accurately. The sharp-interface tracking of the particles is shown to handle the localized extreme deformation situations efficiently that arise during the compaction of the metallic powder bed. Therefore, these simulations demonstrate the capability of the current framework to perform large scale simulations relevant to real-world engineering applications.

Acknowledgements We gratefully acknowledge the financial support by the Air Force Office of Scientific Research under grant numbers FA9550-15-1-0332 (Program Officer: Dr. Martin Schmidt) and SA0000506 (Program Officer: Dr. Fariba Fahroo).

References

Artemieva NA, Shuvalov VV (2008) Numerical simulation of high-velocity impact ejecta following falls of comets and asteroids onto the Moon. Sol Syst Res 42(4):329–334
Boiko VM et al (1997) Shock wave interaction with a cloud of particles. Shock Waves 7(5):275–285
Burcat's Thermodynamic Data
Bürger D et al (2012) Ballistic impact simulation of an armour-piercing projectile on hybrid ceramic/fiber reinforced composite armours. Int J Impact Eng 43:63–77
Chaudhuri A et al (2013) Numerical study of shock-wave mitigation through matrices of solid obstacles. Shock Waves 23(1):91–101
Das P et al (2017) A sharp interface Cartesian grid method for viscous simulation of shocked particle-laden flows. Int J Comput Fluid Dyn 31(6–8):1–23
Das P, UdayKumar HS (2019) A sharp-interface method for the simulation of shock-induced vaporization of droplets. J Comp Phys (in press)
Das P et al (2018a) Metamodels for interphase heat transfer from mesoscale simulations of Shock-Cylinder Interactions. AIAA Journal 56(10):3975–3987
Das P et al (2018b) Strategies for efficient machine learning of surrogate drag models from three-dimensional mesoscale computations of shocked particulate flows. Int J Multiph Flow 108:51–68
Dongmo E, Wenzelburger M, Gadow R (2008) Analysis and optimization of the HVOF process by combined experimental and numerical approaches. Surf Coat Technol 202(18):4470–4478
Eakins D, Thadhani NN (2006) Shock-induced reaction in a flake nickel + spherical aluminum powder mixture. J Appl Phys 100(11):113521
Eakins DE, Thadhani NN (2008a) The shock-densification behavior of three distinct Ni + Al powder mixtures. Appl Phys Lett 92(11):111903
Eakins DE, Thadhani NN (2008b) Mesoscale simulation of the configuration-dependent shock-compression response of Ni + Al powder mixtures. Acta Mater 56(7):1496–1510
Eakins DE, Thadhani NN (2009) Shock compression of reactive powder mixtures. Int Mater Rev 54(4):181–213
Fedkiw RP et al (1999) A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the Ghost Fluid Method). J Comput Phys 152(2):457–492
Field John E et al (1992) Hot-spot ignition mechanisms for explosives and propellants. Philos Trans R Soc London Ser A: Phys Eng Sci 339(1654):269–283
Gottlieb S, Shu C-W (1998) Total variation diminishing Runge-Kutta schemes. Math Comput Am Math Soc 67(221):73–85
Hirt CW, Nichols BD (1981) Volume of fluid (VOF) method for the dynamics of free boundaries. J Comput Phys 39(1):201–225
Houim RW (2011) Modeling the influence of shock waves on the combustion of aluminum droplets
Houim RW, Kuo KK (2013) A ghost fluid method for compressible reacting flows with phase change. J Comput Phys 235:865–900
Huang Y et al (2009) Effect of particle size on combustion of aluminum particle dust in air. Combust Flame 156(1):5–13
Jamaluddin AR et al (2011) The collapse of single bubbles and approximation of the far-field acoustic emissions for cavitation induced by shock wave lithotripsy. J Fluid Mech 677:305–341
Jiang G-S, Shu C-W (1996) Efficient implementation of weighted ENO schemes. J Comput Phys 126(1):202–228
Khan A, Huang S (1995) Continuum theory of plasticity. Wiley, New York
Liu TG, Khoo BC, Wang CW (2005) The ghost fluid method for compressible gas–water simulation. J Comput Phys 204(1):193–221
Marusich TD, Ortiz M (1995) Modelling and simulation of high-speed machining. Int J Numer Meth Eng 38(21):3675–3694
Massoni J et al (1999) A mechanistic model for shock initiation of solid explosives. Phys Fluids 11(3):710–736

Mayer W, Tamura H (1996) Propellant injection in a liquid oxygen/gaseous hydrogen rocket engine. J Propul Power 12(6):1137–1147
Mehta Y et al (2016) Shock interaction with three-dimensional face centered cubic array of particles. Phys Rev Fluids 1(5):054202
Meyers MA (1994) Dynamic behavior of materials. Wiley, New York
Mousel J (2012) A massively parallel adaptive sharp interface solver with application to mechanical heart valve simulations. Theses and Dissertations
Nesterenko VF et al (1994) Controlled high-rate localized shear in porous reactive media. Appl Phys Lett 65(24):3069–3071
Osher S, Sethian JA (1988) Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J Comput Phys 79(1):12–49
Ponthot JP (2002) Unified stress update algorithms for the numerical simulation of large deformation elasto-plastic and elasto-viscoplastic processes. Int J Plast 18(1):91–126
Powell OA et al (2001) Development of hydrocarbon-fueled scramjet engines: the hypersonic technology (HyTech) program. J Propul Power 17(6):1170–1176
Rai NK, Kapahi A, Udaykumar HS (2014) Treatment of contact separation in Eulerian high-speed multimaterial dynamic simulations. Int J Numer Meth Eng 100(11):793–813
Regele JD et al (2014) Unsteady effects in dense, high speed, particle laden flows. Int J Multiph Flow 61:1–13
Sambasivan SK, UdayKumar HS (2009) Ghost fluid method for strong shock interactions Part 1: fluid-fluid interfaces. AIAA J 47(12):2907–2922
Sambasivan S, Kapahi A, Udaykumar HS (2013) Simulation of high speed impact, penetration and fragmentation problems on locally refined Cartesian grids. J Comput Phys 235:334–370
Scardovelli R, Zaleski S (2000) Analytical relations connecting linear interfaces and volume fractions in rectangular grids. J Comput Phys 164(1):228–237
Sethian JA, Smereka P (2003) Level set methods for fluid interfaces. Annu Rev Fluid Mech 35(1):341–372
Shiv Kumar S, UdayKumar HS (2009) Ghost fluid method for strong shock interactions Part 2: immersed solid boundaries. AIAA j 47(12):2923–2937
Shu C-W, Osher S (1989) Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. In: Hussaini PMY , Leer PBV, Rosendale DJV (eds) Upwind and high-resolution schemes. Springer, Berlin, pp 328–374
Sussman M, Smereka P, Osher S (1994) A level set approach for computing solutions to incompressible two-phase flow. J Comput Phys 114(1):146–159
Sussman M et al (1998) An improved level set method for incompressible two-phase flows. Comput Fluids 27(5–6):663–680
Tamura S, Horie Y (1998) Discrete meso-dynamic simulation of thermal explosion in shear bands. J Appl Phys 84(7):3574–3580
Tarver CM, Chidester SK, Nichols AL (1996) critical conditions for impact- and shock-induced hot spots in solid explosives. J Phys Chem 100(14):5794–5799
Thadhani NN (1988) Shock compression processing of powders. Adv Mater Manuf Process 3(4):493–549
Unverdi SO, Tryggvason G (1992) A front-tracking method for viscous, incompressible, multi-fluid flows. J Comput Phys 100(1):25–37
Verwer JG, Sommeijer BP, Hundsdorfer W (2004) RKC time-stepping for advection—diffusion—reaction problems. J Comput Phys 201(1):61–79

Chapter 8
Development and Application of Immersed Boundary Methods for Compressible Flows

Santanu Ghosh and Anand Bharadwaj S

8.1 Introduction

The advent of immersed boundary methods can be traced back to the work of Peskin (1972), which was devised for the simulation of blood flow through the mitral valves in the heart. The initial formulation of the method was used for solving the incompressible Navier–Stokes equations past immersed objects, which were treated as elastic massless and volumeless entities, and geometrically rendered as a set of connected line segments. The effect of the embedded or immersed boundary was introduced as a singular body force in the Navier–Stokes equation. The forces were calculated by determining the deformation of the elastic immersed boundary and using Hookes' law. The deformation of the immersed boundary, from an initial undeformed state, happens as the immersed boundary is made to obey a no-slip condition. Both the application of the no-slip constraint on the immersed boundary and the distribution of the body force in the neighbouring fluid nodes of the immersed boundary made use of semi-discrete Dirac delta functions. In addition to the constraint that the immersed object needed to have elastic boundaries, another drawback of this approach was that the immersed interface 'appeared' effectively smeared to the rest of the flow—an effect of spreading of the boundary force over multiple grid nodes, which is why these methods are also known as diffuse interface methods. The next two decades witnessed advances in the development and application of the immersed boundary approach; notable among these was the extension of Peskin's method by Goldstein et al. (1993) for simulating flow past almost rigid immersed boundaries. Goldstein's method employed feedback forcing where the forcing function was given by $f(x_s, t) = \alpha \int_0^t U(\mathbf{x_s}, t')\mathrm{d}t' + \beta U(\mathbf{x_s}, t')$, where α

S. Ghosh (✉) · Anand Bharadwaj S
Indian Institute of Technology Madras, Chennai 600036, India
e-mail: sghosh1@iitm.ac.in

Anand Bharadwaj S
e-mail: anandbharadwaj1950@yahoo.com

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_8

and β are tunable parameters, and $U(\mathbf{x_s}, t')$ is the velocity of the embedded surface. This method was applied to simulate turbulent flow in ribbed channels. However, the method suffered from a severe constraint in the allowable time step for stable integration of the solution, which made its use somewhat impractical. Fadlun et al. (2000) employed a method of direct forcing, unlike the continuous forcing employed by Peskin (1972), and Goldstein et al. (1993), that did not suffer from the time step restriction of Goldstein's method and was used to simulate the turbulent flow in an IC piston/cylinder assembly. This method did not need require an explicit addition of the forcing term in the momentum equation, but relied on the reconstruction of the velocity field in the neighbourhood of the immersed surface using interpolations, that implicitly made the solution obey the *no-slip boundary condition* at the immersed surface.

Subsequently, a lot of work has been done in the development of sharp-interface immersed boundary methods for incompressible flows. These methods primarily differed in their mode of solution reconstruction—with some methods adopting reconstruction at the fluid nodes in the immediate neighbourhood (outside) of the immersed surface (Yang and Balaras 2006; Choi et al. 2007), while other methods (Tseng and Ferziger 2003) relied on a *ghost* cell approach.

In contrast, immersed boundary methods for compressible flows have been developed more recently. The earliest work involving immersed boundary method for compressible flow applications was published by De Palma et al. (2006). In this work, they employed the direct-forcing approach of Fadlun et al. (2000) for the reconstruction of the flow variables. Their formulation was designed to solve flow at all speeds, for which they used preconditioning of the pseudo-time-derivative. The turbulence model used in their work was the $k - \omega$ model of Wilcox (1994). They presented results, which included contour plots, surface pressure coefficients and integrated drag coefficients, for a wide range of test cases. The results compared well with values reported in literature and computations on body-fitted grids done as part of their work. Ghias et al. (2007) subsequently developed a ghost-cell-based IB approach for subsonic compressible flows. In this particular study, the authors presented 2D and 3D flow simulations using their immersed boundary approach for flow past stationary objects. The method resulted in sharp resolution of interfaces and could be used for both Cartesian and general curvilinear mesh topologies. In the same year, de Tullio et al. (2007) presented a Cartesian grid-based immersed boundary method for simulation of compressible turbulent flows past stationary objects in which they used a local grid refinement (LGR) strategy to avoid the use of globally high-density meshes for the simulation of turbulent flows. The method was shown to be formally second-order accurate. Investigations of shock–obstacle interactions for stationary and moving shocks were presented by Chaudhuri et al. (2011), in which they proposed a quadratic reconstruction of the solution near the immersed surface in a direct-forcing framework. Their results showed good agreement with literature for complex shock interactions using the immersed boundary approach.

In the recent past, Tran and Plourde (2014) have developed a compressible immersed boundary method, with the aim to use it for combustion-related problems and hypersonic flows. This method implements wall-slip- and wall-injection-type

boundary conditions and does not include a no-slip formulation. This method also uses a 2^n tree-type mesh refinement strategy for local mesh refinement. More recently, Brehm et al. (2015) have developed a second-order accurate immersed boundary method suitable for compressible viscous flows with improved stability on Cartesian grids. The method improves the stability by investigating the finite difference coefficients involved in the solution reconstruction at the irregular fluid nodes in the immediate (external) neighbourhood of the IB.

The challenges of formulating an immersed boundary method that can be used for compressible, high Reynolds number flows with turbulence modelling include sharp capturing of shock waves and modelling the energising effects of turbulence on mean velocity profile. While the former requires that the immersed boundary method is sharp-interface type (Ghias et al. 2007; De Palma et al. 2006; de Tullio et al. 2007), the latter ideally requires high grid resolution, with wall-normal spacings of the order of wall units (De Palma et al. 2006), for application with turbulence models that resolve the near-wall turbulent boundary layer (Wilcox 1994; Menter 1994). Although using such grid spacing is commonplace with body-fitted grids, such high resolution can lead to the construction of large grids when used with immersed geometries, as the internal volume of the embedded geometry is also gridded in such cases. The alternative approach is to use adaptive mesh refinement with unstructured Cartesian mesh and/or devise the solution forcing in an ingenious manner such that the energising effects of a turbulent boundary layer is reproduced without the need to accurately resolve the boundary layers in the immediate neighbourhood of the immersed surface. The method presented here (Ghosh et al. 2010) is, to the best knowledge of the authors, the earliest approach wherein a power-law reconstruction of the velocity combined with law-of-the-wall type forcing of the turbulence variables—turbulence kinetic energy, k, and specific dissipation rate, ω (Menter 1994)—was combined to mimic the energising effects of a turbulent boundary layer in compressible high Reynolds number flows on curvilinear grids. The IBM of Ghosh et al. (2010) has been validated and used extensively for the simulation of boundary layer control devices in supersonic turbulent flows (Ghosh 2010; Varma and Ghosh 2017; Sharma et al. 2016; Roy et al. 2017; Sandhu et al. 2018). Subsequent efforts along these lines include the works of Capizzano (2011), wherein a two-layer model of the flow—with separate governing equations for each layer—was adopted, Bernardini et al. (2016), which made use of modified wall functions near the immersed surface, and Tamaki et al. (2017), in which the authors modified the boundary conditions for the tangential velocity profile and turbulence model (Spalart and Allmaras 1992) in the vicinity of the immersed surface.

8.2 Computational Methodology

The immersed boundary method outlined in this work is integrated into a finite volume solver for compressible turbulent flows on block-structured grids. A brief description of the flow solver and the immersed boundary method are presented in this section.

8.2.1 Flow Solver

The immersed boundary method outlined in this work is built into the framework of a finite volume solver for compressible turbulent flows on multi-block-structured grids (Roy and Edwards 2000). The density-weighted Reynolds-averaged Navier–Stokes equations are solved by discretising the equations on curvilinear grids. A flux-splitting method, the low-diffusion flux-splitting scheme (LDFSS) (Edwards 1997), is used to construct the inviscid convective flux and pressure flux, whereas central-difference-based methods are used for the determination of the viscous fluxes. Higher-order solution reconstruction at the cell interfaces is achieved using the piecewise parabolic method of Colella and Woodward (1984). A first-order Euler-implicit time integration with local time stepping is performed for steady flows and the second-order Crank-Nicolson scheme is used for time-accurate simulations. Menter's $k - \omega$/SST (Menter 1994) turbulence model is used for the steady RANS computations and a hybrid large eddy simulation (LES)/RANS method, as outlined in the work of Choi et al. (2011), is used for time-accurate computations. The solver is designed for parallel execution using message passing interface (MPI) paradigm for data communication across processor cores. Extensive details of the solver can be found in the work of Varma and Ghosh (2017).

8.2.2 Discrete Solution Forcing

In this work, the surface of the immersed object is rendered using a cloud of points, wherein the local outward unit normal to the surface is also specified. The unit normal provides information about the surface orientation and classification of the fluid mesh into external and internal cells. A signed distance function, ϕ, is used for this purpose as defined in Eq. 8.1,

$$\Phi = \left(\text{sgn}(\mathbf{x_C} - \mathbf{x_S}) \cdot \hat{n}\right) |\mathbf{x_S} - \mathbf{x_C}| \tag{8.1}$$

where $\mathbf{x}$ represents the position vector of a point designated by its subscript; here, subscripts $\mathbf{C}$ and $\mathbf{S}$ refer to the cell centres of a fluid cell and any surface point respectively, as shown in Fig. 8.1. The nearest surface point to C is then determined using an approximate nearest neighbour algorithm (Arya et al. 1998), and the corresponding signed distance is stored. For closed immersed surfaces, the signed distance returns a positive value for cells external to the immersed surface, which are termed as field cells, and negative value for the cells internal to the immersed surface, which are termed as internal cells accordingly. In addition, any field cell, which has an internal cell as any of its neighbours that share a node with it, is termed as a band cell. Additionally, a sharp Heaviside step function $G[\Phi(\mathbf{x}_c, t)]$ is defined such that it takes a value of unity for interior and band cells, and zero for field cells. A schematic of the classification described above is given in Fig. 8.1.

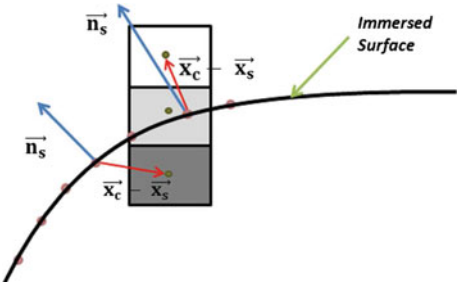


Fig. 8.1 Classification of cells: field cells (white), band cells (light grey) and interior cells (dark grey). Reproduced from Varma and Ghosh (2017) (copyright held by author Santanu Ghosh)

The solution is forced or reconstructed using neighbouring field cell data in the band cells in a manner which implicitly enforces the boundary condition at the immersed surface—which results in what is termed as direct forcing. In order to determine the properties in a band cell, an interpolation stencil is first determined by choosing the immediate neighbours of a band cell; these are cells that share at least a node with the band cell. A point along the normal to the nearest surface point to the band cell is then constructed; this point is referred to as the 'interpolation' point. The determination of the location of the interpolation point and the flow properties at that location are determined using inverse distance weights (Choi et al. 2007).

At the interpolation point and band cell centre, the velocity relative to the velocity at the nearest surface point is split into two components—one normal to the immersed surface ($\mathbf{u}_\mathrm{N}$) and the other tangential to the surface ($\mathbf{u}_\mathrm{T}$)—as shown in Eq. 8.2

$$\mathbf{u} - \mathbf{u}_\mathrm{s} = \mathbf{u}_\mathrm{T}(n) + \mathbf{u}_\mathrm{N}(n) \tag{8.2}$$

where $\mathbf{u}$ is the velocity at the band cell centre or interpolation point, $\mathbf{u}_\mathrm{s}$ is velocity at it the surface point closest to the band cell centre, and n denotes the coordinate normal to the surface. The tangential component of the relative velocity at the band cell ($\mathbf{u}_\mathrm{T}(d_B)$) is then constructed using a power-law function of the distance d from the nearest surface point (Eq. 8.3a), whereas the normal component ($\mathbf{u}_\mathrm{N}(d_B)$) is chosen to satisfy a discrete continuity condition (Ghosh 2010) as shown in Eq. 8.3

$$\mathbf{u}_\mathrm{T}(d_B) = \mathbf{u}_\mathrm{T}(d_I)\left(\frac{d_B}{d_I}\right)^k \tag{8.3a}$$

$$\mathbf{u}_\mathrm{N}(d_B) = \mathbf{u}_\mathrm{N}(d_I)g'(\rho, d_I, d_B) \tag{8.3b}$$

where,

$$\mathbf{u}_\mathrm{N}(d_I) = \{[\mathbf{u}(d_I) - \mathbf{u}(s)].\mathbf{n}_s\}\mathbf{n}_s \tag{8.4a}$$

$$\mathbf{u}_\mathrm{T}(d_I) = [\mathbf{u}(d_I) - \mathbf{u}(s)] - \mathbf{u}_\mathrm{N}(d_I) \tag{8.4b}$$

In Eq. 8.3, subscripts I and B represent the interpolation and band points, respectively. The choice of the power-law index k allows the model to mimic the effects a

turbulent velocity profile ($k = 1/7$ or $k = 1/9$) or a laminar profile ($k = 1$). A value of $k = 1$ is also used for turbulent flows if the wall-normal grid resolution to the IB surface is sufficient to resolve the laminar sub-layer. Here, g' is a scaling function obtained from solving a discrete continuity equation near the surface (Ghosh et al. 2010). For an adiabatic wall, g' is given by:

$$g' = \frac{1}{\tilde{\rho}} \left(\frac{\frac{d_B}{d_I} d^- \tilde{\rho}^-}{\frac{d_B}{d_I} d^- \tilde{\rho}^- + \left(1 - \frac{d_B}{d_I}\right) d^+ \tilde{\rho}^+} \right) \tag{8.5}$$

where,

$$\begin{aligned}
d^+ &= \left(\frac{d_I + d_B}{2 d_I} \right)^k, \quad d^- = \left(\frac{d_B}{2 d_I} \right)^k \\
\frac{1}{\tilde{\rho}} &= 1 + \frac{r(\gamma - 1)}{2\gamma \mathrm{RT}(d_I)} \left[\mathbf{u_T}(d_I) \cdot \mathbf{u_T}(d_I) \right] \left(1 - \left(\frac{d_B}{d_I} \right)^{2k} \right) \\
\frac{1}{\tilde{\rho}^+} &= 1 + \frac{r(\gamma - 1)}{2\gamma \mathrm{RT}(d_I)} \left[\mathbf{u_T}(d_I) \cdot \mathbf{u_T}(d_I) \right] \left(1 - (d^+)^2 \right) \\
\frac{1}{\tilde{\rho}^-} &= 1 + \frac{r(\gamma - 1)}{2\gamma \mathrm{RT}(d_I)} \left[\mathbf{u_T}(d_I) \cdot \mathbf{u_T}(d_I) \right] \left(1 - (d^-)^2 \right)
\end{aligned} \tag{8.6}$$

The temperature in the band cell is reconstructed using Walz's relation (Wilcox 1994) for temperature distribution within a compressible boundary layer on an adiabatic or isothermal wall as given in Eq. 8.7. Here, r is the recovery factor, and $[\mathbf{u_T}(d_I)]^2$ represents the kinetic energy associated with the tangential velocity at the interpolation point.

$$\frac{T_B}{T(d_I)} = 1 + \frac{r(\gamma - 1)}{2\gamma \mathrm{RT}(d_I)} [\mathbf{u_T}(d_I) \cdot \mathbf{u_T}(d_I)] \left[1 - \left(\frac{d_B}{d_I} \right)^{2k} \right] \quad \text{(adiabatic wall)} \tag{8.7a}$$

$$\begin{aligned}
\frac{T_B}{T(d_I)} = \frac{T_w}{T(d_I)} &+ \left(1 - \frac{T_w}{T(d_I)} + \frac{r(\gamma - 1)}{2\gamma \mathrm{RT}(d_I)} [\mathbf{u_T}(d_I) \cdot \mathbf{u_T}(d_I)] \right) \left(\frac{d_B}{d_I} \right)^k \\
&- \frac{r(\gamma - 1)}{2\gamma \mathrm{RT}(d_I)} [\mathbf{u_T}(d_I) \cdot \mathbf{u_T}(d_I)] \left(\frac{d_B}{d_I} \right)^{2k} \quad \text{(isothermal wall)}
\end{aligned} \tag{8.7b}$$

The density in the band cells can be obtained using two different approaches: by integrating the continuity equation in the band cells using the reconstructed velocity and temperature or through the equation of state using the interpolated values of pressure and temperature at the band cell. For the latter approach, the pressure at the band cell is extrapolated from the interpolation point.

The turbulence variables in the band cells are computed using law-of-the-wall-type functions as given in Eq. 8.8.

$$
\begin{aligned}
k_B &= \frac{u_\tau^2}{\sqrt{C_\mu}}, \quad \omega_B = \frac{u_\tau}{\left(\sqrt{C_\mu}\kappa d_B\right)} : d^+ > 10.934 \\
k_B &= k_(d_I)\left(\frac{d_B}{d_I}\right)^2, \quad \omega_B = \frac{60\nu_w}{0.075 d_B^2} : d^+ < 10.934 \quad \text{or} \quad k = 1 \\
d^+ &= \frac{u_\tau d_B}{\nu_w}, \quad u_\tau = \frac{|u_{\mathrm{T}}(d_I)|}{\frac{\ln(d^+)}{\kappa} + 5.1} \quad \text{(iterative solution)}
\end{aligned} \tag{8.8}
$$

The residual vector, $\mathbf{R}$, at any band cell is then constructed by blending the Navier–Stokes residual and an additional source term as,

$$
\mathbf{R}^{n+1,l} = [1 - G(\Phi^{n+1})]\mathbf{R}_{NS}^{n+1,l} + G(\Phi^{n+1})\left[\frac{\mathbf{V}^{n+1,l} - \mathbf{V_B}^{n+1,l}}{\Delta t}\right] \tag{8.9}
$$

where l is a sub-iteration index, and $\mathbf{V} = [\rho, u, v, w, T, k, \omega]$ is the primitive variable vector. The solution (interpolated values) in the band cells are obtained by implicitly solving the system of equations in band cells, with the residual as defined in Eq. 8.9, coupled with other cells using sub-iterations.

8.2.3 Data Reconstruction on Immersed Surfaces

Pressure and shear stress (and/or heat flux) at the immersed surfaces in a flow field are often needed for conjugate stress analysis, FSI problems (Borazjani et al. 2008; Miller and Peskin 2009; Zheng et al. 2010; Sotiropoulos and Yang 2014; Brehm et al. 2015), load estimation (Zastawny et al. 2012), etc. However, surfaces treated as immersed boundaries, in general, do not coincide with grid points, unlike the case of body-fitted grids wherein the grid conforms to all surfaces. The data at the immersed surface, therefore, has to be reconstructed using the solution in the underlying grid (nodes or cell centres) to the points on the immersed surface. Here, we present an approach for interpolation of pressure and shear stress using inverse distances, that has been developed by the authors (Bharadwaj S and Ghosh 2018).

The data reconstruction procedure starts with the construction of interpolation stencil(s) for every point on the immersed surface. Using the solution in the cells in the stencil(s), we first estimate the pressure and the component of velocity—tangential to the local surface—at specific points inside the stencil, referred to as *interpolation points*, and then proceed to reconstruct the pressure and shear stresses at the immersed surface using the values at the interpolation point(s) and the surface point (in case of shear stress estimation).

8.2.3.1 Stencil Construction

In the interpolation methods we consider here, two interpolation stencils are required for every point on the immersed surface as shown in Fig. 8.2. The first interpolation

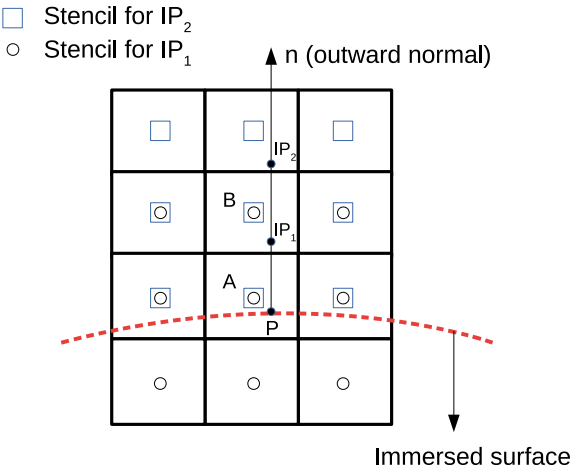


Fig. 8.2 Interpolation stencil

stencil is built by finding the cell centre (A) that lies closest to the surface point (P). Using this cell centre, a 3×3 stencil is built by considering all neighbouring cells that share a grid node with this cell. The cell centres of this stencil are shown by circles. Using the field and band cells of this stencil, the first interpolation point (IP_1) is constructed. The second stencil is built by finding the cell centre (B) in the first stencil that lies closest to the first interpolation point. Using the second stencil, marked as squares, a second interpolation point (IP_2) is determined.

8.2.3.2 Interpolation Point: Location and Properties

In Fig. 8.3, the circles represent the cell centres of the stencil that are used to determine the location and properties at the interpolation point (IP). Consider a cell centre B. The perpendicular distance from B to the normal at the immersed surface is given by $d_{1,i}$. The distance, $d_{2,i}$, is the projection of the distance from B to the surface point P, along the normal.

The distance of the interpolation point from the immersed surface, measured along the normal to the surface, is determined as:

$$d_{\text{IP}} = \frac{\sum_i d_{2,i}/d_{1,i}}{\sum_i 1/d_{1,i}} \tag{8.10}$$

where the summation holds over the field and band cells of the stencil. Any general property, ϕ, at the interpolation point is also interpolated in a similar way:

$$\phi_{\text{IP}} = \frac{\sum_i \phi_i/d_{1,i}}{\sum_i 1/d_{1,i}} \tag{8.11}$$

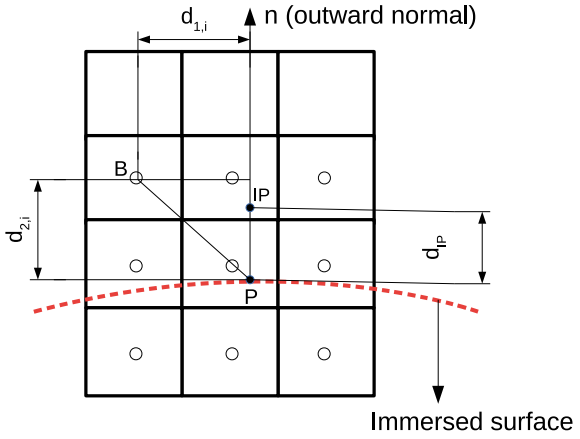


Fig. 8.3 Interpolation point location and properties

Thus, using Eq. 8.11, we interpolate pressure and velocity components at the interpolation point.

8.2.3.3 Pressure Reconstruction

With the interpolated pressures at IP_1 and IP_2, the pressure at the immersed surface (point P) is reconstructed as:

$$P_P = \frac{P_{IP_1} d_{IP_2} - P_{IP_2} d_{IP_1}}{d_{IP_2} - d_{IP_1}} \tag{8.12}$$

where P_{IP_1}, P_{IP_2} are the pressures at the two interpolation points and d_{IP_1}, d_{IP_2} are their respective distances from the immersed surface. Equation 8.12 is obtained by assuming a linear variation of pressure along the normal to the surface.

8.2.3.4 Shear Stress Reconstruction

The velocity at the interpolation points relative to the velocity at the local surface (point P in Fig. 8.3) is first decomposed into components tangential (V_τ) and normal (V_n) to the local surface. We assume a quadratic variation of the relative tangential velocity component along the normal to the surface, given by:

$$V_\tau = an^2 + bn + c \tag{8.13}$$

where n is the coordinate in the direction of the outward normal. The coefficients a, b and c are constants that are determined by using the following three conditions:

$$\begin{aligned} V_\tau|_{n=0} &= 0 \\ V_\tau|_{n=d_{\mathrm{IP}_1}} &= V_{\tau,\mathrm{IP}_1} \\ V_\tau|_{n=d_{\mathrm{IP}_2}} &= V_{\tau,\mathrm{IP}_2} \end{aligned} \tag{8.14}$$

where V_{τ,IP_1} and V_{τ,IP_2} are the respective tangential velocities at the interpolation points, IP_1 and IP_2, measured with respect to the surface velocity. Using Eq. 8.13, the gradient of the tangential velocity in the normal direction can be determined as:

$$\frac{\partial V_\tau}{\partial n} = 2an + b \tag{8.15}$$

which implies that, at the wall,

$$\left.\frac{\partial V_\tau}{\partial n}\right|_{n=0} = b \tag{8.16}$$

The shear stress at the wall is therefore given by:

$$\tau_P = \mu_P \left.\frac{\partial V_\tau}{\partial n}\right|_{n=0} = \mu_P b \tag{8.17}$$

where μ_P is the molecular viscosity at the surface, which is either assumed as a constant for low-speed flows or calculated using the surface temperature and Sutherland's law (Wilcox 1994). The surface temperature in this case can be determined using a similar approach as outlined here for the reconstruction of pressure at the surface.

8.3 Results

Results are presented from simulations of compressible turbulent flows past a flat plate, a wedge-shaped micro-vortex generator (μVG) and for laminar flow past a NACA 0012 airfoil at 10° angle of attack. Further, comparisons of surface pressure and shear stress, and force coefficients with values from literature are also presented for the laminar flow test case. The flow and mesh details for the simulations are listed in Table 8.1.

Table 8.1 Flow and computation details

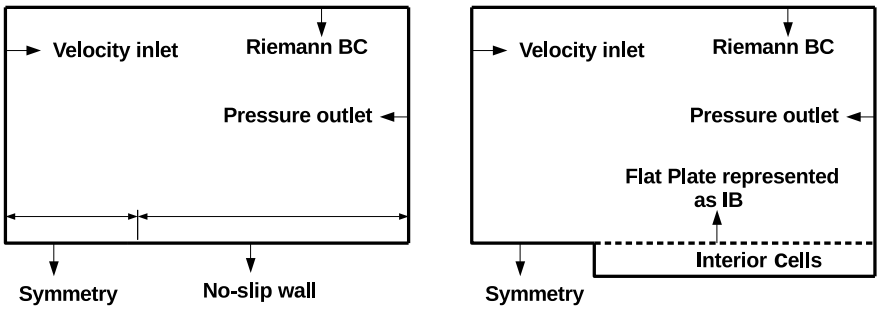
IB	Mach number	Re	$n_x \times n_y \times n_z$	k
Flat plate	0.2	5.0 million	$280 \times 192 \times 1$	1, 1/7
Micro VG	2.5	30 million/m	$747 \times 200 \times 90$	1/7
NACA 0012 airfoil	0.8	500	$800 \times 400 \times 1$	1

8.3.1 Flat-Plate Simulation

This test case is listed at https://turbmodels.larc.nasa.gov/flatplate.html for the verification of flow solvers developed for compressible turbulent flows. Table 8.1 lists the flow Mach number and Reynolds number, and the free-stream pressure and temperature are 1 atm and 300 K respectively. In this work, this case is simulated to compare the predictions of mean velocity and turbulence variables in turbulent boundary layers by the IBM outlined in this work with those obtained using a body-conforming grid.

The domains used for the simulation with a body-fitted grid and and immersed boundary approach are shown in Fig. 8.4; the boundary conditions used for the simulations are also indicated in this plot. As can be observed in Fig. 8.4b, in order to treat the flat plate as an immersed boundary, a sub-domain is created below it to ensure proper cell classification, which is required for the implementation of solution forcing. The extents of the (main) domain are from $X = -0.333330$ m to $X = 2.0$ m, and $Y = 0$ m to $Y = 1$ m. The grid for the body-fitted grid simulation has the same distribution in the wall-normal direction as the secondmost refined grid provided in the NASA archive. The grids for the IB simulation are obtained by starting with the mesh used for the body-fitted grid (and adding a sub-domain below the flat plate along its streamwise extent) and alternately removing points along both X and Y directions. The number of mesh cells in the grids used for the IB simulations along with the spacing at the wall are listed in Table 8.2, wherein L_0 is the finest mesh and L_4 is the coarsest mesh. The body-fitted grid and L_4 grid for the IB simulation are shown in Fig. 8.5. In Fig. 8.5, the flat plate is treated as an immersed boundary and is rendered as a point cloud. A power law of $k = 1$ is used for the IB simulations. The depth of the sub-domain (beneath the flat plate) used for the IB simulations is 0.05 m.

Profiles of normalised velocity, normalised turbulent kinetic energy and normalised turbulence frequency are plotted at $X = 0.7146$ m for the body-fitted grid and IB simulations in Fig. 8.6. A log scale is used for the wall-normal distance in all cases

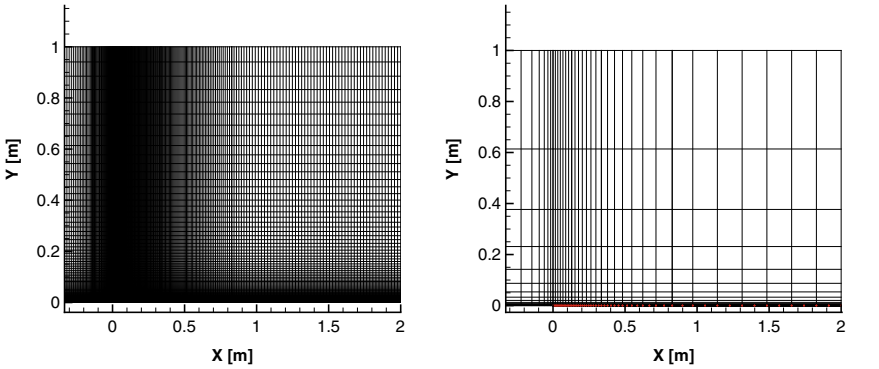


(a) Bodyfitted grid simulation. (b) IB simulation.

Fig. 8.4 Domains for simulation of Mach 0.2 turbulent flow past flat plate

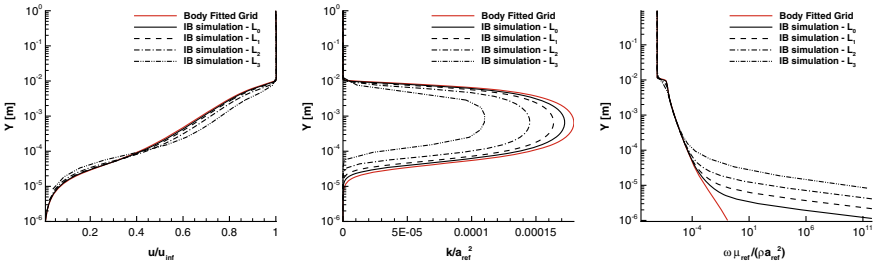
Table 8.2 Grid details for IB simulations of Mach 0.2 flow past flat plate

Grid level	$n_x \times n_y \times n_z$	$\Delta y_{\min}$ (m)	Average $\Delta y(m)$
L_0	$280 \times 192 \times 1$	1.0e−06	0.2
L_1	$140 \times 96 \times 1$	2.0e−06	0.4
L_2	$70 \times 48 \times 1$	4.0e−06	0.8
L_3	$35 \times 24 \times 1$	8.3e−06	1.6



(a) Bodyfitted grid simulation. (b) IB simulation: L_4; IB shown in red.

Fig. 8.5 Grids for simulation of Mach 0.2 turbulent flow past flat plate



(a) Velocity profile. (b) TKE profile. (c) Turbulence frequency profile.

Fig. 8.6 Comparison of results at $X = 0.7$ m for Mach 0.2 turbulent flow past a flat plate

and also for plotting the normalised turbulence frequency. Here, $u_{\text{inf}} = 69.68$ m/s is the free-stream velocity, $a_{\text{ref}} = 384.404$ m/s is the reference speed of sound and $\mu_{\text{ref}} = 1.6286\text{e}{-5}$ Ns/m^2 is the reference dynamic viscosity. It can be observed from Fig. 8.6a that the velocity profiles obtained using the IBM on the L_0 and L_1 grids, and even to an extent the L_2 grid, virtually coincide with that predicted by the body-fitted grid solution, whereas that obtained on the L_3 grid has discernible differences. A somewhat similar trend is observed for the normalised turbulence kinetic energy (TKE) profiles, wherein the maximum difference in the profiles predicted by the

IBM(L_0 grid) and the body-fitted grid do not exceed 5% of the value obtained from the body-fitted grid solution. The profiles of normalised turbulence frequency show strong differences as one approaches the wall. It is apparent from these comparisons that IBM presented here does not provide satisfactory solution of the near-wall turbulence frequency, even for very refined grids. However, the effect of this on the mean velocity profile appears minimal and considering the fact that the near-wall flow is modelled, the results are more than satisfactory. It is apparent from these set of computations, that when the near-wall grid is well resolved, with average Y^+ values of about unity, use of the IBM (Ghosh et al. 2010) with a power law of unity can give good results of mean flow properties.

In order to determine whether the use of a power law of 1/7 can mimic the energising effects of a turbulent boundary layer without sufficient near-wall grid resolution, the flat-plate simulation was also performed on a set of coarser grids. Grids L_4, L_5 and L_6 were designed such that the average Y^+ are 20, 40 and 80, respectively. Figure 8.7 shows the u-velocity profile obtained on grids L_4, L_5 and L_6, compared to that of the body-fitted grid simulation. It is seen that the u-velocity profile of the grid with Y^+ of 80 renders the closest match with that of the body-fitted grid simulation. In all cases though, the velocity profile shows a fullness characteristic of turbulent boundary layers, which is achieved without adequate near-wall grid resolution required to resolve the boundary layers.

8.3.2 Mach 2.5 Flow Past a Micro VG

A micro- or sub-boundary layer vortex generator is a device that can generate pairs of co- or counter-rotating vortices and has height much less than the local boundary layer thickness. The streamwise vortices bring in high-momentum fluid away from the wall to the near-wall region, producing fuller near-wall velocities in this process.

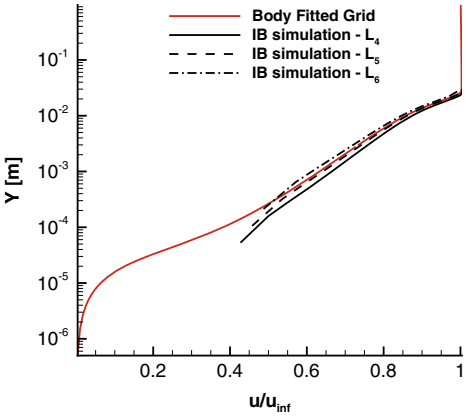


Fig. 8.7 Comparison of u-velocity profile at $X = 0.7$ m for power law $k = 1/7$

Experiments involving the effects of standard micro-VGs (Anderson et al. 2006) on a Mach 2.5 supersonic stream and an oblique shock–boundary layer interaction control have been performed at Cambridge University by Babinsky et al. (2009). The experiments included flow past single μVGs of size 3 and 6 mm, as well as arrays of μVGs. In the former case, the flow is without any additional shock generator, whereas the latter includes a shock generator that produces an oblique shock, which impinges on the boundary layer developing on the lower and sidewalls of the wind tunnel and causes separation. The flow details are mentioned in Table 8.1. The test section of the wind tunnel is 90 mm high and 110 mm wide.

In this chapter, we present results from numerical simulations of flow past single wedge-shaped micro-VGs placed in a Mach 2.5 supersonic stream (Ghosh et al. 2010; Sharma et al. 2016); the flow conditions are based on the experiments of Babinsky et al. (2009). The computational studies presented here render a part of the aforementioned test section (Babinsky et al. 2009) and is shown in Fig. 8.8 with the extents in streamwise, spanwise and wall-normal directions marked; the boundary conditions used for the simulation are also indicated in the figure. Two different VG types are presented here, a baseline or standard VG (Anderson et al. 2006) and a slotted VG (Sharma et al. 2016). The dimensions of the VGs are as prescribed by Anderson et al. (2006) and a schematic of baseline VG is shown in Fig. 8.9a, where $2W = 5.84$ h and $\phi = 66°$. The slotted VG, shown in Fig. 8.9b, additionally has a semi-circular slot running through its base. An advantage of the immersed boundary approach in this context is that the complexity of the IB geometry has no effect on the grid, which made it possible to use the same grid for the computations using both the baseline and slotted VGs. Results are first presented for flow past a single 6 mm baseline VG and then the for slotted vortex generator.

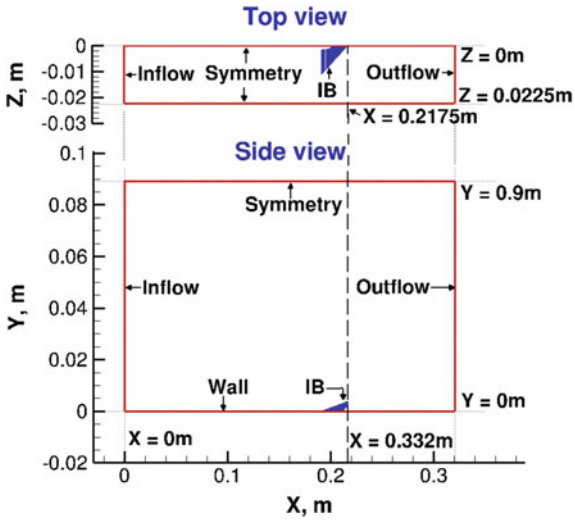
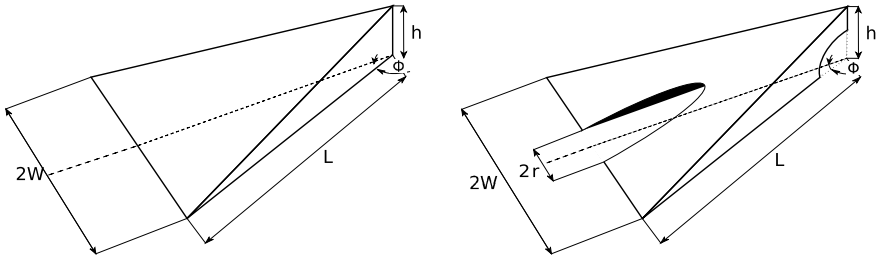


Fig. 8.8 Computational domain with boundary conditions for Mach 2.5 flow past a 6 mm VG. Reproduced from Sandhu et al. (2018) (copyright held by author Santanu Ghosh)



(a) Baseline VG (b) Slotted VG

Fig. 8.9 Isometric view of a vortex generator. Reproduced from Sandhu et al. (2018) (copyright held by author Santanu Ghosh)

The numerical simulations for flow past the 6 mm baseline VG were performed for both Reynolds-averaged Navier–Stokes (RANS) and hybrid large eddy simulation/Reynolds-averaged Navier–Stokes (LES/RANS) turbulence closures in this case and uses the immersed boundary method outlined in this work to render the vortex generator. A power law of 1/7 was used for the computations and both the approaches for determining the density in the band cell—using interpolation and solving the continuity equation—were used for the RANS simulations for this case. The hybrid LES/RANS simulation used the latter approach for calculating density in the band cell.

Figure 8.10 shows the formation of the counter-rotating vortex pair as the incoming flow moves past the baseline vortex generator. The streamlines in black are closer to the lower wall compared to those in red. It can be observed that as the flow moves over the upper surface of the VG and past its edges, it gets entrained in the low pressure wake behind the VG, forming a swirling flow that defines the primary counter-rotating vortex pair. The vortex pair entrains higher velocity flow from outer parts of the boundary layer and energises the flow near the wall, potentially making the resultant flow profile more resistant to separation.

Profiles of streamwise velocity are compared in Fig. 8.11 with experimental data obtained using laser Doppler anemometry (LDA) (Babinsky et al. 2009) at two streamwise stations downstream of the vortex generator, wherein the distances indicated are measured from the trailing edge of the VG; results from a body-fitted grid simulation is also included. The results show that the LES/RANS computations provide the most accurate results, wherein the energising effects of the streamwsie vortices are best captured. The results from the RANS calculations show that the IB simulations tend to under predict the energising effect of the streamwise vortices and over predict the wake effect—a consequence of the momentum sink effect introduced due to the IB rendition of the VG (Ghosh et al. 2010). In an overall sense, the results were satisfactory and showed promise for parametric studies using different VG geometries and layouts vis-a-vis flow control. The IB method has been subsequently used in RANS computations for the design (Sharma et al. 2016) and comparative analysis (Sandhu et al. 2018) of vortex generators.

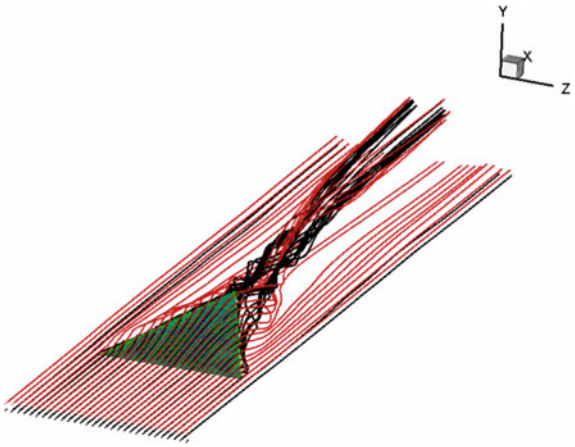


Fig. 8.10 Stream lines past a 6 mm VG in Mach 2.5 flow showing formation of streamwise counter-rotating vortices. Reproduced from Ghosh et al. (2008) with permission of Dr. Jack R. Edwards

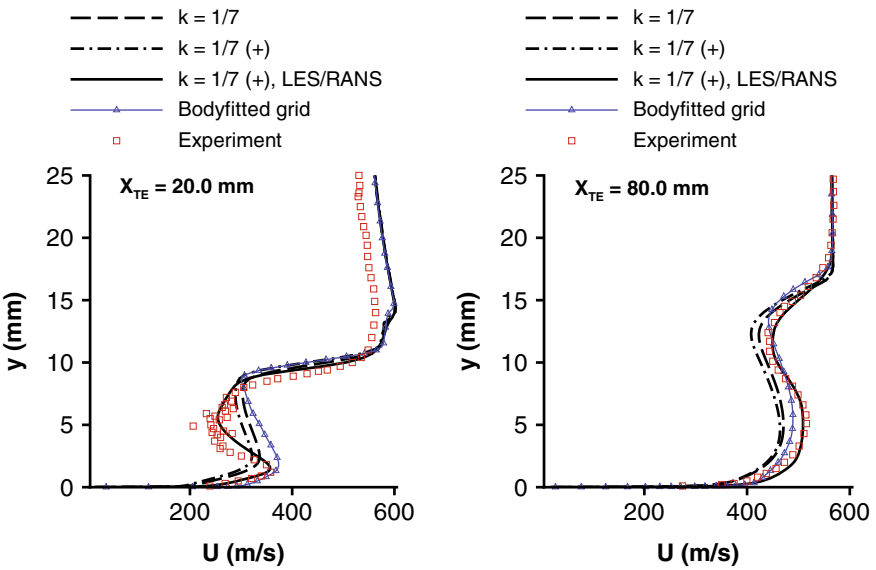


Fig. 8.11 Streamwise velocity profiles downstream of the VG trailing edge; experiment: Babinsky et al. (2009), (+): continuity equation integrated in band cells

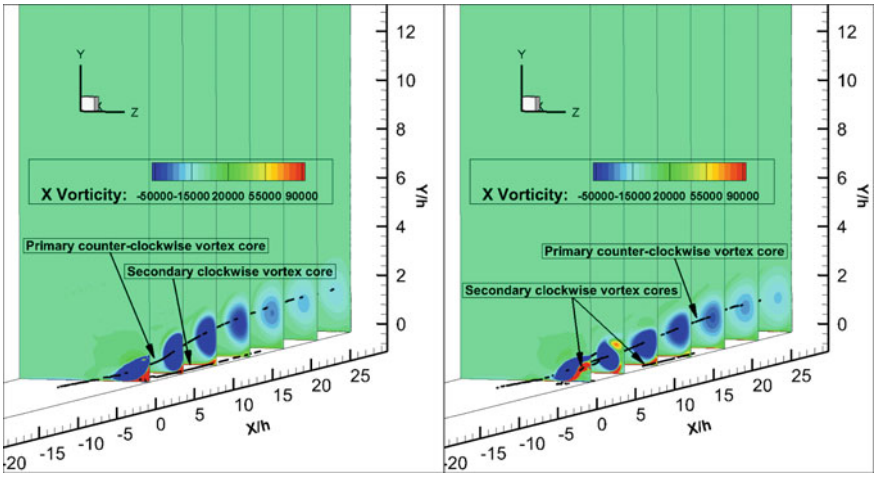


Fig. 8.12 Evolution of streamwise vortices past standard ramp-type VG (left) and slotted VG (right); vortex cores are shown in black lines. Reproduced from Sharma et al. (2016) (copyright held by author Santanu Ghosh)

The evolution of the flow downstream of a slotted and a baseline vortex generator, showing the vortex centres, is plotted in Fig. 8.12. It can be observed from the figure that the immersed boundary approach is capable of capturing both the primary and secondary vortices. Further, it can be seen that the lift-off of the primary vortex is reduced when the vortex generator has a slot.

The formation of the streamwise primary vortex pair has an energising effect on the near-wall flow, which results in fuller velocity profiles. Near-surface streamwise velocities downstream of the VG trailing edge are presented in Fig. 8.13 for the slotted and baseline vortex generators of different sizes. It is clear from this figure that the patches of the flow along the span show higher velocities, when compared to values upstream of the device leading edge, an effect of the induced flow entrainment by the primary vortices. This effect is stronger in the case of the slotted vortex generator.

8.3.3 Mach 0.8 Laminar Flow Past NACA 0012 Airfoil

The extents of the domain in this case are from $X = -10\,\text{m}$ to $X = 16.5\,\text{m}$ and $Y = -10\,\text{m}$ to $Y = 10\,\text{m}$ as shown in Fig. 8.14a. Figure 8.14b shows the Cartesian grid in which NACA 0012 airfoil is embedded as a cloud of points. The grid points are densely distributed within an inner box that almost circumscribes the airfoil. The grid is isotropic near the leading and trailing edges, with a spacing of 0.001 m, and stretches out as one moves away from the airfoil (immersed) surface.

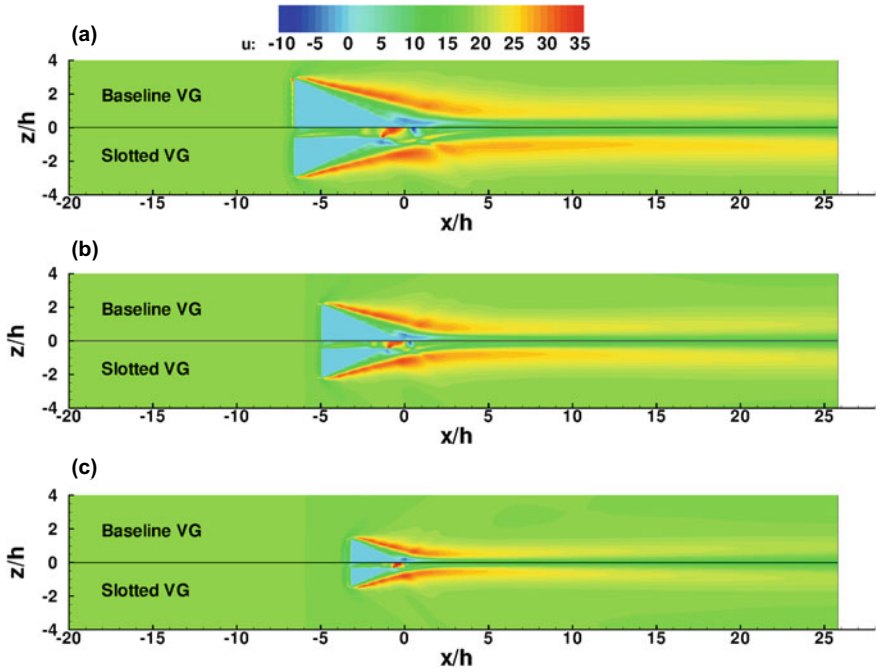


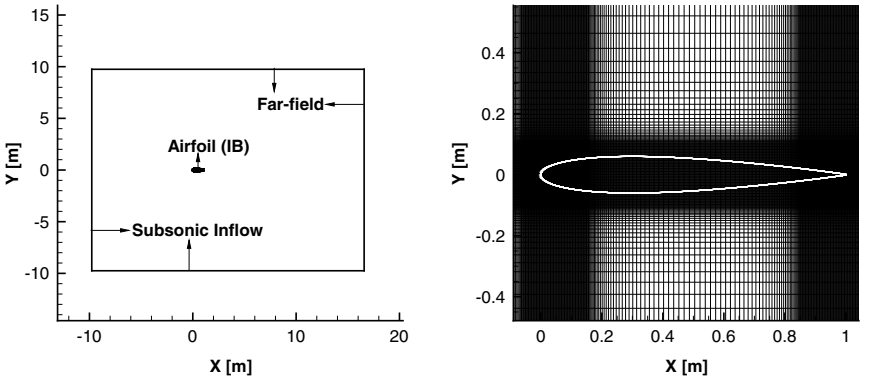
Fig. 8.13 Comparison of near-surface streamwise velocity contours for supersonic flow over a VG; contour plots for VGs with height **(a)** 4 mm, **(b)** 3 mm, and **(c)** 2 mm. $r = 0.6$ h (slotted VG). Reproduced from Sharma et al. (2016) (copyright held by author Santanu Ghosh)

The laminar flow at Mach 0.8 past NACA 0012 airfoil at 10° angle of attack is expected to produce a large separation bubble on the suction side of the airfoil as presented in Qiu et al. (2016) and also shown in Fig. 8.15.

Figure 8.16 shows the C_p plots obtained for three grids, wherein the coarse grids are obtained by removing alternate grid lines from the immediate refined grid. It can be observed that the noise in the reconstructed surface pressure reduces with grid refinement, as is common in most immersed boundary methods that impose a velocity boundary condition at the immersed surface and mentioned by Goza et al. (2016).

Figure 8.17 shows the C_p and C_f plots of the finest grid compared with literature (Qiu et al. 2016). The C_p and C_f plots are obtained by performing the pressure and shear stress reconstruction elaborated in Sect. 8.2.3.3 and 8.2.3.4.

Table 8.3 compares C_l and C_d with that available in literature. It is seen that there is a good agreement in the values.



(a) Domain extents and boundary conditions.

(b) Grid with immersed NACA 0012 airfoil.

Fig. 8.14 Domain and grid for Mach 0.8 laminar flow past NACA 0012 airfoil

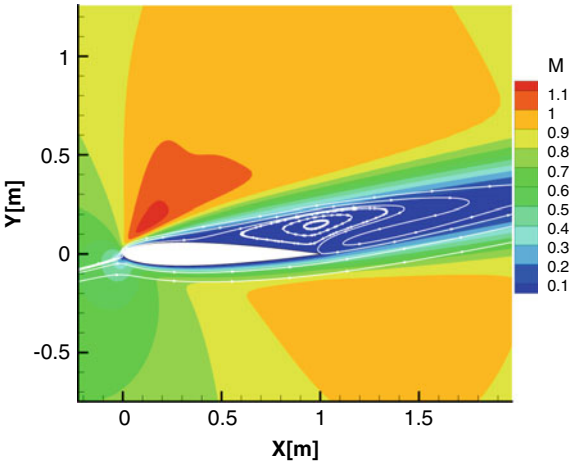


Fig. 8.15 Mach number contours for Mach 0.8 flow past NACA 0012 airfoil; streamlines show the large separation bubble on the airfoil

8.4 Conclusions

The work presented in this chapter discusses the detailed formulation of an immersed boundary method which was designed for compressible turbulent flows and presents results from test cases of varying complexity. Results using the immersed boundary method are compared to either simulations using a body-fitted grid or experimental data and show fair to good agreement. It is shown for the simple test case of turbulent flow past a flat plate that the power-law-based forcing of the solution in the immersed boundary method presented here can lead to near-wall velocity profiles that mimic the solution obtained using a body-fitted grid, with or without adequate grid resolution, by just suitably adjusting the power-law coefficient. Streamline patterns presented for flow past a wedge-shaped vortex generator and NACA 0012 airfoil shows that the

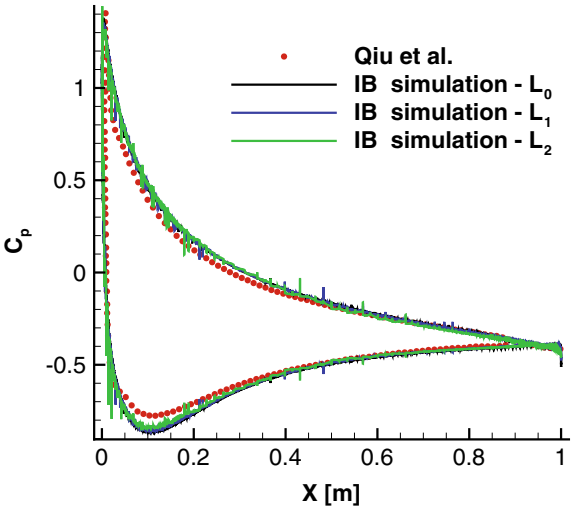
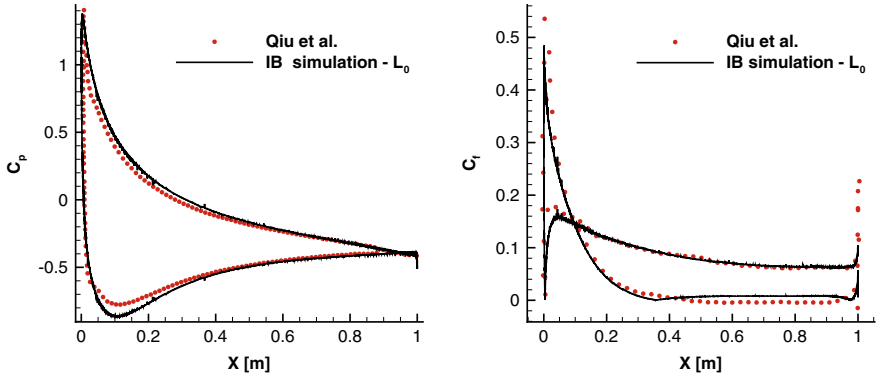


Fig. 8.16 Grid convergence showing C_p plots; C_p from literature Qiu et al. (2016) is also shown



(a) C_p plot. (b) C_f plot.

Fig. 8.17 Comparison of pressure and shear stress reconstructed at immersed surface with literature Qiu et al. (2016)

near-(immersed) surface flow obeys the no-penetration conditions satisfactorily and captures the flow physics. Additionally, a method based on inverse distance weights is discussed for the determination of pressure and shear stress on the immersed surface. The method reconstructs the pressure and surface-parallel velocity as functions of the coordinate normal to any point on the immersed surface, and uses values of pressure and velocity (component parallel to the local surface) interpolated at two points on the surface normal to determine the value of pressure and shear stress at the surface. The predictions of surface pressure and shear stress calculated using this approach for transonic flow past a NACA 0012 airfoil are compared with computational results

Table 8.3 Coefficient of lift and drag for Mach 0.8 flow past NACA 0012 airfoil

References	C_l	C_d
Jawahar and Kamath (2000)	0.49394	0.27216
Dervieux (2013)	0.4145–0.517	0.243–0.2868
Qiu et al. (2016)	0.4323	0.2822
Present	0.4687	0.2795

from literature and are shown to agree well. Integrated quantities of lift and drag are also presented and shown to compare favourably with values computed earlier in literature.

Acknowledgements The authors want to acknowledge Prof. Jack R. Edwards at North Carolina State University, Raleigh, USA, for granting permission to the authors to reuse his copyrighted material.

References

Anderson B, Tinapple J, Surber L (2006) Optimal control of shock wave turbulent boundary layer interactions using micro-array actuation. In: 3rd AIAA flow control conference, American Institute of Aeronautics and Astronautics, fluid dynamics and co-located conferences. https://doi.org/10.2514/6.2006-3197

Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1998) An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J ACM 45(6):891–923. https://doi.org/10.1145/293347.293348

Babinsky H, Li Y, Pitt Ford CW (2009) Microramp control of supersonic oblique shock-wave/boundary-layer interactions. AIAA J 47(3):668–675. https://doi.org/10.2514/1.38022

Bernardini M, Modesti D, Pirozzoli S (2016) On the suitability of the immersed boundary method for the simulation of high-Reynolds-number separated turbulent flows. Comput Fluids 130:84–93. https://doi.org/10.1016/j.compfluid.2016.02.018

Bharadwaj SA, Ghosh S (2018) Second-order interpolation techniques for accurate surface data estimation in immersed-boundary methods. In: 2018 applied aerodynamics conference, American Institute of Aeronautics and Astronautics, pp 1–15. https://doi.org/10.2514/6.2018-3331

Borazjani I, Ge L, Sotiropoulos F (2008) Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies. J Comput Phys 227(16):7587–7620. https://doi.org/10.1016/j.jcp.2008.04.028

Brehm C, Hader C, Fasel H (2015) A locally stabilized immersed boundary method for the compressible Navier-Stokes equations. J Comput Phys 295:475–504. https://doi.org/10.1016/j.jcp.2015.04.023

Capizzano F (2011) Turbulent wall model for immersed boundary methods. AIAA J 10(2514/1):J050466

Chaudhuri A, Hadjadj A, Chinnayya A (2011) On the use of immersed boundary methods for shock/obstacle interactions. J Comput Phys 230(5):1731–1748. https://doi.org/10.1016/j.jcp.2010.11.016. http://www.sciencedirect.com/science/article/pii/S0021999110006248

Choi J, Oberoi R, Edwards J, Rosati J (2007) An immersed boundary method for complex incompressible flows. J Comput Phys 224(2):757–784. https://doi.org/10.1016/j.jcp.2006.10.032

Choi JI, Edwards JR, Rosati JA, Eisner AD (2011) Large eddy simulation of particle re-suspension during a footstep. Aerosol Sci Technol. https://doi.org/10.1080/02786826.2011.631613
Colella P, Woodward PR (1984) The Piecewise Parabolic Method (PPM) for gas-dynamical simulations. J Comput Phys 54(1):174–201. https://doi.org/10.1016/0021-9991(84)90143-8
De Palma P, de Tullio M, Pascazio G, Napolitano M (2006) An immersed-boundary method for compressible viscous flows. Comput Fluids 35(7):693–702. https://doi.org/10.1016/j.compfluid.2006.01.004
de Tullio M, De Palma P, Iaccarino G, Pascazio G, Napolitano M (2007) An immersed boundary method for compressible flows using local grid refinement. J Comput Phys 225(2):2098–2117. https://doi.org/10.1016/j.jcp.2007.03.008
Dervieux A (2013) Numerical simulation of compressible Euler flows: a GAMM workshop, vol 26. Springer
Edwards JR (1997) A low-diffusion flux-splitting scheme for Navier-Stokes calculations. Comput Fluids 26(6):635–659. https://doi.org/10.1016/S0045-7930(97)00014-5
Fadlun E, Verzicco R, Orlandi P, Mohd-Yusof J (2000) Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161(1):35–60. https://doi.org/10.1006/jcph.2000.6484
Ghias R, Mittal R, Dong H (2007) A sharp interface immersed boundary method for compressible viscous flows. J Comput Phys 225(1):528–553. https://doi.org/10.1016/j.jcp.2006.12.007
Ghosh S (2010) An immersed boundary method for simulating the effects of control devices used in mitigating shock boundary layer interactions. PhD thesis
Ghosh S, Choi JI, Edwards J (2008) RANS and hybrid LES/RANS simulation of the effects of micro vortex generators using immersed boundary methods. In: 38th fluid dynamics conference and exhibit. https://doi.org/10.2514/6.2008-3728
Ghosh S, Choi JI, Edwards JR (2010) Numerical simulations of effects of micro vortex generators using immersed-boundary methods. AIAA J 48(1):92–103. https://doi.org/10.2514/1.40049
Goldstein D, Handler R, Sirovich L (1993) Modeling a no-slip flow boundary with an external force field. J Comput Phys 105(2):354–366. https://doi.org/10.1006/jcph.1993.1081
Goza A, Liska S, Morley B, Colonius T (2016) Accurate computation of surface stresses and forces with immersed boundary methods. J Comput Phys 321:860–873. https://doi.org/10.1016/j.jcp.2016.06.014
Jawahar P, Kamath H (2000) A high-resolution procedure for Euler and Navier-Stokes computations on unstructured grids. J Comput Phys 164(1):165–203. https://doi.org/10.1006/jcph.2000.6596
Menter FR (1994) Two-equation eddy-viscosity turbulence models for engineering applications. AIAA J 32(8):1598–1605. https://doi.org/10.2514/3.12149
Miller LA, Peskin CS (2009) Flexible clap and fling in tiny insect flight. J Exp Biol 212(19):3076–3090
Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10(2):252–271. https://doi.org/10.1016/0021-9991(72)90065-4
Qiu YL, Shu C, Wu J, Sun Y, Yang LM, Guo TQ (2016) A boundary condition-enforced immersed boundary method for compressible viscous flows. Comput Fluids 136:104–113. https://doi.org/10.1016/j.compfluid.2016.06.004
Roy CJ, Edwards JR (2000) Numerical simulation of a three-dimensional flame/shock wave interaction. AIAA J 38(5):745–754. https://doi.org/10.2514/2.1035
Roy S, Subramaniam K, Ghosh S (2017) Passive control of normal-shock-wave/boundary-layer interaction using porous medium: computational study. In: 35th AIAA applied aerodynamics conference. https://doi.org/10.2514/6.2017-3912
Sandhu JPS, Ghosh S, Subramanian S, Sharma P (2018) Evaluation of ramp-type micro vortex generators using swirl center tracking. AIAA J 56:1–11. https://doi.org/10.2514/1.J056796
Sharma P, Varma D, Ghosh S (2016) Novel vortex generator for mitigation of shock-induced flow separation. J Propul Power 32(5):1264–1274. https://doi.org/10.2514/1.B35962
Sotiropoulos F, Yang X (2014) Immersed boundary methods for simulating fluid-structure interaction. Prog Aerosp Sci 65:1–21. https://doi.org/10.1016/j.paerosci.2013.09.003

Spalart P, Allmaras S (1992) A one-equation turbulence model for aerodynamic flows. In: 30th AIAA aerospace sciences meeting and exhibit. https://doi.org/10.2514/6.1992-439

Tamaki Y, Harada M, Imamura T (2017) Near-wall modification of Spalart-Allmaras turbulence model for immersed boundary method. AIAA J 55(9):3027–3039. https://doi.org/10.2514/1.J055824

Tran PH, Plourde F (2014) Computing compressible internal flows by means of an Immersed Boundary Method. Comput Fluids 97:21–30. https://doi.org/10.1016/j.compfluid.2014.03.009

Tseng YH, Ferziger JH (2003) A ghost-cell immersed boundary method for flow in complex geometry. J Comput Phys 192(2):593–623. https://doi.org/10.1016/j.jcp.2003.07.024

Varma D, Ghosh S (2017) Flow control in Mach 4.0 inlet by slotted wedge-shaped vortex generators. J Propul Power 33(6):1428–1438. https://doi.org/10.2514/1.B36314

Wilcox DC (1994) Turbulence modeling for CFD. DCW Industries, Inc., CA

Yang J, Balaras E (2006) An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. J Comput Phys 215(1):12–40. https://doi.org/10.1016/j.jcp.2005.10.035

Zastawny M, Mallouppas G, Zhao F, van Wachem B (2012) Derivation of drag and lift force and torque coefficients for non-spherical particles in flows. Int J Multiph Flow 39:227–239. https://doi.org/10.1016/j.ijmultiphaseflow.2011.09.004

Zheng X, Xue Q, Mittal R, Beilamowicz S (2010) A coupled sharp-interface immersed boundary-finite-element method for flow-structure interaction with application to human phonation. J Biomech Eng 132(11):111003. https://doi.org/10.1115/1.4002587

Chapter 9
A Sharp-Interface Immersed Boundary Method for High-Speed Compressible Flows

Shuvayan Brahmachary, Ganesh Natarajan, Vinayak Kulkarni, and Niranjan Sahoo

9.1 Introduction

Numerical simulation of flow past bodies offers several challenges especially when the geometry is complex in nature. Ensuring a good quality body-conformal mesh for the underling geometry is not a trivial proposition and demands user expertise while also being time-consuming. The challenge of creating a good quality body-conformal mesh becomes even more severe when the geometries undergo motion. For such scenarios, Cartesian grid-based methods offer a significant advantage because of the ease with which the complex bodies can be treated on a simple orthogonal grid. This reduces the cost and time associated with grid generation and can be extended in a straightforward manner for moving body problems. The use of a fixed background grid provides the user with significant leverage by avoiding mesh movement and re-meshing that would be necessitated on flow solvers with conformal meshes. One of the approaches in this class is the "cut–cell" method that has been used quite extensively (Clarke et al. 1986; Udaykumar and Shyy 1995; Ye et al. 1999). However, this approach suffers from stability issues due to small cells near the boundary that limit the time step and requires cell-merging strategies to overcome this problem. Another class of Cartesian grid-based methods that have gained popularity in the last few years is the immersed boundary method, originally proposed by Peskin in his seminal work in 1972 (Peskin 1972). Over the last two decades, there have been several variants of the immersed boundary methods, although the development of these techniques for fluid flows and heat transfer has been largely for incompressible flows. A good and comprehensive review of IB methods can be found in Mittal and

S. Brahmachary · G. Natarajan (✉) · V. Kulkarni · N. Sahoo
Indian Institute of Technology Guwahati, Guwahati, Assam, India
e-mail: n.ganesh@iitpkd.ac.in

G. Natarajan
Indian Institute of Technology Palakkad, Palakkad, Kerala, India

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_9

Iaccarino (2005) and Sotiropoulos and Yang (2014), with the latter highlighting some of the interesting applications of IB techniques for complex incompressible flows.

The use of IB approaches for compressible flows has not been as widespread when compared to its incompressible counterparts. The earliest studies in this direction were carried out by Ghias et al. (2007) and de Palma et al. (2006). While the former discussed mostly low subsonic flows using ghost-cell IB methods, the latter extended the IB methodology to low supersonic flows. This was followed by the work on sharp-interface IB methods for transonic flows using local grid refinement by de Tullio et al. (2007). Cho et al. (2007) employed the Brinkman penalisation method, which belongs to the class of "diffuse" interface IB methods for compressible flows over a wide range of Mach numbers. Studies using a hybrid Cartesian immersed boundary (HCIB) method in the subsonic and transonic regimes were carried out by Zhang and Zhou (2014). Sambasivan and Udayakumar devised a sharp-interface variant for multi-material compressible flows (Sambasivan and UdayKumar 2010), and ghost-cell immersed boundary approaches have been employed to study shock diffraction and explosion with moving bodies (Mo et al. 2016). There have also been efforts to develop higher-order finite difference IB methods for compressible flow (Brehm et al. 2015) but most studies have been targeted at flows in the high subsonic and low supersonic flow regimes. Furthermore, while most studies have addressed Euler flows, only a few efforts concentrate on viscous compressible flows (Palma et al. 2006; de Tullio et al. 2007; Ghosh et al. 2010; Pu and Zhou 2018). Importantly, there have been only a handful of studies that have attempted to use the IB approach for high Mach number flows. In particular, the recent studies of Das et al. (2017) and Qu et al. (2018) have employed immersed boundary methods for shocked particle-laden flows and moving rigid bodies, respectively. Nevertheless, even these studies in high-speed viscous flows do not address the issue of resolution of the thin boundary layers and consequently the prediction of wall heat flux and skin friction. To the best of the authors' knowledge, only the studies of Arslanbekov et al. (2011) and Sekhar and Ruffin (2013) have attempted to study the stagnation heat flux estimation using IB methods. While their investigations discussed hypersonic flows, the Reynolds numbers in their studies were quite moderate. An important aspect of immersed boundary approaches that also has not been deeply probed is the issue of discrete conservation. Unlike cut-cell-based methods, IB approaches are clearly not discretely conservative and there have been no major efforts to look into the conservation errors particularly in compressible flows.

The literature survey presented herein, while not exhaustive, clearly points to the need to assess the class of immersed boundary methods for hypersonic laminar flows. This motivates our studies detailed in this chapter where we focus on the development of a sharp-interface immersed boundary method in the finite volume framework and discuss its ability to accurately compute hypersonic inviscid and laminar flows. We specifically discuss the aspects of discrete conservation as well as the efficacy of the IB approach for computing heat flux and skin friction distribution in viscous flows past canonical configurations. The remainder of the chapter is organised as follows. Sections 9.2 and 9.3 describe the numerical framework in details. The investigations

pertaining to conservation errors, Euler flows as well as viscous flow computations form the subject matter of Sect. 9.4. We summarise the salient findings from the present study in Sect. 9.5, where a few recommendations for future research are also outlined.

9.2 Finite Volume Solver for Compressible Flows

In this section, we describe the details of the finite volume flow solver which forms the basic workhorse of the numerical investigations detailed later in this chapter. The immersed boundary method, to be discussed in the following section, is integrated with this flow solver. Based on an unstructured data framework, the flow solver solves the Navier–Stokes equations for a perfect gas which in the conservative form (in two dimensions) reads,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F_I}}{\partial x} + \frac{\partial \mathbf{G_I}}{\partial y} + \frac{\partial \mathbf{F_V}}{\partial x} + \frac{\partial \mathbf{G_V}}{\partial y} + \alpha(\mathbf{S_I} - \mathbf{S_V}) = 0$$

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}; \quad \mathbf{F_I} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho u\left(E + \frac{p}{\rho}\right) \end{bmatrix}; \quad \mathbf{G_I} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho v\left(E + \frac{p}{\rho}\right) \end{bmatrix}$$

$$\mathbf{F_V} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x \end{bmatrix}; \quad \mathbf{G_V} = \begin{bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y \end{bmatrix}$$

where the vectors $\mathbf{S_I}$ and $\mathbf{S_V}$ are the inviscid and viscous source terms, respectively, and are relevant only for axisymmetric flows. These may be expressed as,

$$\mathbf{S_I} = \frac{1}{y} \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ \rho v(E + \frac{p}{\rho}) \end{bmatrix}$$

$$\mathbf{S_V} = \frac{1}{y} \begin{bmatrix} 0 \\ \tau_{xy} - \frac{2}{3} y \frac{\partial}{\partial x}(\mu v/y) \\ \tau_{yy} - \tau_{\theta\theta} - \frac{2v}{3y}\mu - \frac{2}{3} y \frac{\partial}{\partial y}(\mu v/y) \\ u\tau_{xy} + v\tau_{yy} - q_y - \frac{2\mu v^2}{3y} - \frac{2y}{3}\frac{\partial}{\partial y}(\mu v^2/y) - \frac{2y}{3}\frac{\partial}{\partial x}(\mu uv/y) \end{bmatrix}$$

where,

$$\tau_{\theta\theta} = \mu\Big(-\frac{2}{3}\Big(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\Big) + \frac{4v}{3y}\Big)$$

We set $\alpha = 1$ for axisymmetric simulations while for planar two-dimensional studies $\alpha = 0$. Here, U represents the vector of conserved variables (mass, momentum and energy) and F_I and G_I represent the inviscid fluxes while F_V and G_V represent the viscous fluxes. The components of heat flux are represented by q_x and q_y while τ_{xx}, τ_{yy} and τ_{xy} are the components of the symmetric viscous stress tensor. Integrating these vector conservation laws over an arbitrary control volume and applying the Gauss divergence theorem yield the semi-discrete form of the governing equations. The semi-discrete form of the conservation laws in a compact form reads,

$$\frac{d\mathbf{U}_i}{dt} = -\frac{1}{\Omega_i}\sum_{J\epsilon i}\mathbf{H}_{\perp_J}\Delta S_J - \alpha\overline{\mathbf{S}}_i = \mathbf{R}(\overline{U}_i)$$

$$\mathbf{H}_\perp = \begin{bmatrix} \rho u_\perp \\ \rho u u_\perp + p n_x \\ \rho v u_\perp + p n_y \\ (\rho e + p) u_\perp \end{bmatrix} - \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} \\ n_x\tau_{yx} + n_y\tau_{yy} \\ n_x\theta_x + n_y\theta_y \end{bmatrix}; \quad \overline{\mathbf{S}} = \overline{\mathbf{S}}_\mathbf{I} - \overline{\mathbf{S}}_\mathbf{V}$$

where $u_\perp = u n_x + v n_y$, $\theta_x = u\tau_{xx} + v\tau_{xy} - q_x$, $\theta_y = u\tau_{yx} + v\tau_{yy} - q_y$.

The quantity Ω_i is the volume of the ith cell, ΔS is the face area and n_x and n_y are the components of the outward unit normal to the face. The summation is over all faces J of a cell i, and the convective and viscous fluxes at a face are evaluated at the face mid-points using a single-point Gauss quadrature. A second-order accurate linear reconstruction proposed by Barth and Jesperson (2001) is employed to determine the states required for convective flux computations. The convective fluxes are determined using AUSM scheme (Liuo and Steffen 1993), unless otherwise specified and the Venkatakrishnan limiter (Blazek 2001) is used to ensure monotonicity of the solution. Green–Gauss reconstruction (Blazek 2001) is employed to determine the gradients required for viscous flux computations. Time advancement is realised using a five-stage Runge–Kutta scheme (Blazek 2001), although a single-stage RK scheme (equivalent to explicit Euler) is employed for steady flow computations where temporal accuracy is unimportant. This finite volume (FV) solver has been extensively validated in previous work on several problems involving inviscid and viscous compressible flows (John and Kulkarni 2014). The flow solver, being based on unstructured data, is also capable of handling adaptive grids which are constructed by an isotropic refinement strategy (Natarajan 2009) that divides each "parent" cell into four "children" with the regions where adaptation is effected identified by the user apriori.

9.3 Hybrid Cartesian Immersed Boundary Method

The details and implementation of the sharp-interface hybrid Cartesian immersed boundary (HCIB) method are discussed in this section. The HCIB approach was first proposed by Gilmanov and Sotiropoulos (2005) for incompressible fluid flows. The present work is an extension of their methodology to compressible inviscid and viscous flows. In the technique, the solid body is immersed into an underlying Cartesian mesh. The solid boundary is discretised using linear line segments in two dimensions (or surface triangulated in three dimensions). Unlike traditional body-conformal CFD solvers, the mesh does not conform to the geometry, and therefore, the accurate calculation of the near-wall numerical solution is critical. The HCIB approach has been implemented in the finite volume framework described previously in Sect. 9.2 and constitutes two distinct stages that are described below.

9.3.1 Cell Classification

The first stage in the HCIB approach is the classification of the cells (or control volumes) of the underlying Cartesian mesh into three categories. Cells whose cell centres lie inside the solid are classified as solid cells (denoted as S) while the remaining cells are termed as fluid cells (denoted as F). This is effected using a ray-casting algorithm. All F cells which share at least one face with a S cell are then termed as immersed cells (denoted by I cells). The procedure behind this classification step is shown in Fig. 9.1 and distinguishes the regions where the solution is sought (F and I cells) from those where the solution is not necessary (S cells). For stationary cases, it is easy to see that the classification is a one-time affair whereas for moving body problems, it must be repeated at every time step.

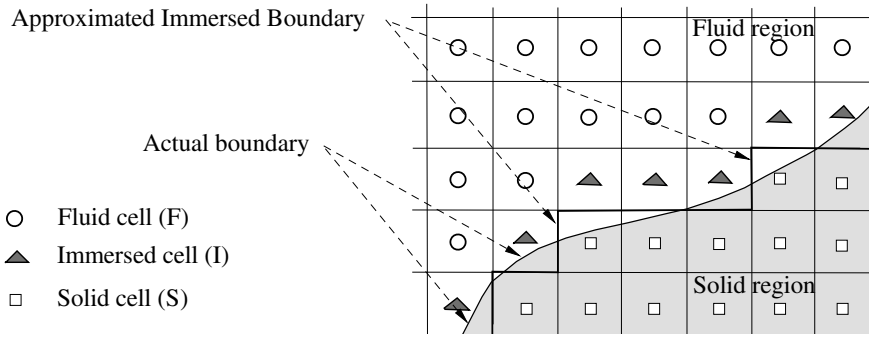


Fig. 9.1 Classification of cells

9.3.2 Solution Reconstruction

The second stage in the HCIB approach involves solution reconstruction where the numerical solution in the near vicinity of the solid body is obtained by enforcing the boundary conditions while preserving the sharp interface of the geometry. The numerical solution needs to be only computed in the F and I cells with those in the F cells obtained by solving the Navier–Stokes equations. The solution in I cells is however obtained using some form of algebraic reconstruction as detailed in this section.

We now describe the solution reconstruction for viscous compressible flows for geometries with adiabatic as well as isothermal walls. The solution reconstruction is an interpolation along the direction locally normal to the interface as shown in Fig. 9.2. The boundary conditions are enforced directly at the sharp interface in this study. To do so, we first identify the nearest face on the solid boundary for each I cell. Following this, we identify points b and f on the body surface and in the fluid domain, respectively, which lie along the normal $\hat{\mathrm{n}}$ to the nearest face and also contain the centroid of the I cell. The point b is the intersection point of the local normal with the geometric boundary while point f is the closest point on this line that cuts a connector joining two F cells (see Fig. 9.2). The boundary condition on the body surface is used to determine the primitive variables at b while we adopt linear interpolation of the solutions at F_1 and F_2 to evaluate the fluid properties at f.

$$\phi_{\mathrm{f}} = \frac{\phi_{\mathrm{F1}} d_2 + \phi_{\mathrm{F2}} d_1}{d_1 + d_2} \tag{9.1}$$

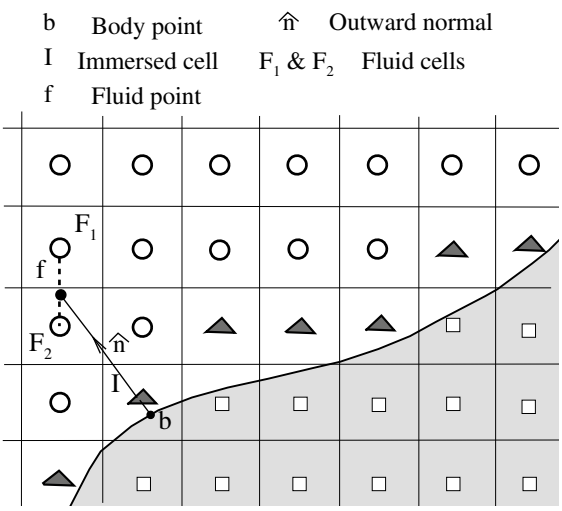


Fig. 9.2 Reconstruction for obtaining ϕ at immersed cells

where d_1 and d_2 refer to distances of the point f from the centroid of cells F_1 and F_2. The primitive variables at the I cells are then obtained by a suitable interpolation at b and f points. The choice of this reconstruction can be either a polynomial or non-polynomial interpolation, and it need not necessarily be identical for all primitive variables. The specific details of this reconstruction strategy when both isothermal and adiabatic surfaces are involved are now discussed.

Assuming that the solution varies linearly and denoting the variable of interest as ϕ, one can write its variation along the normal direction as,

$$\phi = C_1 r + C_2 \tag{9.2}$$

where r is the distance measured from the b point on the sharp interface along the direction of the outward local normal. Subsequently, if the unknowns C_1 and C_2 can be uniquely determined from two independent conditions then the value at the I cell may be computed as,

$$\phi_I = C_1 r_{bI} + C_2 \tag{9.3}$$

The constants C_1 and C_2 are typically evaluated using the boundary conditions at "b" and the interpolated solution at "f".

9.3.3 Reconstruction for Velocities

For viscous flows, the solid wall satisfies both the no-slip as well as the impermeable wall boundary condition, i.e. $u_{||b} = 0$ and $u_{\perp b} = 0$, respectively. The quantities $u_{||}$ and $u_{\perp}$ denote the components of velocity vector along the local tangential and local normal directions, respectively, and may be determined as,

$$u_{||} = u_f n_y - v_f n_x \tag{9.4}$$

$$u_{\perp} = u_f n_x + v_f n_y \tag{9.5}$$

where n_x and n_y now refer to the components of the normal to the interface along which the one-dimensional solution reconstruction is effected. The values of these velocity components at the f point may be computed using Eq. (9.1) where ϕ is chosen as $u_{||}$ or $u_{\perp}$. The use of the known values at b and f points helps to compute the values of $u_{||}$ and $u_{\perp}$ at the I cells using Eq. (9.3) and one can obtain the Cartesian velocity components at the cell centres using the reverse transformation that reads,

$$u_I = (u_{||I}\, n_y) - (u_{\perp I}\, n_x) \tag{9.6}$$

$$v_I = (-u_{||I}\, n_x) - (u_{\perp I}\, n_y) \tag{9.7}$$

For moving bodies whose motion is induced by the flow, the velocity components of the body are nonzero and are evaluated by solving the second-order ODEs describing Newton's second law of motion. This gives,

$$\begin{aligned} u_b^{k+1} &= u_b^k + \frac{\Delta t}{M_b} F_x^k \\ v_b^{k+1} &= v_b^k + \frac{\Delta t}{M_b} F_y^k \end{aligned} \tag{9.8}$$

where k and $k+1$ represent the present and next time step, respectively, and M_b is the mass of the body. The terms (F_x, F_y) represent the force components and are determined from the wall pressure and wall shear stresses. We choose the contour of integration for this purpose as the approximated immersed boundary as highlighted in Fig. 9.1, which leads to an approximated domain stair-step representation as also in Mizuno et al. (2015).

9.3.4 Reconstruction for Pressure

The pressure at the cell I is obtained by invoking the boundary layer approximation that strictly holds for non-separated flows. This gives $\frac{\partial p}{\partial r} = 0$, and as a result, one can impose the pressure outside the boundary layer on the surface. We thus have $p_b = p_I = p_f$, with p_f obtained using Eq. (9.1). While this condition would be strictly valid only for thin layers and unseparated flows, they have been used for inviscid flows (Brahmachary et al. 2018) and appear to work even in scenarios with flow separation (as shall be shown in studies in Sect. 9.4.6).

9.3.5 Reconstruction for Temperature

The value of temperature at the point b depends on whether the wall boundary is adiabatic or isothermal. For isothermal walls, the wall temperature T_w is known and is constant which results in $T_b = T_w$. The temperature at the immersed cell T_I can then be obtained using Eq. 9.3. It is also possible to employ nonlinear and non-polynomial interpolation (Ghosh et al. 2010) although this has not been considered in the present study.

9.3.6 Reconstruction for Density

The density at I cells follows from the equation of state (EOS) for a perfect gas and is computed as,

$$\rho_I = \frac{p_I}{RT_I} \tag{9.9}$$

where R is the gas constant and p_I as well as T_I follow from suitable reconstruction approaches as described in Sects. 9.3.4 and 9.3.5. This methodology of one-dimensional solution reconstruction along the local normal direction closely resembles the extended extrapolation technique in Zhao et al. (2010) except that the normal velocity varies linearly in our approach as opposed to the quadratic variation in the extended extrapolation method.

9.3.7 *Calculation of Wall Pressure, Shear Stress and Heat Flux*

For viscous computations, the major quantities of interest are the surface distribution of pressure, heat transfer and viscous stresses. We provide a concise description of the computation of these parameters in the IB-FV solver to enable reproducibility of results in the following sections.

The pressure at the wall, by virtue of the homogeneous Neumann BC, is $p_w = p_I = p_f$ and may also be quantified using the coefficient of pressure defined by,

$$C_p = (p_w - p_\infty)/(0.5\rho_\infty V_\infty^2)$$

The wall heat flux is related to the temperature gradients at the wall and is defined as,

$$q_w = k_w \frac{\partial T}{\partial r}\bigg|_w$$

where k_w is the fluid thermal conductivity at the wall and is a function of the wall temperature T_b (obtained from Sutherland's law). Simple finite differencing may be employed for linear solution variation of temperature gradient in the near-wall region. The heat flux on the surface then follows as,

$$q_w = k_w \frac{T_I - T_w}{r_{bI}}$$

A non-dimensional wall heat flux may also be defined in terms of Stanton number (St),

$$St = q_w/(0.5\rho_\infty V_\infty c_p(T_o - T_w))$$

where T_o is the total temperature of the freestream and c_p is the constant pressure specific heat capacity.

Like the wall heat flux, the wall shear stress is a scalar quantity whose distribution over the surface is estimated to obtain the viscous drag acting on the body. The wall

shear stress is obtained by adding up all components of the viscous force along the local tangential direction as,

$$\tau_w = (\tau_{xx} n_x + \tau_{xy} n_y) t_x + (\tau_{xy} n_x + \tau_{yy} n_y) t_y$$

where $t_x = n_y$ and $t_y = -n_x$ denote the components of the local tangent vector and n_x and n_y are the components of the local normal to the surface. The conventional representation is to use the non-dimensional wall shear stress, which is referred to as the skin friction coefficient (or simply skin friction), C_f defined as,

$$C_f = \tau_w / (0.5 \rho_\infty V_\infty^2)$$

The computation of C_f warrants the estimation of viscous stress components at the body faces. These may be obtained by first identifying a neighbourhood of F cells associated with each b point (and therefore a unique I cell) and then using an inverse distance weighted averaging of the velocity gradients in these F cells to obtain an estimate at the b point [for further details refer to Brahmachary et al. (2018)]. These estimates are consequently employed to calculate the viscous stresses.

9.3.8 Reconstruction for Euler Flows

It is important to highlight the differences and/or simplifications in the reconstruction approach discussed herein for compressible inviscid flows. In case of Euler flows, the only boundary condition available at b is $u_{\perp b} = 0$ while the values for all remaining quantities ($u_{||}$, p_b and ρ_b) and their gradients are obtained using inverse distance weighting (IDW) as described in Eqs. (9.10) and (9.11),

$$\phi_b = \frac{\sum_{i=1}^{i=n} w_i \phi_i}{\sum_{i=1}^{i=n} w_i} \tag{9.10}$$

$$\nabla \phi_b = \frac{\sum_{i=1}^{i=n} w_i \nabla \phi_i}{\sum_{i=1}^{i=n} w_i} \tag{9.11}$$

where $w_i = 1/|d_i|$, $|d_i|$ being the distance between the centroids of the ith neighbour (which is a F cell) and b. Here, n represents the total number of node-sharing cells in the neighbourhood of the immersed cell.

In the following section, we shall explore the efficacy and versatility of the IB-FV solver that employs this HCIB strategy in an unstructured finite volume framework to estimate aerodynamic forces and heat loads in high-speed flows.

9.4 Numerical Investigations

This section is devoted to the numerical studies using the IB-FV solver for compressible inviscid and viscous flows. The importance of the interpolation strategy on the accuracy of the solution is investigated through a number of test cases involving simple moving bodies and complex stationary geometries on canonical problems in inviscid and laminar hypersonic regimes. We discuss the role of solution reconstruction on conservation errors in compressible flows. One of the critical issues related to non-conformal approaches is that of discrete conservation. Due to the non-conservative nature of the solution reconstruction performed in the I cells (as opposed to the solutions obtained at F cells by solving discrete form of the conservation laws), there is an inherent lack of conservation in IB approaches. This issue has not been addressed at length in the compressible flow regime, and we probe this aspect using two different test problems.

9.4.1 Transonic Flow Past Bump

The first test case is the transonic flow past a 10% thick bump. The computational domain is 3×1, and the associated boundary conditions are shown in Fig. 9.3. The inflow is at a freestream Mach number $M_\infty = 0.675$ while the pressure at the outlet is fixed, $P_{out} = 0.737$. This test case Ni (1982) leads to a normal standing shock nearly three quarters from the leading edge of the bump. We carry out simulations on four meshes, viz. 150×50, 225×75, 300×100 and 450×150 using the IB-FV solver. Computations have also been performed using a FV solver on body-fitted meshes of equivalent grid resolution. In Fig. 9.4a–d, we show the comparison of the solutions obtained using the non-conformal IB-FV solver along with the body-conformal FV solver. One can clearly observe that not only is the shock diffused in the coarsest grid, its location has also been inaccurately estimated with the IB-FV solver. With grid refinement, the shock location approaches that estimated using the FV solver, which is indeed conservative. One can notice that the shock location from all FV solutions is same (except the shock is diffused on coarser grids). We also see the differences between the two solvers from the pressure distributions in Fig. 9.5a and b where the average shock location using IB-FV approach depends on the grid resolution and agrees with those estimated by the FV solver only on the finest mesh and the computed results from Luo et al. (2006).

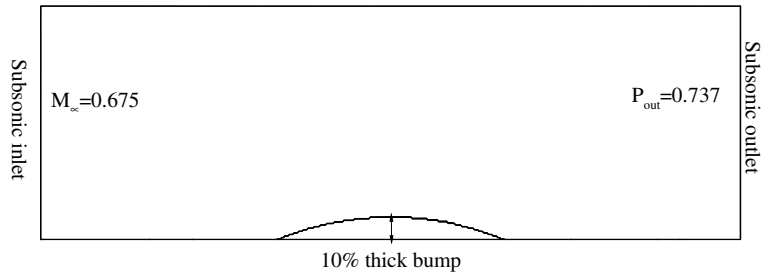


Fig. 9.3 Computational domain for transonic flow past bump along with boundary conditions

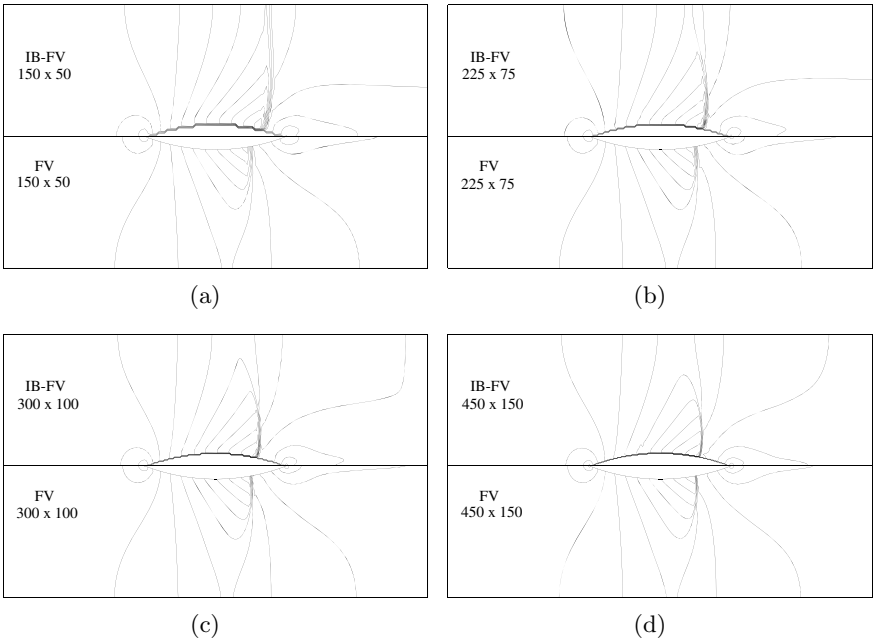


Fig. 9.4 Mach contours depicting normal standing shock for different mesh resolutions **a** 150 × 50, **b** 225 × 75, **c** 300 × 100 and **d** 450 × 150 (Min: 0.1, Δ: 0.1, Max: 1.5) (Top: IB-FV solver; Bottom: FV solver on body-fitted mesh)

These observations confirm that the solutions obtained using the IB-FV solver are not discretely conservative but the conservation errors diminish with grid refinement. We further investigate these discrete conservation errors by considering the supersonic flow past a wedge.

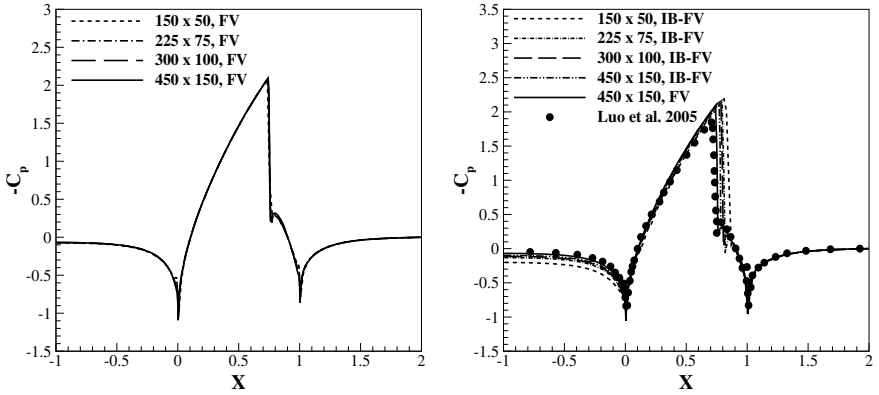


Fig. 9.5 Pressure distribution on the surface of the body from **a** FV solver on conformal grid and **b** IB-FV solver on non-conformal grid and its comparison with Luo et al. (2006)

9.4.2 Supersonic Flow Past Wedge

We numerically simulate the supersonic flow past a wedge with a semi-vertex angle of 20° and a freestream Mach number $M_\infty = 2$. This configuration is however not aligned with the underlying Cartesian mesh nor is the oblique shock that is the flow feature of interest. For this test problem as well, we simulate the flow on four different meshes in a domain of size 0.1×0.15. The details of the mesh are provided in Table 9.1. We compute the mass defect on every mesh as,

$$\Delta m = \sum \rho_f \, U_{\perp,f} \, \Delta S_f \tag{9.12}$$

where the summation is over all the boundary faces of the domain as well as the interior faces shared by S and I cells, i.e. the summation is performed along the stair-step representation in Fig. 9.1. For the FV solver on body-fitted meshes, the mass defect would be of the order of the steady-state residuals, which is a result of the discrete mass conservation. Since the IB-FV framework has been shown to be not discretely conservative, we expect a finite mass defect larger than the steady-state

Table 9.1 Mass defect Δm on different grids

Grid	Characteristic length scale, h	Mass defect, Δm
100×150	1/100	0.115
150×225	1/150	0.075
200×300	1/200	0.057
250×375	1/250	0.045

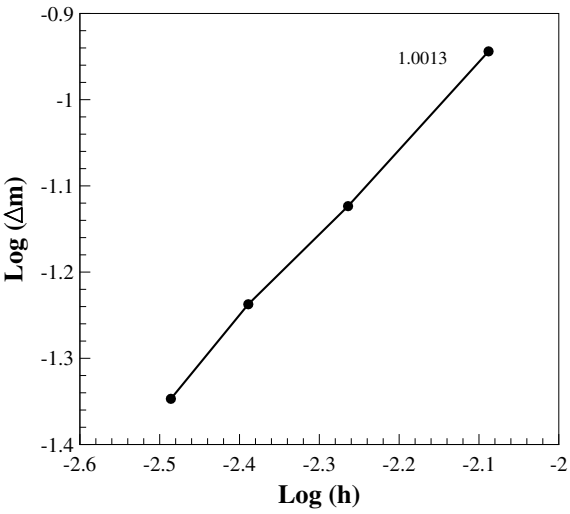


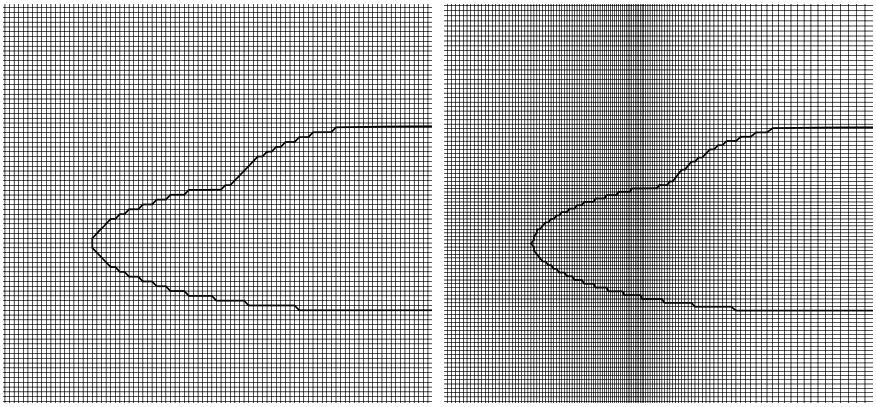
Fig. 9.6 Variation of mass defect Δm with grid refinement

residuals on every mesh. This is confirmed in Table 9.1 which shows the mass defect on the four different meshes. The errors are three orders of magnitude higher than the residuals, but decrease as the grid is refined. From Fig. 9.6, one can see that the mass error falls at a rate close to unity i.e. mass defect fall linearly with grid refinement. Thus, one can remark that there is a finite $\mathbf{O}(\mathrm{h})$ conservation error for the IB-FV approach similar to the observations made for a class of mesh-free methods (Sridar and Balakrishnan 2003) and the framework is strictly conservative only in the limit as grid spacing tends to zero.

9.4.3 Hypersonic Flow Past Double Ellipse

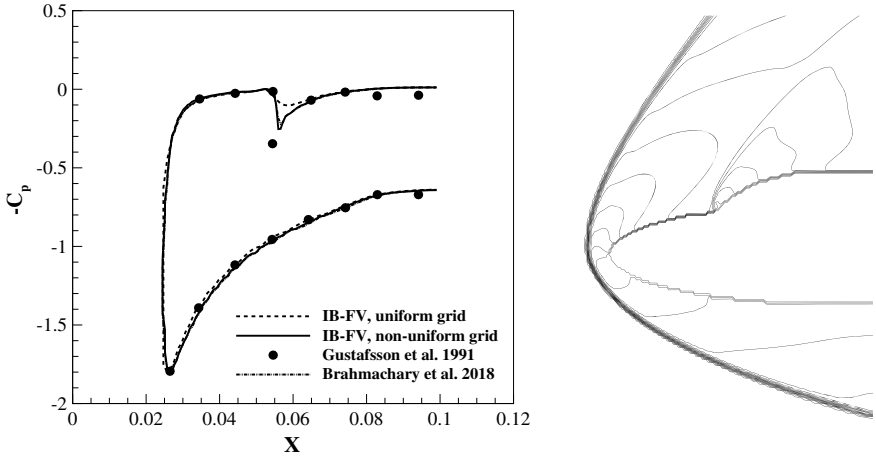
To demonstrate the ability of the IB-FV solver in handling hypersonic Euler flows past complex configuration, we study the high-speed high angle-of-attack flow past a double-ellipse configuration (Gustaffson et al. 1991). The freestream Mach number is $\mathrm{M}_\infty = 8.15$ and the angle of attack of 30° makes it challenging to accurately simulate the flow as well as estimate the surface pressure distribution. The double ellipse is immersed into two different Cartesian meshes with a total number of cells $\mathrm{n_c}$ = 36,000 in a computational domain of $[-0.1, 0.1] \times [-0.1, 0.1]$—while one employs a uniform grid (see Fig. 9.7a) with equal grid spacing the other utilises a non-uniform spacing (see Fig. 9.7b) which is finer near the canopy region.

The significance of the mesh in the context of non-conformal IB-FV solver can be understood from Fig. 9.8a where the surface distribution of pressure coefficient $\mathrm{C_p}$ is shown for both the uniform and non-uniform meshes. It can be observed that while results computed on both these meshes yield an overall fair agreement with the



(a) (b)

Fig. 9.7 **a** Uniform, **b** non-uniform Cartesian grid employed in IB-FV solver for flow over double ellipse



(a) (b)

Fig. 9.8 **a** Pressure distribution and **b** Mach contours for hypersonic flow past double ellipse (Min: 0, Δ: 0.5, Max: 8.15)

numerical data of Gustafsson et al. (1991), the uniform mesh is not very accurate in resolving the pressure jump across the weak canopy shock. The non-uniform mesh clustering however accurately captures the weak canopy shock and hints towards the need for selective mesh refinement in regions where sharp gradients in flow are likely. Figure 9.8b depicts the Mach contours that show the strong detached bow shock as well as the weak canopy shock. We can, therefore, remark that the IB-FV solver is capable of accurately computing flows with sharp flow gradients but necessitates sufficient local mesh resolution to also resolve the complex geometries.

9.4.4 Cylinder Lift-Off

To highlight the applicability of the IB-FV solver for moving body problems, we simulate the supersonic flow past a cylinder initially at rest but "lifts-off" due to the aerodynamic forces. The test case consists of a rectangular domain of size 0.1×0.2 with a cylinder of radius 0.05 m and density $\rho_c = 10.77\,\text{kg/m}^3$ placed at the bottom wall (0.15, 0.05), as shown in Fig. 9.9. The pre-shock condition is maintained as ρ=1.4 kg/m^3 and $p = 1$ Pa, whereas the left boundary is assigned as post-shock state. The cylinder is considered rigid and the influence of gravity is neglected and we compare our solutions with available numerical results in the literature that make the same assumptions. Studies were conducted on uniform Cartesian grids whose details are provided in Table 9.2 which also contains the position of the centre of mass of the cylinder at a final time of $t \sim 0.3$s. It can be observed that while there are differences in the position of the cylinder when compared with the results in Arienti et al. (2003), these differences tend to diminish with increasing mesh resolution. These differences can also be attributed to the fact that the reconstruction strategy in the I cells in the present work are different from those in Arienti et al. (2003). Figure 9.10 shows the pressure contours at two different time instances which clearly depict the complex unsteady flow phenomena. We also compare the time history of the trajectory of the cylinder with those computed by Sambasivan and UdayKumar (2009) in Fig. 9.11 and a good agreement in the results is further proof of the ability of the proposed IB-FV framework in computing high-speed Euler flows accurately.

In the test case to follow, we shall investigate the efficacy of the present IB-FV flow solver for scenarios involving laminar hypersonic flows with an emphasis on the accurate prediction of wall heat flux and skin friction.

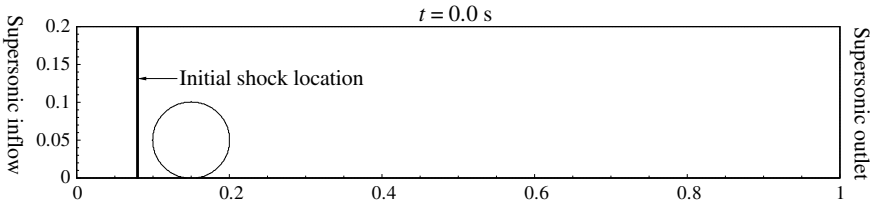


Fig. 9.9 Location of body and shock at time $t = 0$ s

Table 9.2 Position of centre of mass of cylinder (in m) at time $t \sim 0.3$s

IB-FV			Arienti et al. (2003)	
Characteristic grid scale, h	X_c	Y_c	X_c	Y_c
1/500	0.721	0.144	–	–
1/1000	0.697	0.147	0.625	0.145
1/1600	0.684	0.148	–	–

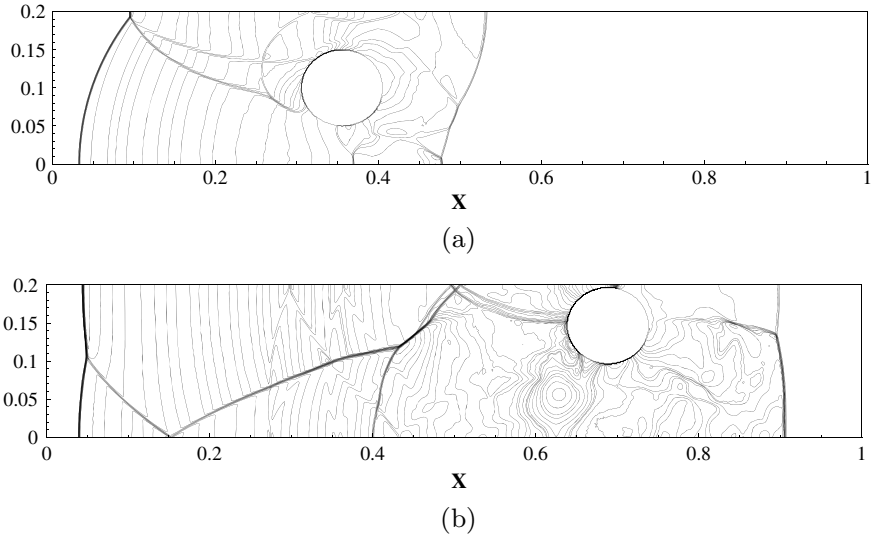


Fig. 9.10 Pressure contours at **a** $t = 0.16$ s (Min: 0, Δ: 0.4, Max: 19.22), **b** $t = 0.3$ s (Min: 0, Δ: 0.4, Max: 19.22) on 1000×200 grid

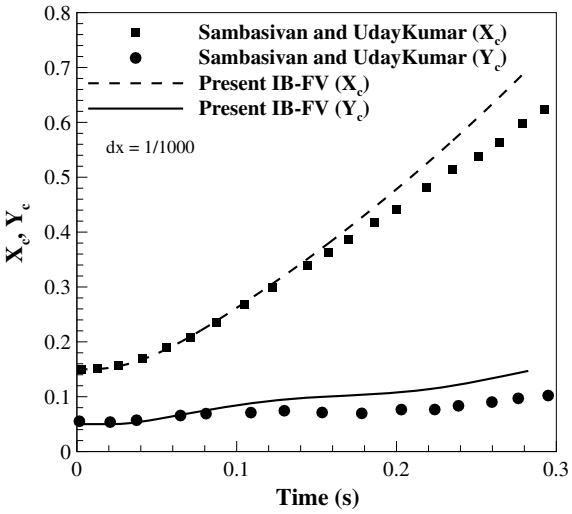


Fig. 9.11 Trajectory of the centre of mass of the cylinder on 1000×200 grid

9.4.5 *Hypersonic Flow Past a Flat Plate*

We first numerically simulate the viscous compressible flow past a flat plate of length $L = 0.5334$ m at a freestream Mach number $M_\infty = 6$. Although the geometry is simple, the high freestream Reynolds number $Re_\infty = 1.4 \times 10^7$ makes it an interesting test case to test the ability of the IB-FV solver in accurately predicting the wall property distributions. The freestream pressure and temperature are $P_\infty = 2211.56$ Pa and $T_\infty = 65$ K, respectively, and the surface of the flat plate is maintained at a constant temperature $T_w = 100$ K. We consider a computational domain of size 0.1×0.5334 which is divided into a total number of control volumes n_c = 50,000, with 250 grid points along the length of the body and 200 grid points normal to it. The grid is non-uniform with clustering near the wall surface and a minimum grid spacing of $\Delta y_{min} = 4 \times 10^{-6}$ m is chosen to ensure that the boundary layers are well-resolved. The computations have also been performed using the FV solver as well for the sake of comparison.

Figure 9.12a shows the comparison of the pressure distribution along the surface of the flat plate obtained from the IB-FV and the FV solvers. The pressure distributions from both the solvers are in excellent agreement as is also the distribution of wall Stanton number in Fig. 9.12b. It is also interesting to note that the heat flux distribution computed suing the IB-FV solver also agrees well with the numerical data of Lillard and Dries (2005). Although the flat plate geometry conforms to the Cartesian grid, the IB-FV solver computes the near-wall quantities using a linear reconstruction as opposed to the FV solver which solves the conservation laws. The excellent comparison of wall pressure and Stanton number distributions is a testimony to the fact that the linear reconstruction suffices to accurately estimate the wall heat fluxes on simple geometries even at high Reynolds numbers.

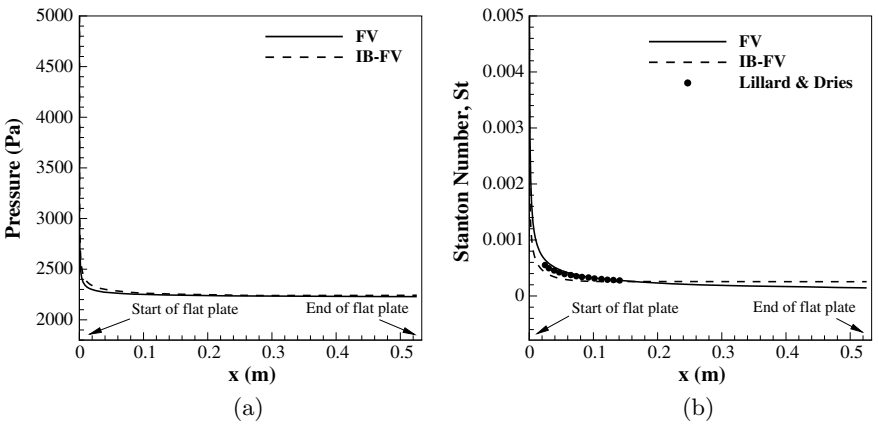


Fig. 9.12 Distribution along the surface of the wall for **a** wall pressure and **b** Stanton number with numerical data of Lillard and Dries (2005)

9.4.6 Hypersonic Flow Past a Compression Ramp

In order to consider non-aligned geometries, we numerically investigate the flow past a compression ramp which has been studied in the past both numerically as well as experimentally (Holden 1978). Figure 9.13a describes the configuration consisting of a straight ramp of length $\mathrm{L} = 0.4394$ m and a semi-vertex angle of 15°. The freestream conditions are taken the same as that of the experimental study with $M_\infty = 11.63$ and Re_∞ = 552,216/m. The freestream temperature is T_∞ = 67.05 K with the ramp surface kept at a constant temperature of T_w = 294.38 K. The computational domain

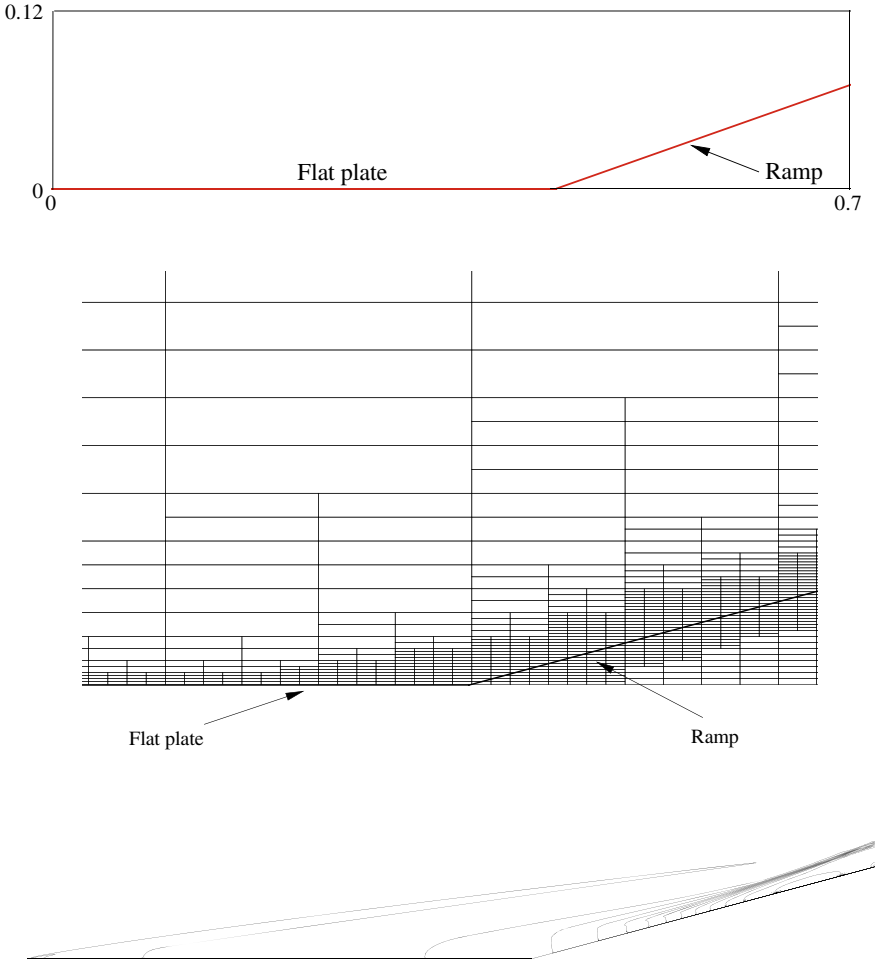


Fig. 9.13 **a** Ramp geometry (schematic, not to scale), **b** locally adapted grid and **c** pressure contour for flow past compression ramp (Min: 0, Δ: 51.66, Max: 620)

is initially divided into n_c = 67,500 volumes with a minimum grid spacing of Δy = 3.2×10^{-4} m. This initial resolution was improved to properly capture the boundary layer by selective adaptation and the final adapted mesh shown in Fig. 9.13b had 229,338 control volumes with a minimum grid spacing of $\Delta y = 2 \times 10^{-5}$ m. This corresponds to a cell Reynolds number (based on freestream conditions and local grid resolution) around 11 and is shown to be sufficient to resolve the boundary layer.

Figure 9.13c shows the pressure contours where the leading edge shock emanating near the compression corner strikes the ramp and this region corresponds to the maximum pressure on the ramp surface. Figure 9.14a compares the surface distribution of coefficient of pressure C_p which shows an excellent agreement with experimental data. The comparison of wall skin friction and Stanton number in Fig. 9.14b and c also agrees quite well with the experimental observations. These results demonstrate that the IB-FV solver is fairly accurate even when the geometries are not conforming to the underlying mesh, provided the mesh resolution near the boundary is sufficient enough. This demands a low value for the cell Reynolds number and can be achieved using selective adaptive refinement as in this study.

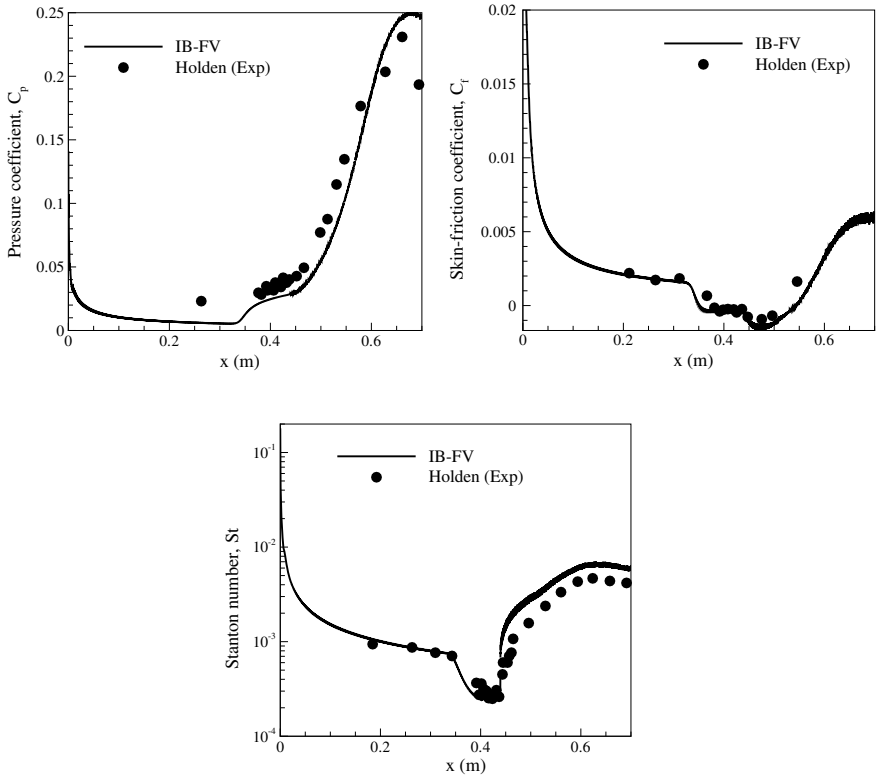


Fig. 9.14 Comparison of **a** pressure coefficient C_p, **b** skin friction coefficient C_f, **c** Stanton number St with experimental data Holden 1978

9.4.7 Hypersonic Flow Past a Cylinder

As a final test case, we consider the laminar hypersonic flow past a blunt configuration that has been studied experimentally in the past. This is a canonical configuration representative of the nose cone of re-entry geometries, and the test case is an ideal one to assess the IB-FV framework for high Reynolds number compressible flows past blunt geometries. The configuration is a circular cylinder of radius 0.0381 m (see Fig. 9.15a) in a hypersonic flow with freestream Mach number of $M_\infty = 8.03$ and freestream Reynolds number of $Re_\infty = 1.835 \times 10^5$. The cylinder walls are maintained at $T_w = 294.44$ K while the freestream temperature is $T_\infty = 124.94$ K, following the experimental study by Wieting (1987). The computational domain of size 0.07 m × 0.14 m is initially discretised by a uniform Cartesian mesh with a grid spacing of 2.3×10^{-4} m, into which the cylinder is immersed. This grid resolution is improved significantly by four levels of local mesh refinement resulting in a final grid of 244,433 control volumes that have a near-wall resolution of 1.43×10^{-5} m.

Figure 9.15b shows the pressure contours from the steady-state solution where the detached bow shock is clearly visible. The surface pressure distribution in Fig. 9.16 is in excellent agreement with the experimental data of Wieting (1987) (for both initial and final grids), indicating that the grid resolution is not critical in predicting the wall pressures. However, one can observe from Table 9.3 that this is not the case for stagnation point heat flux, which is severely under-predicted when compared

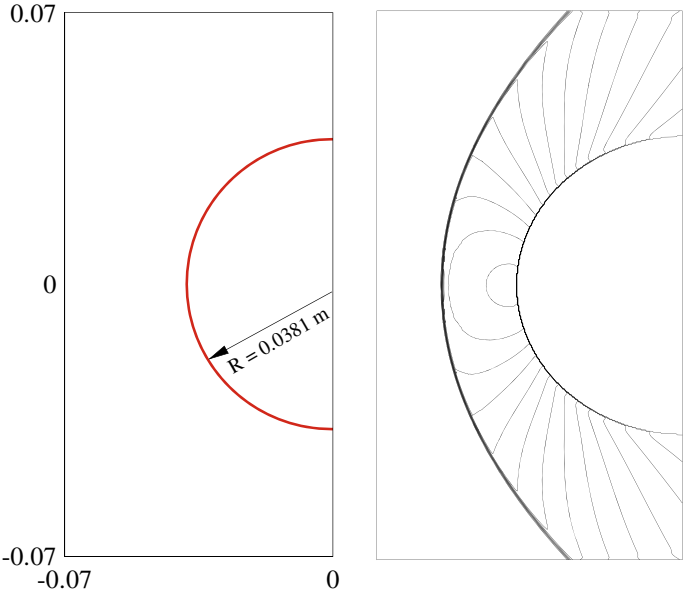


Fig. 9.15 **a** Cylinder geometry and **b** pressure contour for flow past cylinder (Min: 0, Δ: 5087, Max: 71,220)

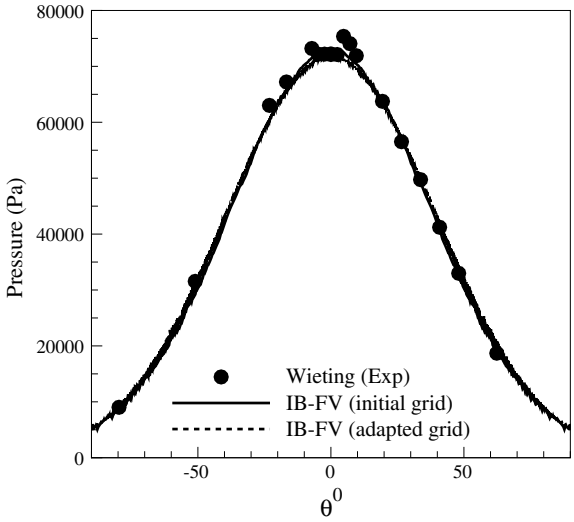


Fig. 9.16 Pressure distribution along the cylinder and its comparison with experimental data (Wieting 1987)

Table 9.3 Comparison of stagnation point heat flux q_o

Method	q_o (W/cm^2)
Wieting (Wieting 1987) (Exp)	72
IB-FV (initial grid)	0.876
IB-FV (adapted grid)	5.58

to the experimental data. Surprisingly, the estimates of stagnation point wall heat flux showed little improvement with local adaptation. This is in stark contrast to the performance of the solver for the test case in Sect. 9.4.6 where the Stanton number distribution on the ramp surface was quite accurately predicted. The inaccurate predictions of the wall heat flux raise concerns regarding the reconstruction strategy and recent studies in Brahmachary (2019) point to the need to evolve physics-driven reconstruction with possibly non-linear/non-polynomial interpolants for velocities and temperature.

9.5 Conclusions

The present work is directed towards the development and assessment of a sharp-interface immersed boundary/finite volume approach for high-speed compressible flows in an unstructured Cartesian mesh framework. The methodology adopts a one-

dimensional linear reconstruction that preserves the sharp geometric interface and the framework is rigorously tested for both inviscid and viscous flows. We show that the framework is not discretely conservative but the conservation errors are found to decrease with grid refinement, pointing to the consistency of the approach. Numerical investigations on inviscid test problems with stationary and moving bodies show that the IB-FV solver can quite accurately predict the wall pressures and aerodynamic forces. However, the numerical framework was not entirely accurate for laminar hypersonic flow problems. While the IB-FV solver could compute the wall pressures accurately for the three geometric configurations studied herein at high Reynolds number, it could compute the surface heat fluxes accurately only for non-blunt geometries. On blunt configurations, the IB-FV solver significantly underestimated the stagnation point heat flux and these estimates were largely unaffected by increasing grid resolution. The source of these errors in heat flux predictions clearly lies in the reconstruction accuracy and conservation errors, and a thorough diagnostic analysis (Brahmachary 2019) needs to be undertaken to uncover the deficiencies of the sharp-interface IB-FV solver and improve its performance for high Reynolds number hypersonic flows.

Acknowledgements The financial support through ISRO-RESPOND project during the course of this work is gratefully acknowledged. The authors would also like to thank Mr. Vinod Kumar and Dr. V. Ashok from VSSC Thiruvananthapuram for their valuable comments on the work.

References

Arienti M, Hung P, Morano E, Shepherd JE (2003) A level set approach to Eulerian-Lagrangian coupling. J Comput Phys 185:213–251. https://doi.org/10.1016/S0021-9991(02)00055-4

Arslanbekov RR, Kolobov VI, Frolova AA (2011) Analysis of compressible viscous flow solvers with adaptive Cartesian mesh. In: 20th AIAA computational fluid dynamics conference, Honolulu, Huwaii. https://doi.org/10.2514/6.2011-3381

Blazek J (2001) Computational fluid dynamics: principles and applications. Elsevier, Amsterdam

Brahmachary S (2019) Finite volume/immersed boundary solvers for compressible flows: development and applications. Ph.D. thesis dissertation, Indian Institute of Technology Guwahati

Brahmachary S, Natarajan G, Kulkarni V, Sahoo N (2018) A sharp-interface immersed boundary framework for simulations of high-speed inviscid compressible flows. Int J Numer Methods Fluids 86:770–791. https://doi.org/10.1002/fld.4479

Brehm C, Hader C, Fasel HF (2015) A locally stabilized immersed boundary method for the compressible Navier-Stokes equations. J Comput Phys 295:475–504. https://doi.org/10.1016/j.jcp.2015.04.023

Cho Y, Chopra J, Morris PJ (2007) Immersed boundary method for compressible high-Reynolds number viscous flow around moving bodies. AIAA 2007-125: https://doi.org/10.2514/6.2007-125

Clarke DK, Salas MD, Hassan HA (1986) Euler calculations for multi-element airfoils using Cartesian grids. AIAA 24:353–358. https://doi.org/10.2514/3.9273

Das P, Sen O, Jacobs G, UdayKumar HS (2017) A sharp interface Cartesian grid method for viscous simulation of shocked particle-laden flows. Int J Comput Fluid Dyn 31:269–291. https://doi.org/10.1080/10618562.2017.1351610

de Tullio MD, Palma PD, Iaccarino G, Pascazio G, Napolitano M (2007) An immersed boundary method for compressible flows using local grid refinement. J Comput Phys 225:2098–2117. https://doi.org/10.1016/j.jcp.2007.03.008

Ghias R, Mittal R, Dong H (2007) A sharp interface immersed boundary method for compressible viscous flows. J Comput Phys 225:528–553. https://doi.org/10.1016/j.jcp.2006.12.007

Ghosh S, Choi JI, Edwards JR (2010) Numerical simulations of effects of micro vortex generators using immersed-boundary methods. AIAA 48:92–103. https://doi.org/10.2514/1.40049

Gilmanov A, Sotiropoulos F (2005) A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. J Comput Phys 207:457–492. https://doi.org/10.1016/j.jcp.2005.01.020

Gustaffson B, Enander EP, Sjogreen B (1991) Solving flow equations for high Mach numbers on overlapping grids. In: Hypersonic flows for reentry problems, pp 585–599. Springer, Berlin

Holden MS (1978) A study of flow separation in regions of shock wave-boundary layer interaction in hypersonic flow. In: AIAA 11th fluid and plasma dynamics conference, Seattle, Washington. https://doi.org/10.2514/6.1978-1169

John B, Kulkarni V (2014) Numerical assessment of correlations for shock wave boundary layer interaction. Comp Fluids 90:42–50. https://doi.org/10.1016/j.compfluid.2013.11.011

Lillard R, Dries K (2005) Laminar heating validation of the overflow code. In: 43rd AIAA aerospace sciences meeting and exhibit, p. 689. https://doi.org/10.2514/6.2005-689

Liuo MS, Steffen CJ Jr (1993) A new flux splitting scheme. J Comput Phys 107:23–39. https://doi.org/10.1006/jcph.1993.1122

Luo H, Baum JD, Löhner R (2006) A hybrid Cartesian grid and grid less method for compressible flows. J Comput Phys 214:618–632. https://doi.org/10.1016/j.jcp.2005.10.002

Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261. https://doi.org/10.1146/annurev.fluid.37.061903.175743

Mizuno Y, Takahashi S, Nonomura T, Nagata T, Fukuda K (2015) A simple immersed boundary method for compressible flow simulation around a stationary and moving sphere. Math Probl Eng 2015:1–17. https://doi.org/10.1155/2015/438086

Mo H, Lien FS, Zhang F, Cornin DS (2016) A sharp interface immersed boundary method for solving flow with arbitrarily irregular and changing geometry. Phys Fluid Dyn. eprint arXiv:1602.06830

Natarajan G (2009) Residual error estimation and adaptive algorithms for fluid flows. Ph.D. thesis dissertation, Indian Institute of Science, Bangalore

Ni RH (1982) A multiple-grid scheme for solving the Euler equations. AIAA J 20:1565–1571. https://doi.org/10.2514/6.1981-1025

Palma PD, de Tullio MD, Pascazio G, Napolitano M (2006) An immersed boundary method for compressible viscous flows. Comp Fluids 35:693–702. https://doi.org/10.1016/j.compfluid.2006.01.004

Peskin CS (1972) Flow patters around heart valves: a numerical method. J Comput Phys 10:252–271. https://doi.org/10.1016/0021-9991(72)90065-4

Pu TM, Zhou CH (2018) An immersed boundary/wall modeling method for RANS simulation of compressible turbulent flows. Int J Numer Methods Fluids 87:217–238. https://doi.org/10.1002/fld.4487

Qu Y, Shi R, Batra RC (2018) An immersed boundary formulation for simulating high-speed compressible viscous flows with moving solids. J Comput Phys 354:672–691. https://doi.org/10.1016/j.jcp.2017.10.045

Sambasivan SK, UdayKumar H (2009) Ghost fluid method for strong shock interactions part 2: Immersed solid boundaries. AIAA J 47:2923–2937. https://doi.org/10.2514/1.43153

Sambasivan SK, UdayKumar HS (2010) Sharp interface simulations with local mesh refinement for multi-material dynamics in strongly shocked flows. Comp Fluids 39:1456–1479. https://doi.org/10.1016/j.compfluid.2010.04.014

Sekhar S, Ruffin SM (2013) Predictions of convective heat transfer rates using a Cartesian grid solver for hypersonic flows. In: 44th AIAA thermophysics conference, San Diego, CA. https://doi.org/10.2514/6.2013-2645

Sotiropoulos F, Yang X (2014) Immersed boundary methods for simulating fluid-structure interaction. Prog Aero Sci 65:1–21. https://doi.org/10.1016/j.paerosci.2013.09.003

Sridar D, Balakrishnan N (2003) An upwind finite difference scheme for mesh less solvers. J Comput Phys 189:1–29. https://doi.org/10.1016/S0021-9991(03)00197-9

Udaykumar HS, Shyy W (1995) Simulation of inter facial instabilities during solidification—I. Conduction and capillarity effects. Int J Heat Mass Transf 38:2057–2073. https://doi.org/10.1016/0017-9310(94)00315-M

Wieting AR (1987) Experimental study of shock wave interface heating on a cylindrical leading edge. NASA TM-100484

Ye T, Mittal R, Udaykumar HS, Shyy W (1999) An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. J Comput Phys 156:209–240. https://doi.org/10.1006/jcph.1999.6356

Zhang Y, Zhou CH (2014) An immersed boundary method for simulation of inviscid compressible flows. Int J Numer Methods Fluids 74:775–793. https://doi.org/10.1002/fld.3872

Zhao H, Hu P, Kamakoti R, Dittakavi N, Xue L, Ni K, Mao S, Marshall DD, Aftosmis M (2010) Towards efficient viscous modeling based on Cartesian methods for automated flow simulation. In: 48th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, Orlando, Florida, (2010). https://doi.org/10.2514/6.2010-1472

Chapter 10
A Higher-Order Cut-Cell Methodology for Large Eddy Simulation of Compressible Viscous Flow Problems with Embedded Boundaries

Balaji Muralidharan and Suresh Menon

10.1 Introduction

The advantages of using embedded boundary (EB) methods for computational fluid dynamics (CFD) applications are well known. The foremost being ease of grid generation for complex geometries and moving boundaries. In EB approaches, the domain boundaries are not resolved by the numerical grid, rather the numerical schemes used to solve the flow governing equations are modified appropriately to account for the presence of physical boundaries. It is in this numerical treatment of the embedded boundary the various EB approaches vary. An excellent overview of the existing methods to represent embedded boundaries within the background mesh is provided by Mittal and Iaccarino (2005). For the purposes of this study, we only consider the cut-cell-based EB method in which regular mesh elements cut by the intersection of the solid boundary are reshaped to conform to the shape of the interface. The cut-cell approach is designed to satisfy the underlying conservation laws for the cells near the interface. Strict global and local conservation of mass, momentum, and energy is guaranteed by resorting to a finite volume discretization even for the cut-cells. The Cartesian cut-cell finite volume methods (Clarke et al. 1986; Udaykumar et al. 1996; Hartmann et al. 2011; Muralidharan and Menon 2016) are, therefore, in comparison to finite difference ghost cell methods (Kim 2001; Majumdar 2001), attractive as they enforce strict conservation and also can avoid the generation of spurious pressure fluctuations that are observed typically with ghost fluid methods (Cecere and Giacomazzi 2014; Mittal and Iaccarino 2005; Merlin et al. 2012).

B. Muralidharan
Combustion Research and Flow Technology, 6210 Keller's Church Rd., Pipersville, PA 18947, USA
e-mail: balajim@craft-tech.com

S. Menon (✉)
School of Aerospace Engineering, Georgia Institute of Technology, Atlanta 30332, GA, USA
e-mail: suresh.menon@ae.gatech.edu

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_10

One of the motivations of developing EB methods is to apply them to solve realistic flow problems involving complex geometries. Due to highly turbulent nature of most of the practical flows, resolving all scales of motion, as is done in a Direct Numerical Simulation (DNS), is not possible due to the high computational costs involved. The alternative is to employ large eddy simulation (LES) in which only the most energy-containing eddies are resolved by the numerical grid and effect of small scales of motion on the larger scales is modeled. Adaptive mesh refinement (AMR) is another popular strategy for reducing computational cost by providing higher grid resolution only in the regions of interest. AMR was originally proposed for shock hydrodynamics (Berger and Colella 1989) and has been traditionally applied to mainly inviscid flows to capture features such as shocks, contact discontinuities, and expansions.

The introduction of unconventional numerical techniques such as embedded boundary methods and AMR can complicate the closure problem for LES. The majority of subgrid closures for LES have been developed for body-conformal, uniform grids without local refinement. The behavior of the closure models for unconventional methodologies such as dynamic mesh refinement (Berger and Colella 1989) and embedded boundary techniques (Mittal and Iaccarino 2005) is not completely understood. Additionally, a common problem with most EB methods is that they are of lower-order accuracy near boundary. Besides, these methods also suffer from issues such as mass loss and noisy reconstruction of flow solution quantities such as wall shear stress and heat flux (Coirier and Powell 1996). In the context of turbulence modeling using LES technique, the numerical errors at the boundary can strongly interact with the subgrid closure models introducing a significant uncertainty in the simulation results (Kravchenko and Moin 1997). Therefore, use of high-order EB schemes with smooth behavior of flow quantities and their derivatives at the boundary becomes particularly relevant for LES as the truncation errors from lower-order schemes can exceed the magnitude of the subgrid-scale term (Kravchenko and Moin 1997).

To date, there have been only a few reported works on modeling turbulence using the cut-cell-based EB methods. Meyer et al. (2010) developed a conservative second-order accurate immersed interface method suitable for LES of high Reynolds number incompressible flows. However, an implicit LES approach in the capacity of ALDM approach was employed for the turbulence closure. Essentially, the numerical dissipation of the scheme was assumed to mimic the physical dissipation due to action of small-scale unresolved turbulence. In a recent article, Berger and Aftosmis (2012) extensively analyzed modeling of steady viscous compressible flows using Cartesian cut-cell finite volume method. They explored the use of wall models for laminar and turbulent flows to suppress numerical oscillations in the second derivatives used for viscous flux computations. To the best of the author's knowledge, there have not been many studies in the area of LES with embedded boundary methods and dynamic refinement for turbulent flow problems.

A high-order accurate adaptive Cartesian cut-cell method has been recently developed by Muralidharan and Menon (2016, 2018) that addresses most of the shortcomings of the previous approaches. A high-order solution was achieved by using a k-exact reconstruction based on a piecewise polynomial approximation of the flow

solution locally near the embedded boundary. A novel cell clustering approach was employed that was based on previously established cell-linking approaches for dealing with the 'small cell' problem afflicting all the cut-cell methods. One of the key strengths of the cell clustering approach is that it preserves the order of accuracy of the underlying numerical scheme both *locally* and *globally*. Additionally, the approach ensured smooth reconstruction of quantities involving flow gradient such as the skin friction coefficient. These features make this approach very suitable for LES of turbulent flow problems. In an another recent study by the authors, a multi-level subgrid closure for LES of compressible flow problem with local adaptive mesh refinement was developed (Muralidharan and Menon 2019) (henceforth called as AMRLES). Consistent and conservative behavior of the subgrid kinetic energy across the multiple levels was demonstrated using the AMRLES approach. The goal of this study is to extend the multi-level closure for LES to problems with embedded boundaries. Appropriate closure model corrections to AMRLES framework suited to the cut-cell EB method are proposed. The cut-cell-AMRLES strategy is then assessed for canonical flow problems. Detailed evaluation of the closure model coefficients is performed and reported.

The organization of the paper is as follows. In the first section, the mathematical formulation and the numerical approach are described. The details of the closure of the subgrid-scale turbulence in the presence of a locally refined grid and embedded boundary are also detailed in this section. The results for canonical turbulent flow problems with the proposed cut-cell-AMRLES framework are reported in the next section. Finally, summary of the work is presented along with future directions in the conclusion section.

10.2 Mathematical Formulation and Numerical Approach

10.2.1 Governing Equations for Multi-level AMRLES

In the current study, block-structured adaptive mesh refinement is performed near embedded boundaries to better resolve the near-wall flow features. To perform block-based refinement, the flow solver is interfaced with BoxLib AMR library developed at LBNL. Accordingly, the Favre-filtered compressible LES governing equations for a multi-level AMR grid with $l = 1, 2, \ldots, N$ levels of refinement as detailed in a previous work (Muralidharan and Menon 2019) are given by:

$$\frac{\partial}{\partial t}\begin{pmatrix} \overline{\overline{\rho}}^{l} \\ \overline{\overline{\rho}}^{l}\widetilde{\widetilde{u}}_j^{\,l} \\ \overline{\overline{\rho}}^{l}\widetilde{\widetilde{E}}^{l} \end{pmatrix} + \frac{\partial}{\partial x_j}\begin{pmatrix} \overline{\overline{\rho}}^{l}\widetilde{\widetilde{u}}_i^{\,l} \\ \overline{\overline{\rho}}^{l}\widetilde{\widetilde{u}}_i^{\,l}\widetilde{\widetilde{u}}_j^{\,l} + \overline{\overline{p}}^{l}\delta_{ij} - \widetilde{\widetilde{\tau}}_{ij}^{\,l} + \tau_{ij}^{\mathrm{sgs},l} \\ (\overline{\overline{\rho}}^{l}\widetilde{\widetilde{E}}^{l} + \overline{\overline{p}}^{l})\widetilde{\widetilde{u}}_j^{\,l} - \widetilde{\widetilde{u}}_i^{\,l}\widetilde{\widetilde{\tau}}_{ij}^{\,l} + \overline{\overline{q}}_j^{\,l} + H_j^{\mathrm{sgs},l} + \sigma_j^{\mathrm{sgs},l} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{10.1}$$

where ρ, u_i, E, *andp* are the density, velocity components, total energy, and pressure, respectively. τ_{ij} is the viscous stress tensor, and q_i is the thermal conductivity flux in the ith direction. In the above equations, all the subgrid-scale terms, indicated with a sgs superscript, are unclosed, and therefore, require modeling. The multi-level filtering operation, denoted by $\overline{\overline{\cdot}}^{l}$, can be defined as:

$$\overline{\overline{\phi}}^{l} = \mathcal{G}_l * \mathcal{G}_{l+1} * \cdots * \mathcal{G}_N \phi. \tag{10.2}$$

for any flow quantity ϕ. $\mathcal{G}_l$ is the filtering operator associated with level l which can vary from $l = 1, 2, \ldots, N$ with N being the maximum level of refinement. The representation of the filtered quantity on a multi-level AMR grid is shown both in the wavenumber space and in the physical hierarchical grid system in Fig. 10.1. The wavenumber corresponding to each AMR level and the corresponding filtered quantity at that level is indicated in the figure.

The closure models for each of the sgs, l terms are summarized below:

$$\tau_{ij}^{\mathrm{sgs},l} = -2\overline{\overline{\rho}}^{l}\, \nu_t^l \left(\widetilde{\widetilde{S}}_{ij}^{l} - \frac{1}{3}\widetilde{\widetilde{S}}_{kk}^{l}\delta_{ij} \right) + 2/3\overline{\overline{\rho}}^{l}\overline{\overline{k}}^{\mathrm{sgs},l}\delta_{ij}, \tag{10.3}$$

$$\nu_t^l = C_\nu^l \sqrt{k^{\mathrm{sgs},l}}\Delta^l, \tag{10.4}$$

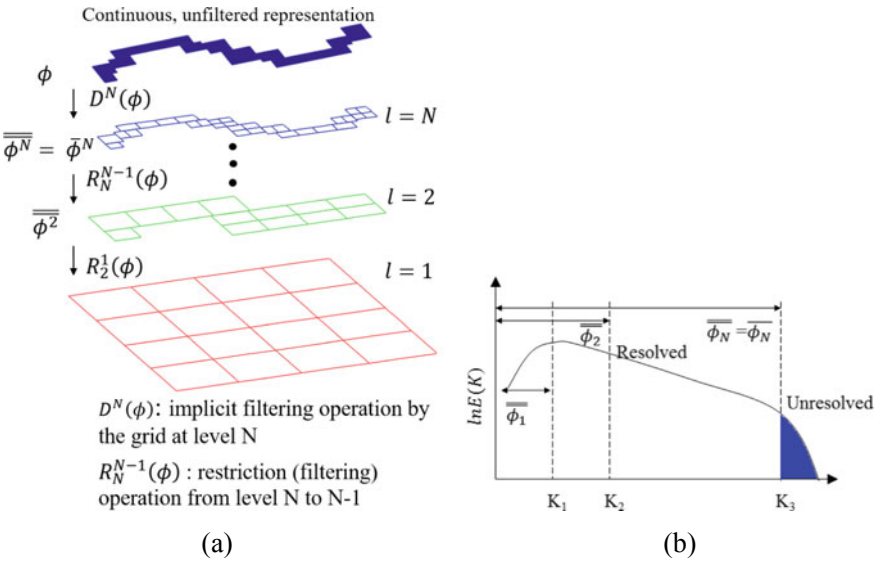


Fig. 10.1 Schematic of turbulent kinetic energy spectra in **a** physical space and **b** wavenumber space. The multi-level filtering of a flow quantity ϕ and the associated wave number are also indicated. Reprinted with permissions from Muralidharan and Menon (2019)

and

$$H_i^{\mathrm{sgs},l} + \sigma_i^{\mathrm{sgs},l} = -\left(\overline{\overline{\rho}}^{l} \nu_t^l + \mu^l\right) \frac{\partial \overline{\overline{k}}^{\mathrm{sgs},l}}{\partial x_i} - \frac{\overline{\overline{\rho}}^{l} \nu_t^l c_p^l}{Pr_t^l} \frac{\partial \widetilde{\widetilde{T}}^{l}}{\partial x_i} + \widetilde{\widetilde{u}}^{l}{}_j \tau_{ij}^{\mathrm{sgs},l}. \tag{10.5}$$

The transport equation for the subgrid kinetic energy for a multi-level AMR grid system is given by:

$$\frac{\partial \overline{\overline{\rho}} k^{\mathrm{sgs},l}}{\partial t} + \frac{\partial}{\partial x_i}\left(\overline{\overline{\rho}}^{l} \widetilde{\widetilde{u}}_i^{l} k^{\mathrm{sgs},l}\right) = \mathcal{T}_{k^{\mathrm{sgs},l}} + pd_{k^{\mathrm{sgs},l}} + P_{k^{\mathrm{sgs},l}} - D_{k^{\mathrm{sgs},l}}, \tag{10.6}$$

with the different closure terms in the k^{sgs} equation taking the following form:

$$\mathcal{T}_{k^{\mathrm{sgs},l}} = \frac{\partial}{\partial x_i}\left[(\overline{\overline{\rho}}^{l} \nu_t^l + \mu)\frac{\partial k^{\mathrm{sgs},l}}{\partial x_i} + \frac{\overline{\overline{\rho}}^{l} \nu_t^l \widetilde{\widetilde{R}}^{l}}{Pr_t^l} \frac{\partial \widetilde{\widetilde{T}}^{l}}{\partial x_i}\right], \tag{10.7}$$

$$pd_{k^{\mathrm{sgs},l}} = \alpha_{pd}^l M_t^{\mathrm{sgs},l2} \left(\frac{\overline{\overline{\rho}}^{l} \widetilde{\widetilde{S}}^{l} k^{\mathrm{sgs},l}}{D^{\mathrm{sgs},l}}\right)^2 \left(P_{k^{\mathrm{sgs},l}} - D_{k^{\mathrm{sgs},l}}\right), \tag{10.8}$$

$$P_{k^{\mathrm{sgs},l}} = -\tau_{ij}^{\mathrm{sgs},l} \widetilde{\widetilde{S}}_{ij}^{l}, \tag{10.9}$$

$$D_{k^{\mathrm{sgs},l}} = \overline{\overline{\rho}}^{l} C_\epsilon^l \left(k^{\mathrm{sgs},l}\right)^{3/2} / \Delta^l. \tag{10.10}$$

Equations (10.3–10.10) are in fact exact equivalents of a single-level k^{sgs} transport equation with single-level flow variables now replaced with their multi-level representation. The coefficients, C_ν^l , C_ϵ^l, α_{pd}^l, and Pr_t^l, are computed still computed dynamically for each level after employing a test filter with twice the local grid size and using a least square approach (Génin and Menon 2010).

As detailed in the previous study (Muralidharan and Menon 2019), the sgs terms in Eq. (10.1) are closed using the standard single-level closures for each level independently. The only difference is in the treatment of the subgrid turbulent kinetic energy $k^{\mathrm{sgs},l}$ for which an additional correction is performed as given by:

$$\overline{\overline{\rho}} k^{\mathrm{sgs},l} = \overline{\overline{\rho k^{\mathrm{sgs}}}}^{l} + \overline{\overline{\rho\delta}}^{\mathrm{sgs},l}, \tag{10.11}$$

$$\overline{\overline{\rho\delta}}^{\mathrm{sgs},l} = \overline{\overline{\rho u_i u_i}}^{l} - \frac{\overline{\overline{\rho u_i}}^{l} \overline{\overline{\rho u_i}}^{l}}{\overline{\overline{\rho}}^{l}}. \tag{10.12}$$

The multi-level correction procedure is illustrated in Fig. 10.2. For unrefined regions, the single-level transport equation-based closure is employed. But for the refined regions indicated by yellow and blue colors, the correction described by Eq. (10.12) is applied. The multi-level formulation can be seen as a mixed model

that employs, the transport equation-based sgs model at the finest resolution and adding a correction based on the explicit filtering of represented turbulent kinetic energy on the grid resolution finer than the current level.

10.2.2 Extension of the Multi-level AMRLES to Embedded Boundaries

While in theory the multi-level formulation can be naturally extended for wall bounded flows with embedded boundary representation, the procedure for dynamically computing the coefficients becomes more complicated as test-level filtering is not clearly defined at the embedded boundary. To overcome this problem, a two-layer approach to the closure model is suggested. On the finest level comprising the wall boundary, the flow is solved without any closure model (in a DNS mode) and away from the boundary on the coarser underlying grids, the multi-level sgs closure is employed. The multi-level correction from the finest (N) to the coarser grid ($N-1$) injects the filtered subgrid turbulent kinetic energy ($k^{\mathrm{sgs},N-1}$) which is then transported on the coarser grid levels. The two-layer sgs closure with EB is illustrated in Fig. 10.2.

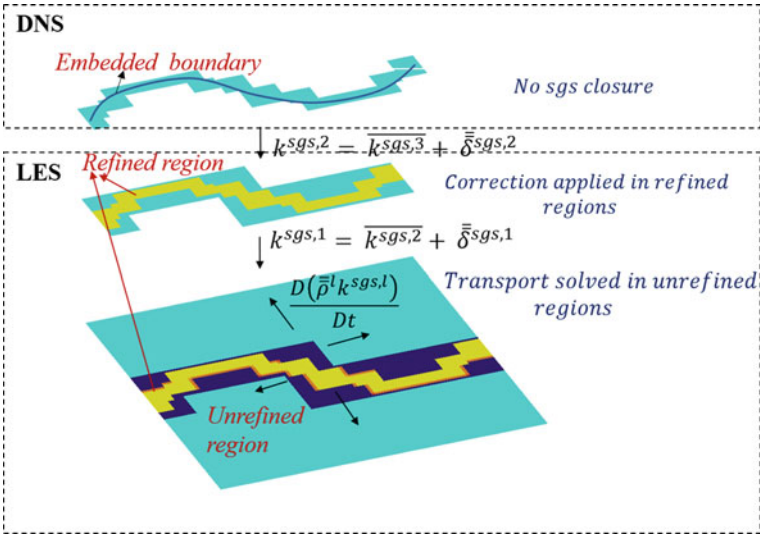


Fig. 10.2 Schematic of the multi-level correction for k^{sgs} on a AMR mesh with EB

10.2.3 Formulation and Implementation of Cut-Cell Method

A Cartesian-based strictly conservative cut-cell method that is upto third-order accurate for viscous problems with embedded boundaries has been developed in the past. Readers are referred to a previously published work of the authors (Muralidharan and Menon 2016) for the detailed formulation and validation of the high-order cut-cell method. A summary of the high-order cut-cell method is provided below.

Cut-cell method (Hartmann et al. 2008; Yang et al. 2000) is used in this work to represent embedded boundaries on a Cartesian grid. Information for defining the cut-cells at the embedded boundary is extracted from a levelset field description. Levelset, as defined by Osher and Sethian (1988), Osher and Fedkiw (2003), is a continuous scalar field having values $\phi > 0$ in the fluid region, $\phi < 0$ in the solid region and $\phi = 0$ at the interface. Once the levelset field is described completely, all the cut-cell metrics can be computed.

To create a cut-cell, the levelset field is assumed to be piecewise linear in a cell and is given as:

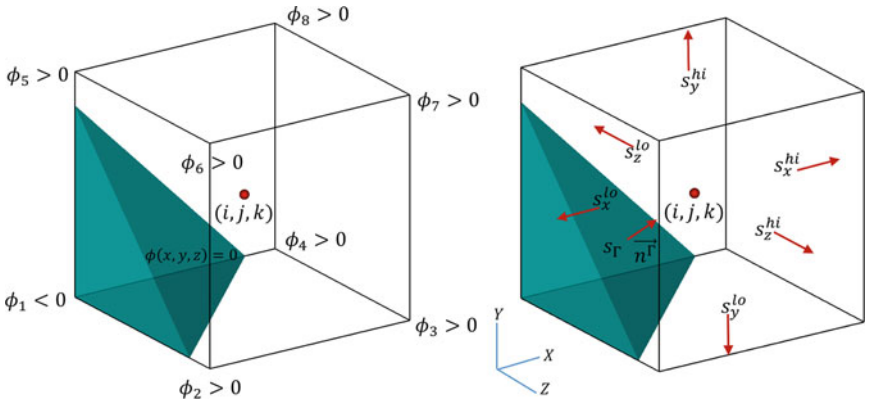
$$\phi(x, y, z) = \sum_{p1=0}^{1} \sum_{p2=0}^{1} \sum_{p3=0}^{1} x^{p1} y^{p2} z^{p2} a_{p1,p2,p3}, \quad (p1 + p2 + p3) \leq 1 \tag{10.13}$$

in which the coefficients, $a_{p1,p2,p3}$, are determined based on the nodal values, ϕ_i, $i = 1, 8$ for a given computational cell. The embedded boundary surface is defined by the function $\phi(x, y, z) = 0$. The boundary equation along with the linear system of equations representing the cut-cell edges is solved simultaneously to provide the points of intersection of the boundary with the edges. The process of finding the cut surface is illustrated in Fig. 10.3a. As shown, the embedded surface is approximated by a planar cut in a given computation cell (i, j, k).

The main idea behind achieving a higher-order accuracy at the embedded boundaries is use of a piecewise high-order polynomial approximation of cell-centered flow quantities as proposed by Ivan and Groth (2014). Accordingly, the following reconstruction polynomial of order k for any conservative or primitive flow quantity u in a given cell i is defined as follows:

$$u_i^k(x, y, z) = \sum_{p_1=0}^{k} \sum_{p_2=0}^{k} \sum_{p_3=0}^{k} (x - x_{c,i})^{p_1} (y - y_{c,i})^{p_2} (z - z_{c,i})^{p_3} D^k_{p_1,p_2,p_3}, \quad p_1 + p_2 + p_3 \leq k \tag{10.14}$$

where $(x_{c,i}, y_{c,i}, z_{c,i})$ are the cell center coordinates and $D^k_{p_1,p_2,p_3}$ are coefficients of kth-order approximation of u, which can be proved to be scalar multiples of derivatives of u using Taylor series expansion. Once these coefficients are determined, the above polynomial approximation in Eq. (10.14) can be employed to reconstruct, anywhere within the cell i, the quantity u and its pth derivative with the order of



(a) Cut-cell creation (b) Cut-cell definition

Fig. 10.3 Schematic of a three-dimensional cut-cell: **a** creation from levelset description ϕ with cut surface described by $\phi(x, y, z) = 0$. **b** Various geometric variables for defining a cut-cell to represent an embedded boundary. Reprinted with permissions from Muralidharan and Menon (2016)

accuracy $(k+1)$ and $(k-p+1)$, respectively. Using the volume-averaged values of the current cell $\overline{u_i}$,

$$\overline{u_i} = \frac{1}{V}\int_v \sum_{p_1=0}^{k}\sum_{p_2=0}^{k}\sum_{p_3=0}^{k}(x-x_{c,i})^{p_1}(y-y_{c,i})^{p_2}(z-z_{c,i})^{p_3}D^k_{p_1,p_2,p_3}dv, \quad (10.15)$$

and the neighboring cell $\overline{u_j}$, the coefficients $D^k_{p_1,p_2,p_3}$ can be found by solving a system of linear equations defined as follows:

$$\overline{u}_j - \overline{u}_i = \sum_{p_1=0}^{k}\sum_{p_2=0}^{k}\sum_{p_3=0}^{k}(\widehat{x^{p_1}y^{p_2}z^{p_3}})_{ij}D^k_{p_1,p_2,p_3} \mid j = 1,\ldots,n_p,$$
$$p_1+p_2+p_3 \le k \quad (10.16)$$

where n_p represents the number of neighbors that are required to solve the ith cell-centered quantity and depends on the order of reconstruction. In Eq. (10.16), $\widehat{x^{p_1}y^{p_2}z^{p_3}}$ is the geometric moment of jth cell about ith cell center given by:

$$(\widehat{x^{p_1}y^{p_2}z^{p_3}})_{ij} = \int_{v_j}(x-x_{c,i})^{p_1}(y-y_{c,i})^{p_2}(z-z_{c,i})^{p_3}dv. \quad (10.17)$$

More details on solving Eq. (10.17) can be found in a previous work by the authors (Muralidharan and Menon 2016).

10.2.4 Numerical Approach

The fluid solver is a finite volume, compressible, time-accurate LES code capable of solving multi-phase, reacting, turbulent flows in both simple and complex geometries using a structured, parallel multi-block scheme with second- and/or fourth-order accuracy (Chakravarthy and Menon 2001; Génin and Menon 2010). Unless stated otherwise, to evaluate the inviscid and viscous fluxes away from the embedded boundary, Mac-Cormack's predictor–corrector (MacCormack 2003) method is employed on the full cells. The finite volume version of the Mac-Cormack's method couples the time and spatial integration schemes. First-order or second-order extrapolation of cell-averaged values that alternates between the downwind and the upwind directions at each step is performed to compute the fluxes on the cell faces. This results in a second-order accurate scheme in both time and space. A higher-order extrapolation can increase the accuracy of the scheme to fourth order.

At the embedded boundary, the Central Essentially Non-Oscillating (CENO) scheme using the k-exact reconstruction is used. The viscous fluxes are computed using central finite difference, and the inviscid fluxes are evaluated by solving the Riemann problem at the cell interfaces using the Hartmann-Lax-van Leer family of approximate Riemann solvers (HLL and HLLC) (Toro 2009).

10.3 Results and Discussion

The goal of the following numerical case studies is to assess the multi-level cut-cell-AMRLES subgrid closure for performing LES of flows with embedded boundaries. To demonstrate the accuracy of the scheme, order of accuracy analysis for a Laplace problem on a domain with embedded boundaries is reported. Results are then presented for LES of transitional flow past a cylinder and sphere.

10.3.1 Order of Accuracy Analysis for the Cut-Cell EB Method

To demonstrate the accuracy of the cut-cell finite volume scheme, the following Laplace's problem:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = 0, \tag{10.18}$$

is solved on a series of successively refined grids and with two different orders of reconstruction: $k = 2$ and $k = 3$. The exact solution of Eq. (10.18) is: $\psi_{\text{exact}} =$

$\sin x \exp y$. Although a high-order accurate reconstruction of the flow-field quantities and their derivatives iss obtained using the k-exact approach, the solution accuracy degrades due to the aforementioned cell-mixing process. Particularly for viscous flow problems, the classical cell-mixing method achieves numerical stability in computations but causes significant noise in the reconstruction of the derivative quantities, e.g., shear stress and heat flux (Muralidharan and Menon 2016).

The cell clustering scheme is now assessed for this Laplace's problem on a domain, D with a embedded boundary, Γ. The boundary is defined by a levelset description ϕ on a 1×1 unit domain given by:

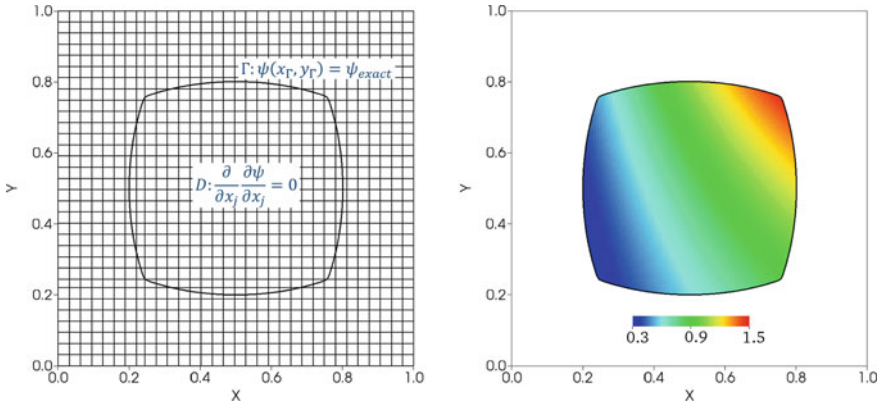
$$\phi_1(x, y) = 1 - \sqrt{\frac{(x - x_c)^2}{r_1^2} + \frac{(y - y_c)^2}{{r_2}^2}}, \tag{10.19}$$

$$\phi_2(x, y) = 1 - \sqrt{\frac{(x - x_c)^2}{r_2^2} + \frac{(y - y_c)^2}{{r_1}^2}}, \tag{10.20}$$

$$\phi(x, y) = \min(\phi_1, \phi_2) \tag{10.21}$$

where (x_c, y_c) is set at $(0.5, 0.5)$ and $r_1 = 0.3$, $r_2 = 0.5$. The boundary represented by the above levelset description is shown in the following Fig. 10.4.

Equation (10.18) is solved using the finite volume approach described in Sect. 10.2. All the conserved quantities are frozen, and an additional scalar equation is solved for ψ with a Dirichlet boundary condition $\psi(x_\Gamma, y_\Gamma) = \psi_{\text{exact}}$ imposed at the immersed boundaries. The L^1, L^2, and L^∞ norm of the errors are computed as $L^p(e_\psi) = \left(\frac{1}{\sum_i v_i} \sum_i v_i |e_\psi|^p\right)^{\frac{1}{p}}$ with v_i being the volume of cell, p is error norm, and



(a) (b)

Fig. 10.4 **a** Immersed domain for the Laplace's problem represented using cut-cells. **b** Exact solution to Laplace's problem, $\psi_{\text{exact}} = \sin x \exp y$. Reprinted with permissions from Muralidharan and Menon (2016)

Table 10.1 Error norms for solution to Laplace equation for different orders of k-exact reconstruction [Reprinted with permissions from Muralidharan and Menon (2016)]

Grid	L^1 Norm	Order	L^2 Norm	Order	L^∞ Norm	Order
$k = 2$						
40^2	2.03×10^{-6}	–	2.37×10^{-6}	–	4.173×10^{-6}	–
80^2	5.04×10^{-7}	**2.01**	5.90×10^{-7}	**2.0**	1.053×10^{-6}	**1.99**
160^2	1.24×10^{-7}	**2.02**	1.46×10^{-7}	**2.01**	2.60×10^{-7}	**2.02**
320^2	3.15×10^{-8}	**1.98**	3.70×10^{-8}	**1.98**	6.59×10^{-8}	**1.98**
$k = 2$						
40^2	9.06×10^{-8}	–	1.14×10^{-7}	–	3.14×10^{-7}	
80^2	1.0×10^{-8}	**3.17**	1.26×10^{-8}	**3.17**	4.48×10^{-8}	**2.81**
160^2	1.23×10^{-9}	**3.03**	1.54×10^{-9}	**3.03**	6.30×10^{-9}	**2.83**
320^2	1.55×10^{-10}	**2.99**	1.95×10^{-10}	**2.98**	8.65×10^{-10}	**2.86**

The bold lettering in the table has been used to emphasize and highlight the order of accuracy of the numerical scheme

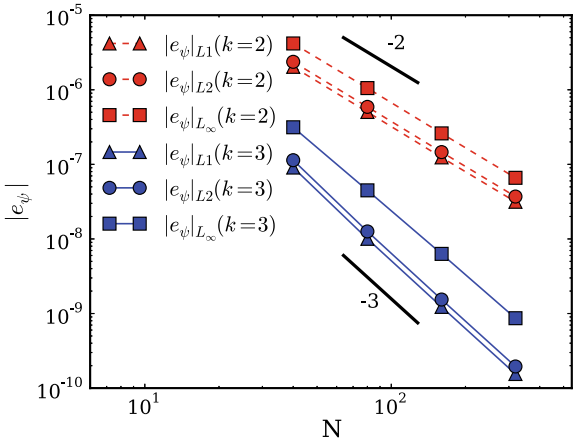


Fig. 10.5 Error norms of ψ for the solution to the Laplace's problem at different grid resolutions for the Laplace's problem with different orders of k-exact reconstruction [Reprinted with permissions from Muralidharan and Menon (2016)]

$|e_\psi| = |\psi - \psi_{\text{exact}}|$. The error norms are reported for different mesh sizes and for $k = 2$ and $k = 3$ in Table 10.1. The plot of the error norms along with the design order of accuracy is shown in Fig. 10.5. To maintain consistency of the error analysis, the k-exact-based CENO reconstruction is used for both the full and cut-cells to evaluate the viscous fluxes.

The error in the solution includes the effects of small cell clustering and mixing. With the cell clustering algorithm, the design order of accuracy is achieved for both $k = 2$ and $k = 3$. Previous studies employing cut-cell (Hartmann et al. 2011; Cecere and Giacomazzi 2014) have only reported the reconstruction error which does not account for the small cell effects. It is noted that in the current approach, the design order of accuracy is achieved both locally and globally. This clearly indicates the robustness of the proposed cell clustering approach in handling complex surface topologies and still achieves higher order. The Laplace's problem is representative of the class of viscous flow problems since it involves elliptic, diffusion like term, and therefore, the inferences made on order of accuracy for this simple problem should be applicable to compressible viscous flow problems in general.

10.3.2 LES of Re_d = 3900 Flow Past a Cylinder

In this study, LES is employed to simulate the turbulent flow of Re_d = 3900 over a cylinder of diameter, d. The simulations are performed in a large rectangular domain of size $(30d \times 30d \times \pi d)$ with a base resolution of $(150 \times 150 \times 20)$. As shown in Fig. 10.6, six AMR levels are employed such that the effective resolution at the cylinder surface is $0.003125d$, which falls in at around $y^+ = 4$, where $+$ indicates non-dimensionalization by the viscous length scale. The first point of the wall is located at $y^+ = 2$. The grid resolution is comparable to a previous study of the same

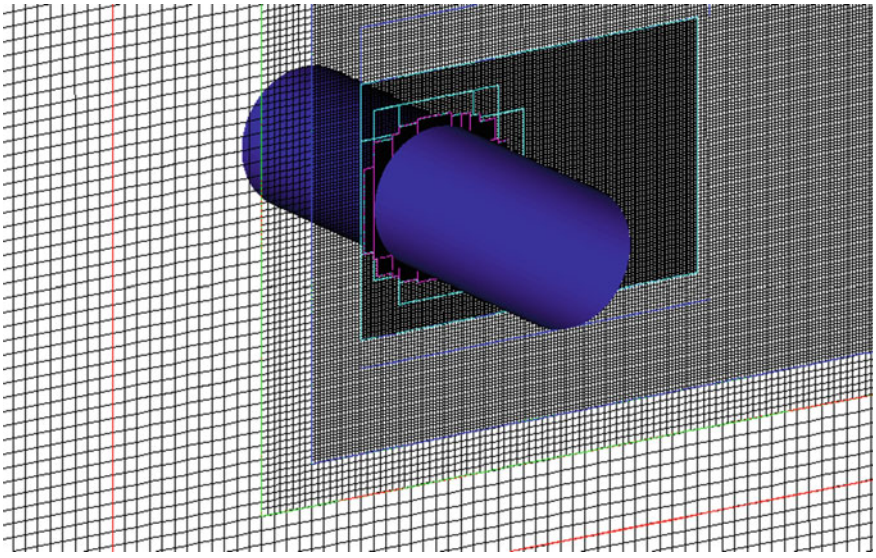


Fig. 10.6 Snapshot of local mesh refinement near cylinder surface for Re_d = 3900 flow past a cylinder

problem (Ranjan and Menon 2015). This problem has been extensively studied using both body conformal and immersed boundary approaches in the past and therefore is an ideal reference case for evaluating the current AMRLES closure with embedded boundaries. Characteristic-based subsonic inflow is used in the left boundary, while subsonic outflow condition is prescribed to the top, bottom, and right boundaries. Front and back surfaces are prescribed with periodic boundary condition.

The stringent wall resolution requirement is due to lack of use of any wall modeling for performing AMRLES which makes this a wall-resolved LES. The coefficients for the subgrid closure models are evaluated dynamically using the LDKM approach (Génin and Menon 2010). The flow Mach number is set at $M = 0.2$ which is low enough to avoid any compressibility effects. The time history of the drag and lift coefficient plots is shown in Fig. 10.7.

The average drag coefficient of $\overline{C_d} \approx 1$ matches with the data from past studies (Son and Hanratty 1969; Ranjan and Menon 2015). The amplitude changes in the lift coefficient are due to vortex shedding events occurring downstream of the cylinder. The vortex structures in the wake of the cylinder are identified by the iso-surface of Q-criterion colored with streamwise velocity and are shown in Fig. 10.8. It can be observed that the boundary layer separates around the top and bottom of cylinder and forms shear layers which breaks up into coherent structures and eventually into small-scale turbulence within a couple of diameters downstream of the cylinder.

The instantaneous snapshots of vorticity magnitude, subgrid kinetic energy, and eddy viscosity ratio are shown in Fig. 10.9. An important observation from the subgrid kinetic energy plot is that the k^{sgs} is generated in shear layer following the coarsening of the finest AMR mesh covering the cylinder surface. As noted in the grid turbulence case study discussed in a previous study (Muralidharan and Menon 2019), the generation of k^{sgs} from a fine/coarse AMR interface occurs solely due to the multi-level subgrid closure. The inflow is laminar and therefore in the free-

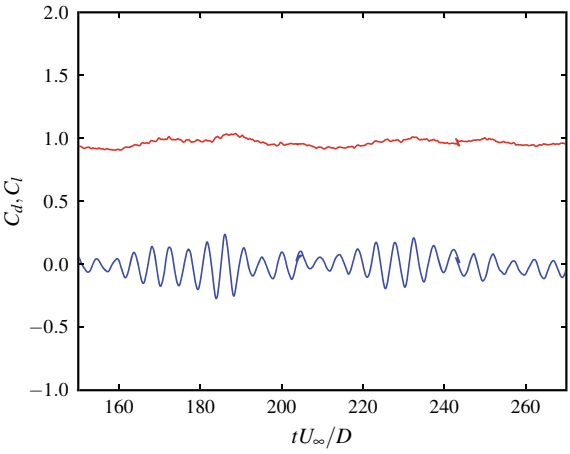


Fig. 10.7 Time history of drag (C_d) and lift (C_l) coefficient of $\text{Re}_d = 3900$ flow past a cylinder

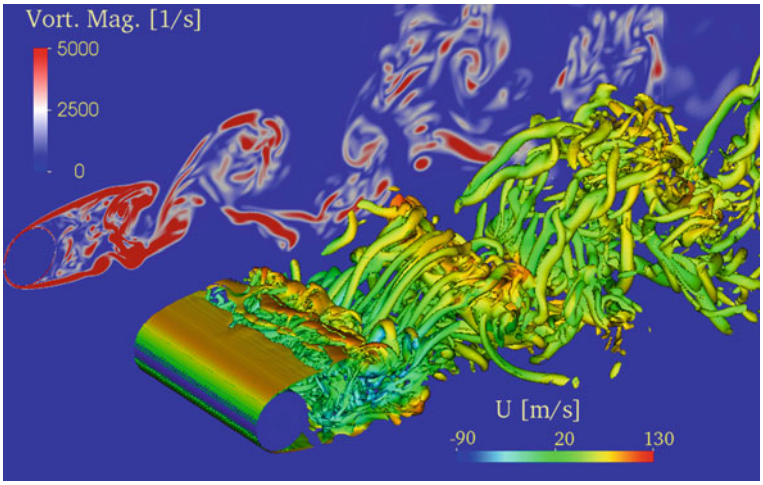


Fig. 10.8 Vortex structures' visualization by iso-surface of Q-criterion colored with streamwise velocity

stream $k^{sgs} = 0$. Without the correction, k^{sgs} will remain zero in the wake resulting in insufficient dissipation at small scales.

Statistics are collected for 100 non-dimensionalized time units, $t = d/u_\infty$. In Fig. 10.10, the average pressure coefficient $\overline{C_p}$ and the skin friction coefficient $\overline{C_f}$ are plotted over the surface of the cylinder. The data was averaged in space and also along the z-direction. Excellent agreement is obtained for the point of separation and pressure coefficient data. The skin friction coefficient is also matching well with the past data. Also, note the smoothness in the pressure and skin friction coefficient. To the best of the author's knowledge, such a smooth reconstruction, especially in the skin friction coefficient has never been shown in any of the past IB studies. Overall, the Cut-cell-AMRLES approach captures the near-wall solution very well. There are some oscillations in the skin friction coefficient plot around 50°. These oscillations indicate that the flux reconstruction in the cell present in these regions is not accurate. More investigation is needed to ascertain the source of these numerical artifacts, but the current hypothesis is that the cell clustering and thus the polynomial reconstruction are affected because of some degenerated small cells. Nevertheless, in other regions, the skin friction coefficient distribution is smooth.

To further assess the performance of the subgrid closure, the time-averaged plots of various flow- and closure-related quantities are presented in Fig. 10.11. The streamline plots along with the streamwise velocity contours clearly show two recirculation bubbles in the back of the cylinder which are close to symmetric with respect to the streamwise direction. The reattachment length from for bubble is around two diameters which matches with observations from past experimental studies. From the figure, the generation of k^{sgs} in the free shear layer formed from the boundary layer separation is clearly seen in the mean sense. The closure model parameters C_ν, C_ϵ

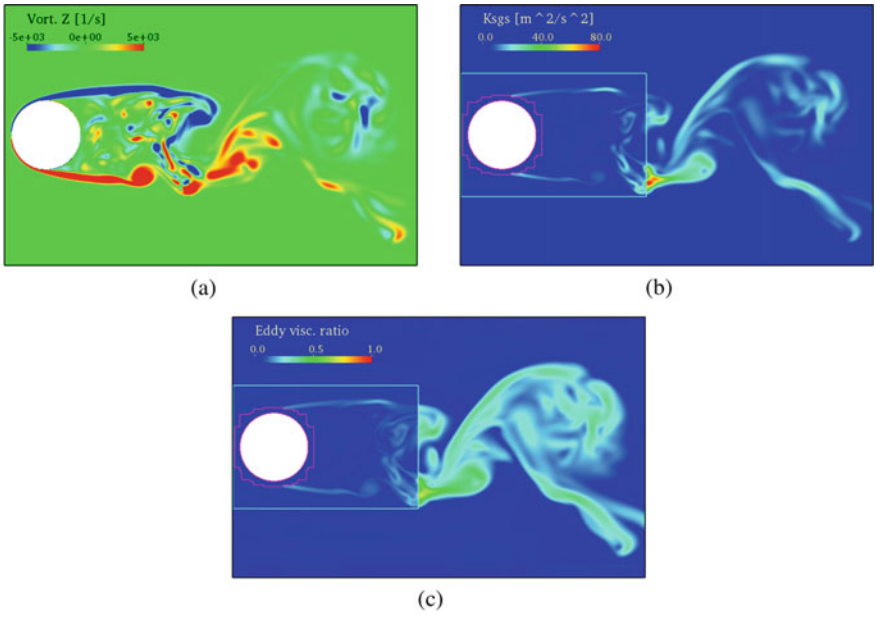


Fig. 10.9 Instantaneous snapshot of **a** vorticity magnitude, **b** subgrid kinetic energy, and **c** eddy viscosity ratio in the center x-y plane

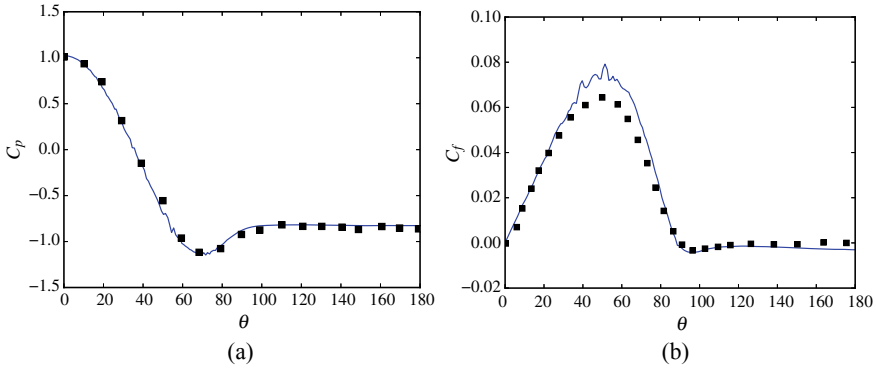


Fig. 10.10 Time and spatially averaged (in homogeneous direction) data of **a** pressure coefficient $\overline{C_p}$ and **b** skin friction coefficient for $\mathrm{Re}_d = 3900$ flow past cylinder. Block dots in **a** represent data from a past experimental study (Norberg 1987) and **b** represent data from a body-fitted LES (Ranjan and Menon 2015)

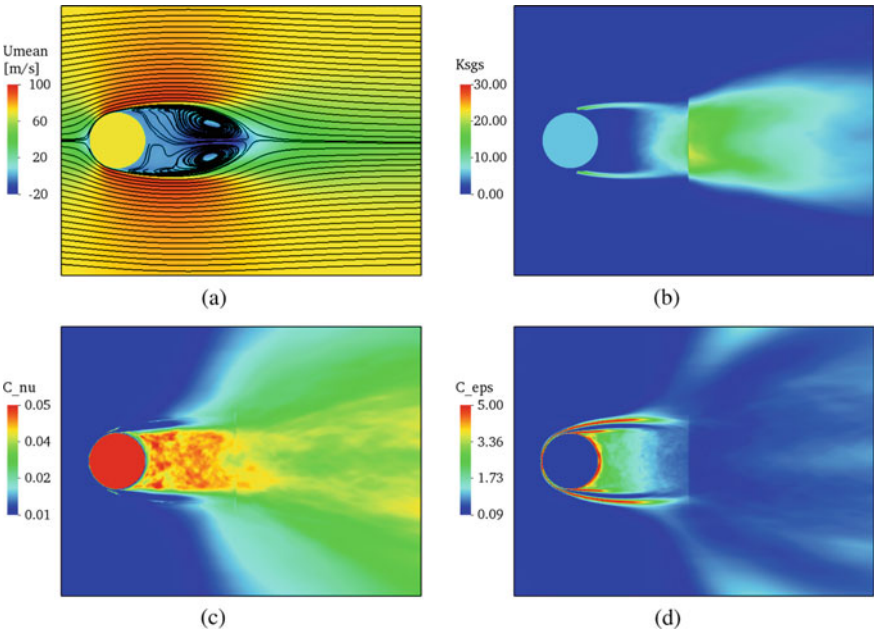


Fig. 10.11 Time-averaged plots of **a** streamwise velocity, **b** subgrid kinetic energy, **c** LDKM parameter C_ν, and **d** LDKM parameter C_ϵ in center x-y plane

that are computed dynamically show a wide variation, especially in the wake region. As expected there is a significant increase in the value of parameter $\overline{C_\nu}$ in the wake region where the large-scale vortex structures breakdown and flow become turbulent. The increase in C_ν in turn increases the contribution of the subgrid stress to the momentum equation through Eq. (10.4). The plot of the time-averaged C_ϵ shows high values near the boundary and the shear layer. Downstream of the cylinder in the turbulent wake, the value of the $C\epsilon$ drops. Since this parameter is a scaling coefficient for the model of dissipation of subgrid turbulent kinetic energy, a high value of $C\epsilon$ implies increased subgrid dissipation in the near-wall region and shear layer.

The quality of the wake predictions by the Cut-cell-AMRLES approach is assessed by comparing the mean streamwise velocity along the centerline of the cylinder with previous data in Fig. 10.12. Overall, the velocity deficit and recovery post reattachment is captured well in the current simulation. But it appears that the length of the recirculation bubble is over-predicted which is causing a delayed reattachment. Since the near-wall predictions are in excellent agreement with past data, the reason for this discrepancy is suspected to be mainly because of lack of convergence of the temporal statistics. A previous study (Ranjan and Menon 2015) performed time averaging after 700 non-dimensionalized time units for an interval of 250 time units, whereas in the current study, time statistics were collected after 150 time units for

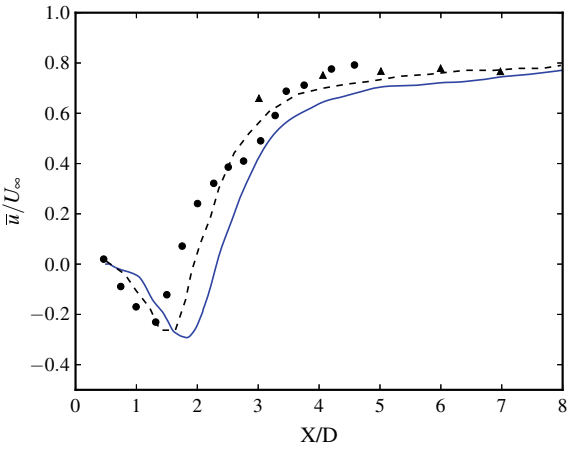


Fig. 10.12 Time-averaged streamwise velocity along the cylinder centerline. Blue solid line—Cut-cell-AMRLES, black dotted line—body-fitted LES (Ranjan and Menon 2015), black filled dots—experimental (Shih et al. 1993), black filled triangles—experimental (Ong and Wallace 1996)

only an additional 100 time units. The wake predictions are expected to improve with collection of more time-averaged data.

10.3.3 LES of Re_d = 3700 Flow Past a Sphere

Simulations of flow past spheres can be quite challenging with traditional body-conformal structured grid methods mainly because of the complexity involved in generating a good quality mesh especially near the wake region. Here, the Cut-cell-AMRLES approach is employed to simulate the turbulent flow of Re_d = 3700 over a sphere of diameter, d. The simulations are performed in a rectangular domain of size $30d \times 30d \times 30d$ with a base resolution of $(150 \times 150 \times 150)$. The AMR levels and grid resolution are kept same as the previous $\mathrm{Re}_d = 3900$ study as the Reynolds numbers are comparable. The plot of the AMR refinement for the sphere is shown in Fig. 10.13. Characteristic-based subsonic inflow is used in the left boundary, while subsonic outflow condition is prescribed to all the other boundaries. DNS simulation of the same Reynolds number has been performed in the past (Rodriguez et al. 2011) using an body-conformal unstructured approach.

The time history of the drag and lift coefficient plots is shown in Fig. 10.14. The average value of the drag coefficient is found to be $\overline{C_d} = 0.38$. This is close to the value of $\overline{C_{d,\mathrm{DNS}}} = 0.39$ predicted by the DNS study. To visualize the vortex structures in the wake of the sphere, the iso-surface of Q-criterion colored with streamwise velocity is shown in Fig. 10.15. Similar to the cylinder case, the boundary separates from the sphere surface and forms shear layer envelope which breaks down rapidly into small-scale turbulence within a couple of diameters downstream. Due to the three-dimensional nature of the free shear layer, the break down to small-scale turbulence is much faster compared to flow past a cylinder.

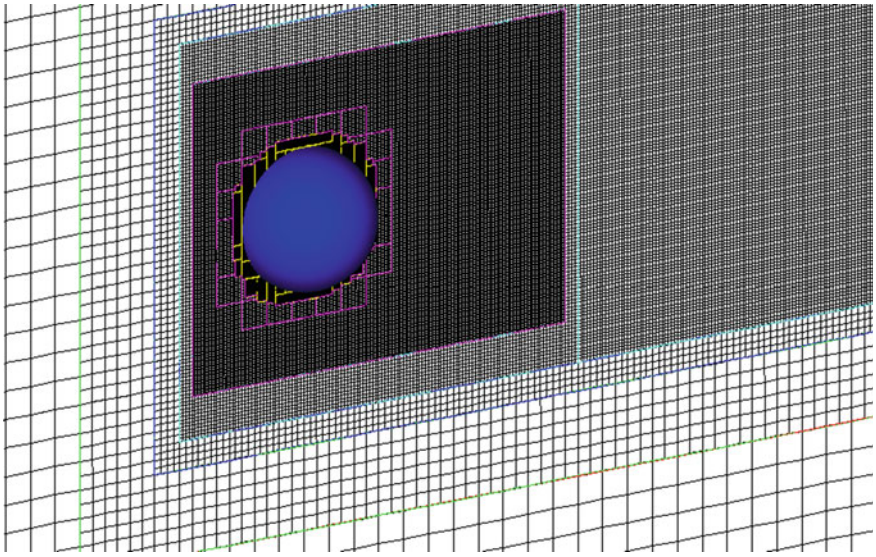


Fig. 10.13 Snapshot of local mesh refinement near surface for $\mathrm{Re}_d = 3700$ flow past a sphere

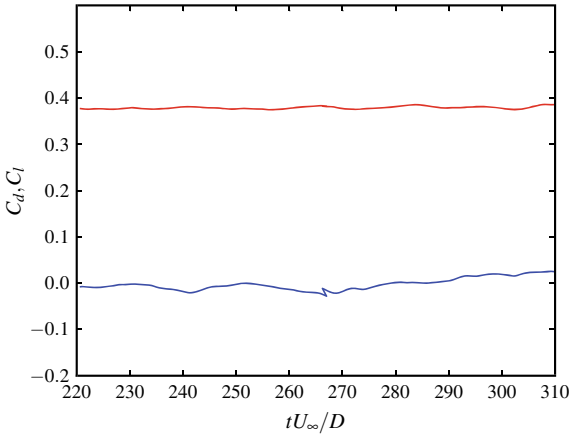


Fig. 10.14 Time history of drag (C_d) and lift (C_l) coefficient of $\mathrm{Re}_d = 3700$ flow past a sphere

The instantaneous snapshots of subgrid kinetic energy and eddy viscosity ratio are shown in Fig. 10.16. It can be seen from the figure that the behavior of the various flow-field quantities is similar to the previous case of flow past cylinder. The multi-

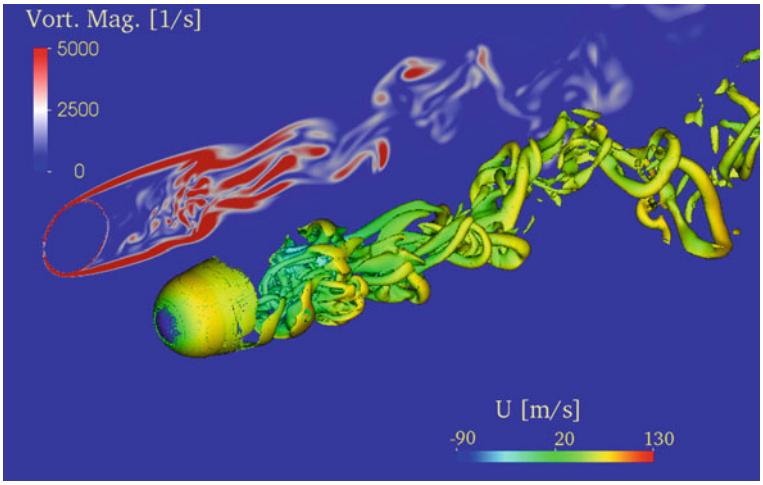


Fig. 10.15 Vortex structures visualization by iso-surface of Q-criterion colored with streamwise velocity

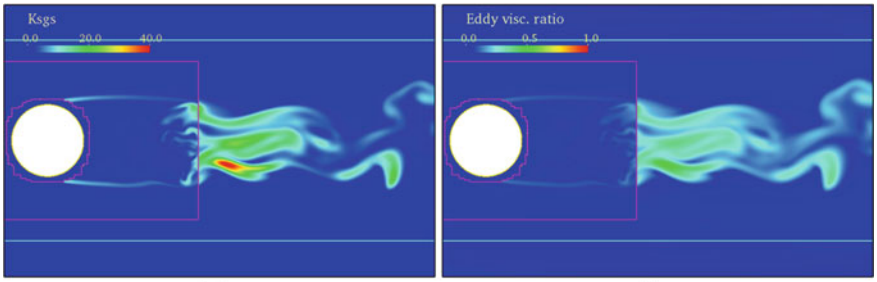


Fig. 10.16 Instantaneous snapshot of **a** subgrid kinetic energy and **b** eddy viscosity ratio in the center *x*-*y* plane

level closure injects k^{sgs} when near-wall refinement ends into the shear layer. A jump in the subgrid kinetic energy and the eddy viscosity is observed after a fine/coarse AMR interface.

The data is time averaged over 100 non-dimensionalized time units. The plots of the average pressure coefficient $\overline{C_p}$ and the skin friction coefficient $\overline{C_f}$ extracted along the midplane of the sphere are presented in Fig. 10.17. The current results show excellent agreement with the data from DNS and an experimental study for the C_p, C_f, the back pressure and the point of separation. Again it has to be reiterated that to the best of the author's knowledge, such a good match has never been reported in addition to smooth reconstruction of pressure and especially skin friction coefficient, in any of the past studies employing an embedded boundary technique. The contour plot of the pressure distribution on the sphere surface is shown in Fig. 10.18.

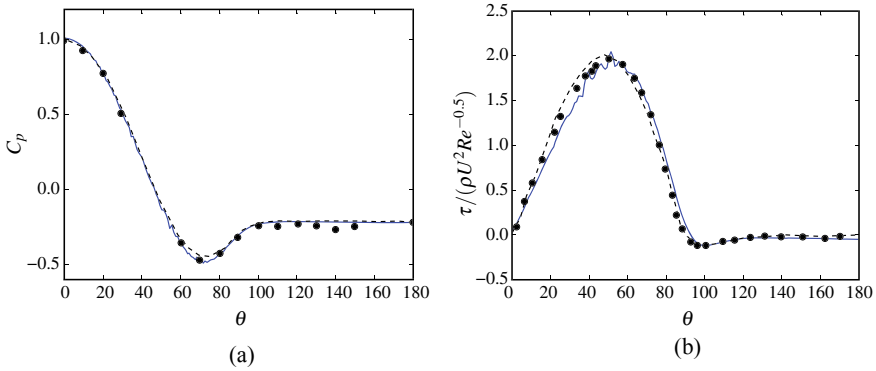


Fig. 10.17 Time-averaged data of **a** pressure coefficient $\overline{C_p}$ and **b** skin friction coefficient for Re_d = 3700 flow past cylinder. Solid blue line- Cut-cell-AMRLES Block dots in **a** represent data from a past experimental study (Norberg 1987) and **b** represent data from a body-fitted LES (Ranjan and Menon 2015)

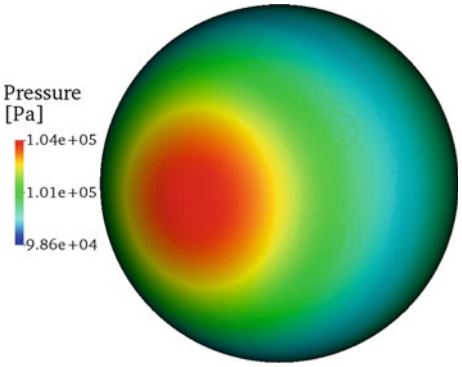


Fig. 10.18 Smooth distribution of pressure on the sphere surface for $\mathrm{Re}_d = 3700$ flow past a sphere

The wake predictions are assessed by comparing the mean streamwise velocity and the RMS of streamwise velocity along the centerline with previous DNS results in Fig. 10.19. The velocity deficit and recovery post reattachment are captured well in the current simulation. The magnitude of the RMS of streamwise velocity is slightly over-predicted, but the peak locations match well with the DNS data. As for the cylinder study, the wake predictions are expected to improve with collection of more time-averaged data.

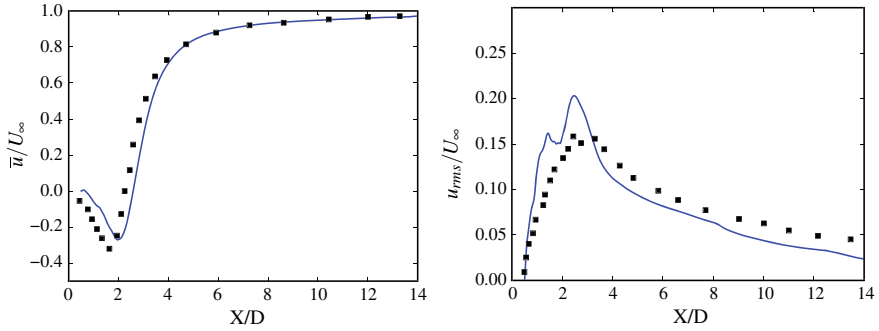


Fig. 10.19 Time-averaged streamwise velocity along the cylinder centerline. Blue—Cut-cell-AMRLES, Black dots—Body-fitted unstructured DNS (Rodriguez et al. 2011)

10.4 Conclusions

We have presented a multi-level subgrid closure model suited to a cut-cell-based EB approach for LES of compressible flow problems. In the proposed framework, the multi-level formalism of the block-structured refinement is exploited to build a two-layer closure model. At the finest level comprising of the cut-cells used to represent the embedded boundary, all the turbulence length scales are resolved and therefore no subgrid closure is employed. The multi-level correction recently developed for AMRLES is then applied to provide the filtered subgrid kinetic energy to the underlying coarser grids. A key advantage of the approach is that the multi-level correction of the subgrid kinetic energy (k^{sgs}) naturally introduces subgrid turbulence into the coarser grids thus facilitating essentially a multi-level boundary condition for k^{sgs} at the embedded boundaries. The cut-cell-AMRLES framework thus builds on the earlier works of the authors (Muralidharan and Menon 2016, 2019) combining a high-order cut-cell EB approach with the AMRLES subgrid closure.

To demonstrate the accuracy of the cut-cell method, grid convergence studies are presented for a 2D elliptic problem. The error analysis performed indicates that the method achieves the design order of accuracy both locally and globally. The cut-cell-AMRLES approach is then applied to study transitional turbulent flow past a cylinder and sphere. Results show that there is a good agreement of the pressure and skin friction coefficient data with past studies. The streamwise velocity and its fluctuation are also compared well with the past data. The detailed analysis of the various turbulent model parameters presented indicates that the proposed model behavior is consistent and addresses some of the problems faced in the past related to LES of AMR with embedded boundaries. For high Reynolds number fully turbulent flow problems, resolving the near-wall turbulent can become considerably more expensive since the adaptive refinement is isotropic. To handle such high Reynolds number flow regimes, integration of the cut-cell-AMRLES approach with wall-modeled LES (WMLES) (Kawai and Larsson 2012) can be attempted is a part of the future work.

Acknowledgements This work was supported in part by the U.S. Air Force Research Laboratory (AFRL), Eglin Air Force Base through grant AFB FA8651-15-1-0 0 07, and by the Defense Threat Reduction Agency (DTRA) through grant HDTRA1-14-1-0034. Computational support provided by the DoD HPC Centers at the Air Force Research Laboratory (Wright Patterson AFB, Ohio) and the Engineer Research and Development Center (Vicksburg, Mississippi) is gratefully acknowledged.

References

Berger M, Colella P (1989) Local adaptive mesh refinement for shock hydrodynamics. J Comput Phys 82:64–84

Berger M, Aftosmis M (2012) Progress towards a Cartesian cut-cell method for viscous compressible flow. In: 50th AIAA conference, Nashville, TN, p 1301

Cecere D, Giacomazzi E (2014) An immersed volume method for large eddy simulation of compressible flows using a staggered-grid approach. Comput Methods Appl Mech Eng 280:1–27

Chakravarthy V, Menon S (2001) Large eddy simulation of turbulent premixed flames in the flamelet regime. Combust Sci Technol 162:175–222

Clarke D, Salas M, Hassan H (1986) Euler calculations for multi-element airfoils using Cartesian grids. AIAA J 24:1128–1135

Coirier W, Powell K (1996) Solution-adaptive Cartesian cell approach for viscous and inviscid flows. AIAA J 34(5):938–945

Génin F, Menon S (2010) Dynamics of sonic jet injection into supersonic crossflow. J Turbul 11:1–30

Hartmann D, Meinke M, Schröder W (2008) An adaptive multilevel multigrid formulation for Cartesian hierarchical grid methods. Comput Fluids 37:1103–1125

Hartmann D, Meinke M, Schröder W (2011) A strictly conservative Cartesian cut-cell method for compressible viscous flows on adaptive grids. Comput Methods Appl Mech Eng 200:1038–1052

Ivan L, Groth C (2014) High-order solution-adaptive central essentially non-oscillatory (CENO) method for viscous flows. J Comput Phys 257(A):830–862

Kawai S, Larsson J (2012) Wall-modeling in large eddy simulation: length scales, grid resolution, and accuracy. Phys Fluids 24(1):015105

Kim C-S (2001) An immersed-boundary finite volume method for simulations of flow in complex geometeries. J Comput Phys 171:132–150

Kravchenko AG, Moin P (1997) On the effect of numerical errors in large eddy simulations of turbulent flows. J Comput Phys 131(2):310–322

MacCormack R (2003) The effect of viscosity in hypervelocity impact cratering. J Spacecraft Rockets 40(5):757–763

Majumdar S, Iaccarino G, Durbin P (2001) RANS solver with adaptive structured boundary non-conforming grids. Annu Res Briefs Cent Turb Res, pp 353–364

Merlin C, Domingo P, Vervisch L (2012) Immersed boundaries in large eddy simulation of compressible flows. Flow Turbul Combust 90(1):29–68

Meyer M, Devesa D, Hickel S, Hu X, Adams N (2010) A conservative immersed interface method for large eddy simulation for incompressible flows. J Comput Phys 229:6300–6317

Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261

Muralidharan B, Menon S (2016) A high-order adaptive cartesian cut-cell method for simulation of compressible viscous flow over immersed bodies. J Comput Phys 321:342–368

Muralidharan B, Menon S (2018) Simulation of moving boundaries interacting with compressible reacting flows using a second-order adaptive cartesian cut-cell method. J Comput Phys 357:230–262

Muralidharan B, Menon S (2019) A consistent multi-level subgrid scale closure for large eddy simulation of compressible flow using adaptive mesh refinement. Comput Fluids 180:159–175

Norberg C (1987) Effects of Reynolds number and a low-intensity freestream turbulence on the flow around a circular cylinder, vol 87. Chalmers University, Goteborg, Sweden, Technological Publications, p 2

Ong L, Wallace J (1996) The velocity field of the turbulent very near wake of a circular cylinder. Exp Fluids 20(6):441–453

Osher S, Fedkiw R (2003) Level set methods and dynamic implicit surfaces. Applied mathematical science. Springer, New York,

Osher S, Sethian J (1988) Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J Comput Phys 79:12–49

Ranjan R, Menon S (2015) On the application of the two-level large-eddy simulation method to turbulent free-shear and wake flows. J Turbul 16(2):136–166

Rodriguez I, Borell R, Lehmkuhl O, Segarra CDP, Oliva A (2011) Direct numerical simulation of the flow over a sphere at Re= 3700. J Fluid Mech 679:263–287

Shih WCL, Wang C, Coles D, Roshko A (1993) Experiments on flow past rough circular cylinders at large reynolds numbers. J Wind Eng Ind Aerodyn 49(1–3):351–368

Son JS, Hanratty TJ (1969) Velocity gradients at the wall for flow around a cylinder at reynolds numbers from 5×10^3 to 10^5. J Fluid Mech 35(02):353–368

Toro E (2009) Riemann solvers and numerical methods for fluid mechanics, 3rd edn. Springer, Berlin

Udaykumar H, Shyy W, Rao M (1996) A mixed Eulerian-Lagrangian method for fluid flows with complex and moving boundaries. Int J Numer Methods 22:691–705

Yang G, Causon D, Ingram D (2000) Calculation of compressible flows about complex moving geometries using a three-dimensional Cartesian cut cell method. Int J Num Methods Fluids 33:1121–1151

Part III
Applications

Chapter 11
Recent Developments on Employing Sharp-Interface Immersed Boundary Method for Simulating Fluid–Structure Interaction Problems

Rajneesh Bhardwaj

11.1 Introduction

Understanding fluid–structure interaction (FSI) in several engineering and biological systems is crucial for their design and innovation. In engineering, vortex-induced or flow-induced vibration (VIV or FIV) of a structure subjected to a fluid flow could be used in several energy-harvesting devices. In biology, FIV of vocal folds in larynx, cardiac hemodynamics, fish propulsion, insect flying, etc., are some examples. The secondary flows generated due to the FSI could be useful to augment heat transfer in certain systems. Thermoregulation in elephants via flapping of their large ears is one such example.

The above-mentioned FSI problems are computationally challenging. In particular, such problems involve the treatment of moving fluid–structure interface in the fluid domain and complex three-dimensional shape of the structure and/or fluid domain. The moving interface is a result of large-scale flow-induced deformation. Structure modeling may involve geometric and/or material nonlinearity. For high-speed flows, the boundary layer on the deforming interface should be adequately resolved.

In general, Arbitrary-Eulerian-Lagrangian (ALE) and immersed boundary (referred to as IB, hereafter) methods have been used to tackle the computational challenges mentioned earlier. ALE utilizes body-fitted grids while IB uses a fixed/Cartesian grid, as compared in Fig. 11.1. The main difference between the two approaches is as follows. A distorted grid needs to remesh in the former method due to the large-scale motion of the structure while the fluid domain grid remains fixed in the IB method. In the IB method, the motion of the structure is tracked through marker points on the structure using a Lagrangian framework. Since the grid

R. Bhardwaj (✉)
Department of Mechanical Engineering, Indian Institute of Technology Bombay, Mumbai 400076, India
e-mail: rajneesh.bhardwaj@iitb.ac.in

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_11

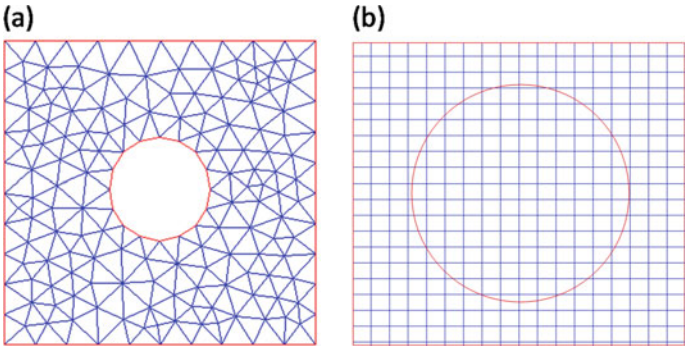


Fig. 11.1 **a** A structure conformal finite-element grid for ALE method. **b** A non-conformal Cartesian grid for IB method

generation is much easier in the IB method and the ALE method is prone to numerical errors due to the inherent remeshing procedure, the IB method is an attractive choice to simulate FSI problems. As mentioned in a notable review by Mittal and Iaccarino on the IB method (Mittal and Iaccarino 2005), the implementation of the boundary condition at the interface is realized by *continuous forcing* or *discrete forcing*. The forcing term is added as a source term in the Navier–Stokes equations before the discretization in the former approach. On the other hand, the forcing is applied after the discretization in the latter, also known as the sharp-interface IB method.

In this brief review, we discuss the applications of the sharp-interface IB method for computing challenging FSI problems. The layout of this chapter is as follows. A brief review of the sharp-interface IB method and coupling scheme of flow and structural solvers are presented in Sects. 11.2 and 11.3, respectively. We discuss the vortex-induced vibration of a cylinder in Sect. 11.4. FSI benchmarks of an elastic and viscoelastic splitter plate attached on a cylinder, involving large-scale flow-induced deformation, have been presented in Sects. 11.5 and 11.6, respectively. We discuss thermal augmentation in the FSI benchmark in Sect. 11.7. The flow-induced deformation of a thin elastic plate due to blast loading has been presented in Sect. 11.8 and concluding remarks are present in Sect. 11.9.

11.2 Sharp-Interface Immersed Boundary Method

In a typical IB method, the governing equations of the flow domain are solved on a fixed Cartesian (Eulerian) grid while the movement of the immersed structure surfaces is tracked in the Lagrangian framework. The flow is governed by unsteady, viscous and incompressible Navier–Stokes equations, expressed as follows,

$$\frac{\partial v_i}{\partial x_i} = 0,$$

$$\frac{\partial v_i}{\partial t} + \frac{\partial v_i v_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}} \frac{\partial^2 v_i}{\partial x_j^2}, \tag{11.1}$$

where $i, j = 1, 2, 3$, and v_i, t, p and Re are velocity components, time, pressure and Reynolds number, respectively. In a series of papers (Luo et al. 2008; Mittal et al. 2008, 2011; Zheng et al. 2010), Mittal and co-workers developed a multi-dimensional ghost-cell methodology in which the immersed structure boundary is represented using an unstructured grid with triangular elements in Cartesian volume grid of the flow domain. The following numerical methodology was adapted to solve the above governing equations in their work (Luo et al. 2008; Mittal et al. 2008, 2011; Zheng et al. 2010). The equations are discretized in space using a cell-centered, collocated (non-staggered) arrangement of primitive variables v_i, p and a second-order, central-difference scheme is used for all spatial derivatives. The unsteady Navier–Stokes equation is marched in time using a fractional-step scheme which involves two steps: solving an advection–diffusion equation followed by a pressure Poisson equation. During the first step, the convective terms are discretized using second-order Adam-Bashforth method while the viscous terms are treated implicitly using the Crank-Nicolson scheme to improve numerical stability. In this step, the modified velocity is given by (Mittal et al. 2008),

$$\frac{u_i^* - u_i^n}{\Delta t} + \frac{1}{2}(3N_i^n - N_i^{n-1}) = -\frac{1}{\mathrm{Re}} \frac{\delta p^n}{\delta x_i} + \frac{1}{2}(D_i^* + D_i^n), \tag{11.2}$$

where N_i and D_i are convective and diffusive terms, respectively. In the second step, the pressure Poisson equation is solved with the constraint that the final velocity is divergence-free. Once the pressure is obtained, the velocity field is updated to its final value in the final sub-step.

In this method (Luo et al. 2008; Mittal et al. 2008, 2011; Zheng et al. 2010), the surface of the immersed body is represented by an unstructured surface mesh which consists of triangular elements. Cells whose centers are located inside the structure are identified as *body*-cells and the other cells outside the structure are identified as *fluid*-cells (Fig. 11.2). A body-cell which has at least one fluid-cell as a neighbor is called a *ghost-cell* (Fig. 11.2). The kinematic boundary condition at the interface is prescribed by specifying an appropriate value at this ghost-cell.

While tackling a moving fluid–structure interface, the sharp-interface IB methods are generally prone to spurious pressure oscillations. As described in the literature (Mittal et al. 2008; Seo and Mittal 2011), these oscillations result from *fresh* and *dead* cells. The fresh (dead) cells are those fluid (solid) cells which were solid (fluid) cells in the previous time step. In order to tackle these oscillations, an extrapolation method for surface pressure has been utilized in the previous reports (Yang and Balaras 2006). In this context, Seo and Mittal (2011) employed a cut-cell-based discretization only to the pressure Poisson equation and velocity correction equations in a finite-difference-based sharp-interface IB method. The time step (Δt) is limited by the maximum CFL number ($\mathrm{CFL}_\mathrm{max}$), which is taken as 0.6 in the simulations. The expression of Δt is as follows,

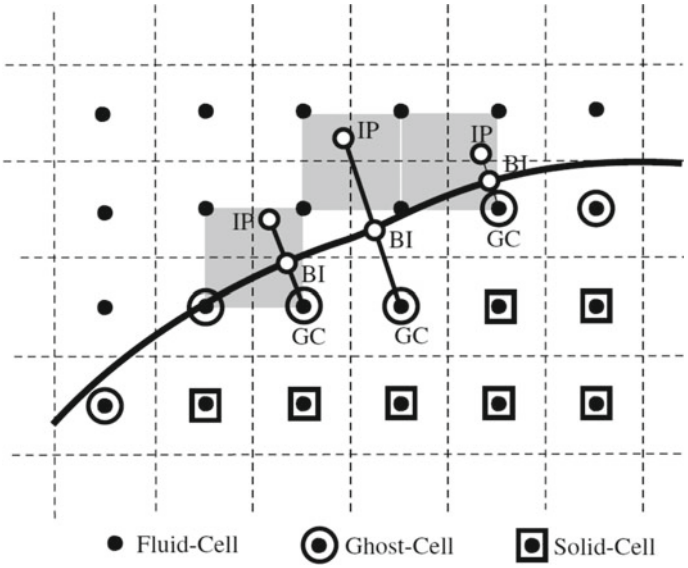


Fig. 11.2 Schematic of the ghost-cell method representing the fluid-cells, solid-cells and ghost-cells, originally described by Mittal et al. (2008). Reprinted with permission from Mittal et al. (2008). Copyright Elsevier (2008)

$$\left(\frac{u_1}{\Delta x_1}+\frac{u_1}{\Delta x_1}+\frac{u_1}{\Delta x_1}\right)\Delta t \leq \mathrm{CFL}_{\max} \tag{11.3}$$

11.3 Coupling of Flow and Structural Solver

The governing equations for the structure are the Navier equations (momentum balance equation in the Lagrangian form) and are written in non-dimensional form as:

$$\rho_s \frac{\partial^2 d_i}{\partial t^2} = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i, \tag{11.4}$$

where i and j range from 1 to 3, ρ_s is the dimensionless density of the structure with respect to the fluid density, d_i is the displacement component in the i direction, t is time, σ is the Cauchy stress tensor and f_i is component of the body force per unit volume in i direction. The coupling of flow and structural solver could be achieved by a monolithic or partitioned approach. As discussed by Bailoor et al. (2017), the governing equations for the flow and structure domains are discretized together in the former approach and the nonlinear system of equations is solved as a whole. In

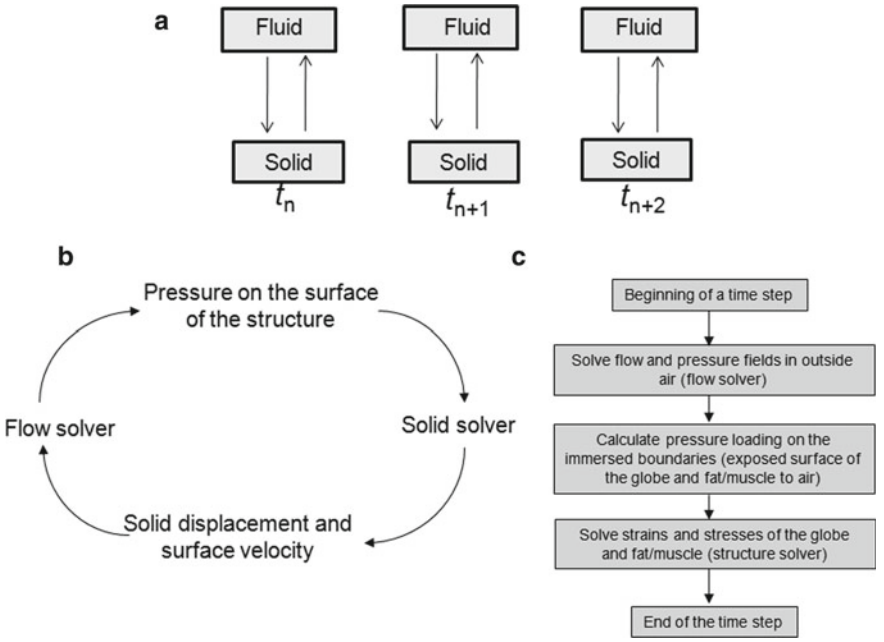


Fig. 11.3 **a** Partitioned approach, **b** data exchange between flow and solid solver and **c** algorithm of FSI solver. Reprinted with permission from Bhardwaj et al. (2014). Copyright Springer (2014)

the partitioned approach, an existing IB method could easily be coupled with another existing structural solver (Fig. 11.3a). The data exchange between the two solvers and a possible algorithm are shown in Fig. 11.3b, c, respectively. This approach was employed by Bhardwaj and Mittal (2012) and Bailoor et al. (2017). Both studies utilized an open-source finite-element solver Tahoe© as a structural solver and employed an implicit or two way coupling (Bhardwaj and Mittal 2012), which is numerically stable at low structure-fluid density ratio. In general, the implicit coupling is used if density ratio of structure to that of fluid is small. On the other hand, one way or explicit coupling is suited for the problem for which the density ratio is large and is computationally inexpensive.

11.4 Vortex-Induced Vibration (VIV) of a Cylinder

Vortex-induced vibration (VIV) of a cylinder is a classical FSI system, in which an elastically mounted cylinder oscillates due to vortex shedding past the cylinder. The coupled physics and associated characteristics of such a system are described in a notable review by Williamson and Govardhan (2004). Depending upon the application, VIV needs to be suppressed or agitated. The latter is useful in harnessing

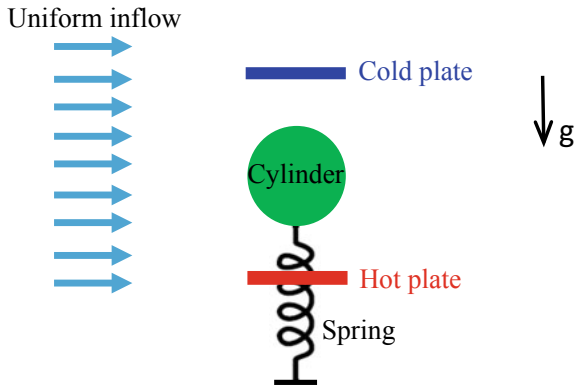


Fig. 11.4 Schematic of the VIV problem considered by Garg et al. (2018)

ambient wind or hydrokinetic energy. The modeling of VIV has been attempted successfully in the available reports. Yang et al. (2008) presented an embedded-boundary formulation with a strong coupling between the flow and structure solvers, and predicted VIV characteristics with a reasonable fidelity with the developed solver. Borazjani and Sotiropoulos (2009) successfully reported VIV characteristics of two tandem cylinders by using the IB method. Recently, Griffith and Leontini (2017) used a sharp-interface IB method for simulating the VIV. In their study, the spurious pressure oscillations decreased with an increase in grid refinement and increased with the cylinder velocity. Recently, Garg et al. (2018) used a sharp-interface IB method for a VIV system shown in Fig. 11.4 and demonstrated that the VIV of an elastically mounted cylinder can be achieved at a very low Reynolds number (Re) by invoking thermal buoyancy. The thermal buoyancy was invoked by two cold and hot parallel plates, kept in the direction of flow and symmetrically placed around the cylinder (Fig. 11.4). They showed that in the absence of the thermal buoyancy, the VIV does not occur for $Re = 20$ due to stable shear layers. By contrast, the thermal buoyancy induces flow instability and the vortex shedding helps to achieve the VIV at $Re = 20$ (Fig. 11.5). In a series of papers (Garg et al. 2018, 2019), authors showed that a significantly larger vibration amplitude of the cylinder over a wide range of reduced velocity can be obtained in the presence of the thermal buoyancy, as compared to that in the absence of the thermal buoyancy.

11.5 Flow-Induced Vibration (FIV) of a Thin Elastic Plate Attached to a Circular Cylinder

A FSI benchmark involving large-scale flow-induced deformation was first proposed by Turek and Hron (2006), in which an elastic plate is attached to the lee side of a rigid cylinder, as shown in Fig. 11.6a. The fluid is Newtonian and incompressible, while the structure is elastic and compressible. In this benchmark, the constitutive

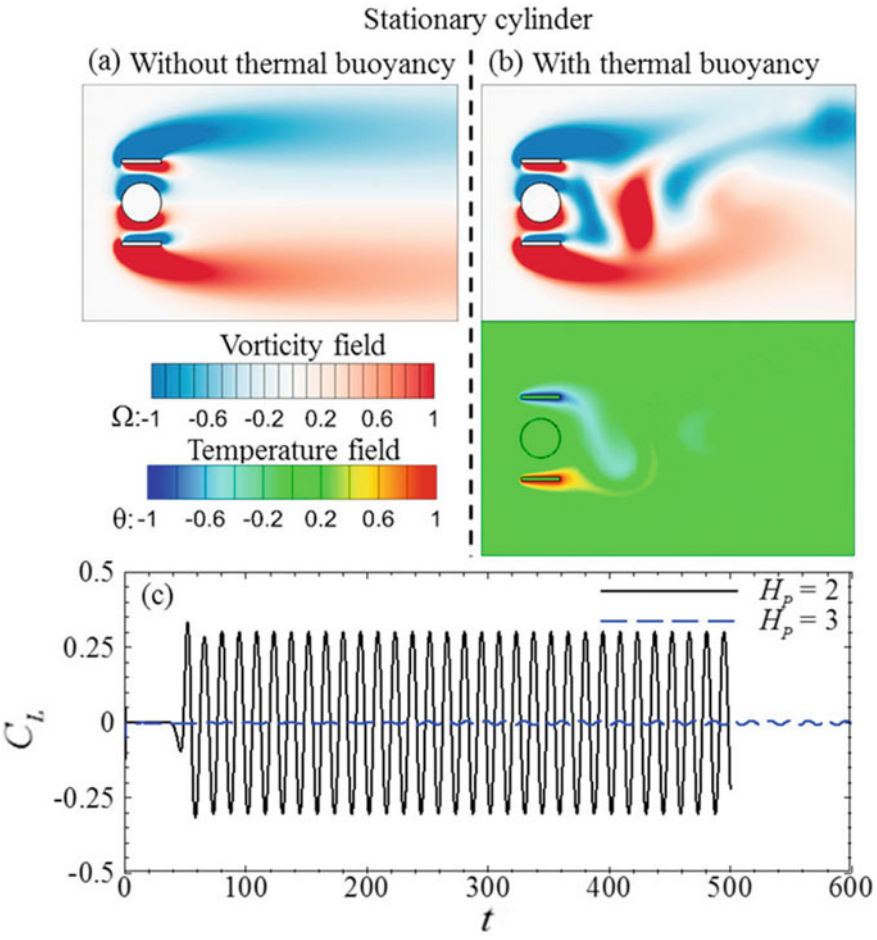


Fig. 11.5 Adapted from Garg et al. (Garg et al. 2018). **a** Vorticity field computed for an elastically mounted cylinder in the absence of the thermal buoyancy, **b** vorticity field and temperature field in the presence of the thermal buoyancy at $Re = 20$, $Pr = 7.1$, and $Ri = 1$, and **c** comparison of time-varying lift coefficient in the presence of the thermal buoyancy for different cases of distance between the two plates ($H_P = 2$ and $H_P = 3$). Reprinted with permission from Garg et al. (2018). Copyright American Institute of Physics (2018)

model for the structure is Saint Venant–Kirchhoff material and plane strain condition is considered. The computational domain and boundary conditions are shown in Fig. 11.6a. A parabolic inflow velocity profile is prescribed at the left boundary of the channel. No-slip boundary conditions are applied at the top and bottom walls. Zero normal gradient of velocity is applied at the outflow. The Reynolds number and dimensionless Young Modulus are defined as follows, $Re = U_{\text{mean}}D/\nu$ and $E = E^*/\rho_f U^2_{\text{mean}}$, where U_{mean} is mean inflow velocity, D is cylinder diameter, ν is kinematic viscosity of the fluid [m^2s^{-1}] and E^* is the Young Modulus [Nm^{-2}].

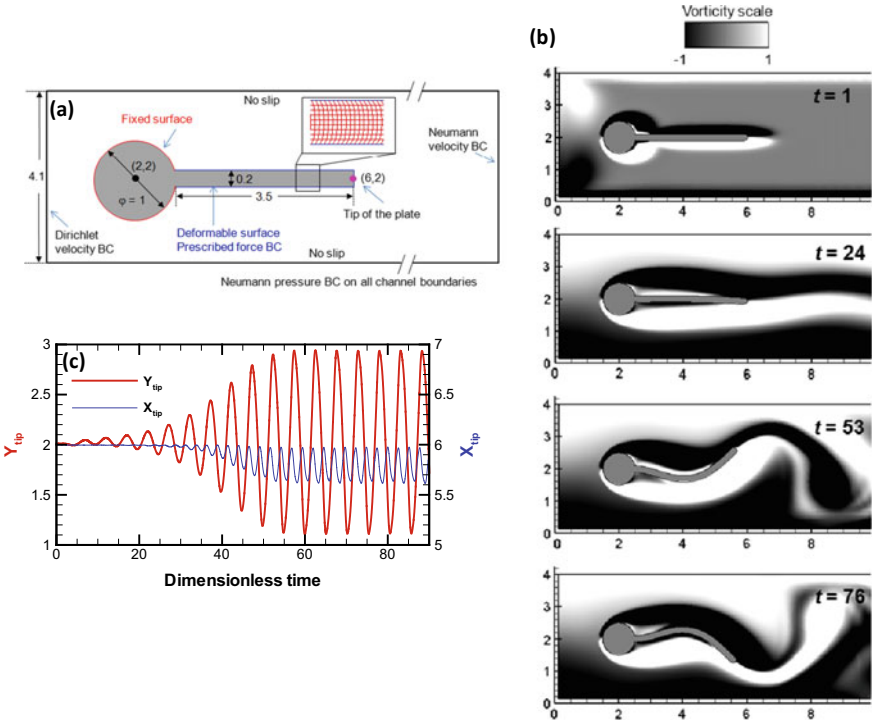


Fig. 11.6 Adapted from Bhardwaj and Mittal (2012). **a** Schematic and boundary conditions (BC) of the benchmark problem. All numbers shown are dimensionless with respect to the cylinder diameter. The inset shows the finite-element mesh for the plate. **b** The vorticity field and deformation of the elastic plate shown at different times for the baseline case. **c** Temporal variation of X and Y displacements of the tip of the plate. Reprinted with permission from Bhardwaj and Mittal (2012). Copyright Prof. R. Mittal (2011)

Bhardwaj and Mittal (2012) used a sharp-interface IB method to simulate the FSI benchmark proposed by Turek and Hron (2006) at $Re = 100$ and $E = 1400$. Figure 11.6b shows the calculated vorticity field and deformation of the plate at different time instances, reproduced from Bhardwaj and Mittal (2012). Vortices shed from the tip of the plate, with the frequency on the order of the oscillation frequency of the plate. The time-varying Y and X displacements of the tip of the plate are shown in Fig. 11.6c, which shows that the plate attains self-sustained periodic oscillation with plateau amplitude after a short time.

11.6 Flow-Induced Vibration (FIV) of a Thin Viscoelastic Plate Attached to a Circular Cylinder

Recently, Mishra et al. (2019) used a sharp-interface immersed boundary method to simulate the response of a viscoelastic plate attached on a circular cylinder (Fig. 11.7, top). Extending the FSI benchmark by Turek and Hron (2006), authors used the standard linear solid (SLS) model is used to represent the viscoelasticity of the plate. They showed the effect of two parameters on the response of the plate, namely, ratio of non-equilibrium to equilibrium Young's modulus (R) and ratio of dimensionless material damping to dimensionless non-equilibrium Young's modulus (τ). The dimensionless amplitude of the tip displacement is found to be a non-monotonic

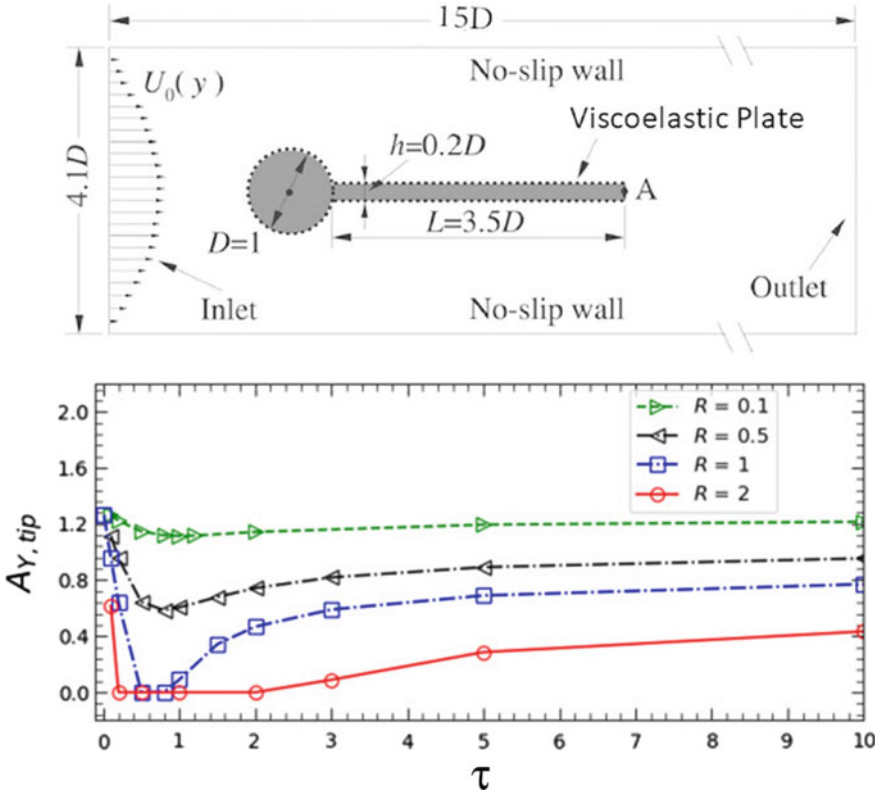


Fig. 11.7 Adapted from Mishra et al. (2019). (Top) Schematic of flow domain with details of boundary conditions. The FSI benchmark, proposed by Turek and Hron (2006), was extended to account for viscoelasticity of the plate. (Bottom) Variation of amplitude of the tip of the viscoelastic plate with τ. Reprinted with permission from Mishra et al. (2019). Copyright Elsevier (2019)

function of τ (Fig. 11.7, bottom). Their numerical results show that these two parameters could help either to suppress or to agitate the plate displacement. The latter has applications in designing energy-harvesting devices.

11.7 Thermal Augmentation Using FIV of a Thin Flexible Plate

The secondary flows generated during the interaction of flexible structures with the fluid flow could be used to improve heat transfer between the fluid and a heated solid. Such a method does not require any active control of the structure and could improve the thermal performance of an FSI system. Soti et al. (2015) employed a sharp-interface IB method to numerically demonstrate large-scale flow-induced deformation as an effective passive heat transfer enhancement technique. They showed several validations of convective heat transfer module of the FSI solver, which utilizes the IB method. They considered convective heat transfer in the FSI benchmark problem, proposed by Turek and Hron (2006). Figure 11.8 (top and bottom) plots the vorticity and isotherms in the channel with the heated walls at different time instances, respectively. Soti et al. (2015) pointed out that the wake vortices generated due to the self-sustained motion of the plate sweep higher sources of vorticity generated on the heated walls in the bulk of the fluid [Fig. 11.8 (top)]. This causes an effective mixing in the bulk fluid and thereby improves convective heat transfer at the wall.

11.8 Blast Loading on a Thin Flexible Plate

Coupled physics of the loading or interaction of a blast wave with human eyes or brain is crucial to understand the mechanism of injury and associated risks to the eyes or brain. The sharp-interface IB method has been successfully employed to simulate the blast loading on the human eye by Bhardwaj et al. (2014), wherein a partitioned approach was used to strongly couple the IB flow solver with an open-source finite-element structure dynamics solver. In a follow-up study, Bailoor et al. (2017) proposed an FSI solver based on the sharp-interface IB method for the interaction of compressible flow with a flexible structure. The flow solver was based on a higher-order finite-difference method using a Cartesian grid and employed ghost-cell methodology. Authors showed higher-order accuracy near the immersed boundary by combining the ghost-cell approach with a weighted least squares error method and validated the solver with previous measurement of displacement of a thin elastic steel panel subjected to blast loading in a shock tube (Giordano et al. 2005), as shown in Fig. 11.9a, b. The oscillating behavior of the tip of the panel and the computed shock wave propagation (Fig. 11.9c) was found in good agreement with the published results.

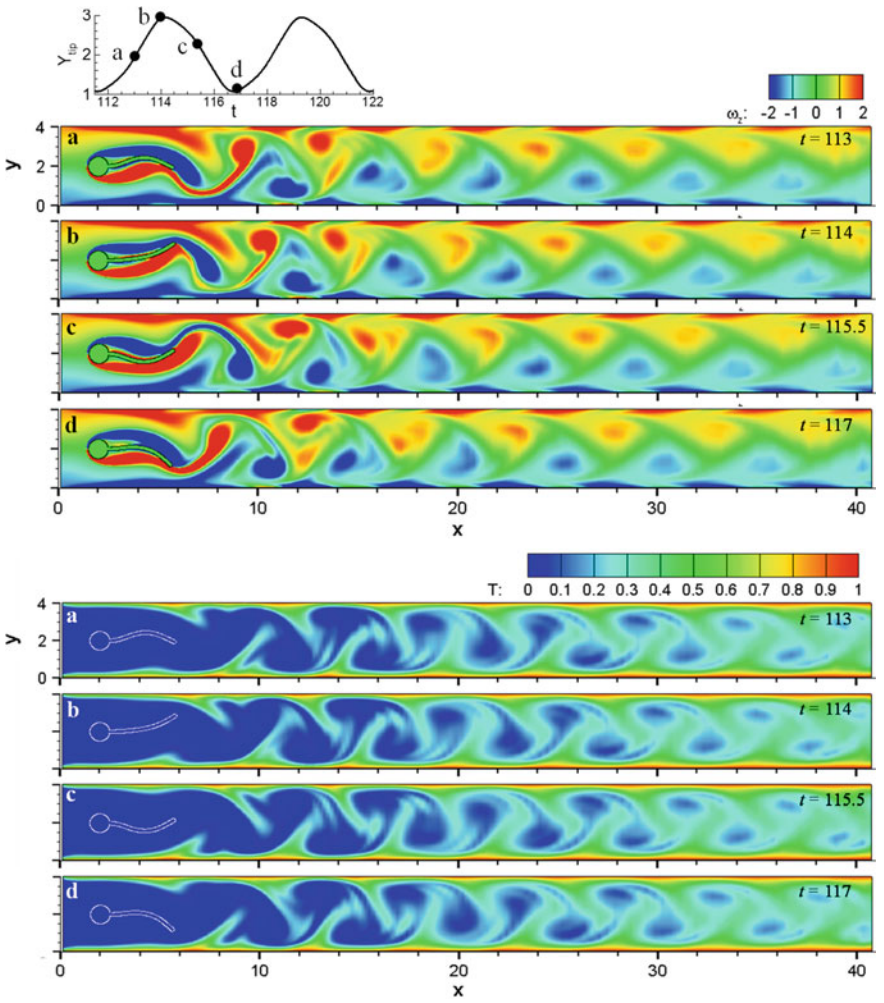


Fig. 11.8 Adapted from Soti et al. (2015). Vorticity contours (top) and isotherms (bottom) in a channel with a cylinder attached to an elastic plate at different time instances for $Re = 100$. The time instances are shown in the inset as black dots on the time-varying position of the tip of the plate, during a typical cycle of the plate oscillation. Reprinted with permission from Soti et al. (2015). Copyright Elsevier (2015)

11.9 Conclusions

A brief review of recent developments on employing a sharp-interface immersed boundary method for simulating fluid–structure interaction (FSI) problems was presented. First, a brief comparison of Arbitrary-Eulerian-Lagrangian (ALE) and

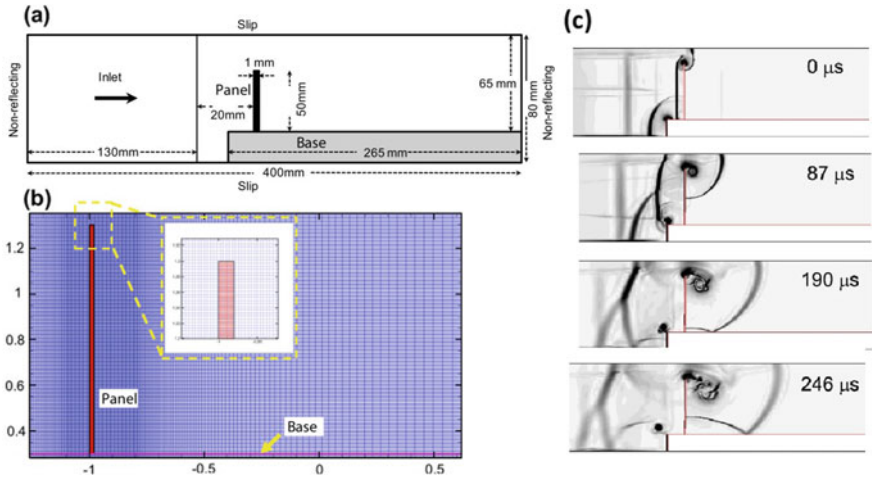


Fig. 11.9 Adapted from Bailoor et al. (2017). **a** Computational domain used in the simulation setup for the FSI benchmark problem, proposed by Giordano et al. (2005). **b** Non-uniform Cartesian mesh with grid stretching in the downstream direction employed. A finite-element mesh used in the panel and its surrounding Cartesian mesh are shown in the inset. **c** Computed shock propagation during the shock impact on the elastic panel. Snapshots of the contours of the magnitude of computed density gradient (numerical schlieren) are shown. Reprinted with permission from Bailoor et al. (2017). Copyright Elsevier (2017)

immersed boundary (IB) method was discussed. A general framework of sharp-interface IB method and scheme to strongly couple flow and structural solvers were presented. The following applications were discussed: Vortex-induced vibration of a cylinder (VIV), FSI benchmark of an elastic splitter attached on a cylinder involving large-scale flow-induced deformation, an extension of the FSI benchmark to a viscoelastic plate, convective heat transfer augmentation in the FSI benchmark and flow-induced deformation of a thin plate by blast loading. Overall, the sharp-interface IB method can handle large-scale flow-induced deformation of the structure and is able to successfully simulate the coupled fluid and structural dynamics of an FSI system with reasonable fidelity.

Acknowledgements R. B. gratefully acknowledges financial support from Naval Research Board (NRB), New Delhi, India through grant NRB-403/HYD/17-18. R. B. thanks the following Ph.D. students and technical staff for their contribution to the development and testing of immersed boundary method based in-house fluid–structure interaction solver: Dr. Atul Kumar Soti, Dr. Anup Kundu, Dr. Hemanshul Garg, Mr. Rahul Mishra and Mr. Shantanu Bailoor.

References

Bailoor S, Annangi A, Seo JH, Bhardwaj R (2017) Fluid–structure interaction solver for compressible flows with applications to blast loading on thin elastic structures. Appl Math Model 52:470–492

Bhardwaj R, Mittal R (2012) Benchmarking a coupled immersed-boundary-finite-element solver for large-scale flow-induced deformation. AIAA 50:1638–1642

Bhardwaj R, Ziegler K, Seo JH, Ramesh KT, Nguyen TD (2014) A computational model of blast loading on the human eye. Biomech Model Mechanobiol 13:123–140

Borazjani I, Sotiropoulos F (2009) Vortex-induced vibrations of two cylinders in tandem arrangement in the proximity–wake interference region. J Fluid Mech 621:321–364

Garg H, Soti AK, Bhardwaj R (2018) A sharp interface immersed boundary method for vortex-induced vibration in the presence of thermal buoyancy. Phys Fluids 30:023603

Garg H, Soti AK, Bhardwaj R (2019) Vortex-induced vibration of a cooled circular cylinder. Phys Fluids 31:083608

Giordano J, Jourdan G, Burtschell Y, Medale M, Zeitoun DE, Houas L (2005) Shock wave impacts on deforming panel, an application of fluid-structure interaction. Shock Waves 14:103–110

Griffith MD, Leontini JS (2017) Sharp interface immersed boundary methods and their application to vortex-induced vibration of a cylinder. J Fluids Struct 72:38–58

Luo H, Mittal R, Zheng X, Bielamowicz SA, Walsh RJ, Hahn JK (2008) An immersed-boundary method for flow-structure interaction in biological systems with application to phonation. J Comput Phys 227:9303–9332

Mishra R, Kulkarni SS, Bhardwaj R, Thompson MC (2019) Response of a linear viscoelastic splitter plate attached to a cylinder in laminar flow. J Fluids Struct 87:284–301

Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261

Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227:4825–4852

Mittal R, Zheng X, Bhardwaj R, Seo JH, Xue Q, Bielamowicz S (2011) Toward a simulation-based tool for the treatment of vocal fold paralysis. Front Physiol 2

Seo JH, Mittal R (2011) A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations. J Comput Phys 230:7347–7363

Soti AK, Bhardwaj R, Sheridan J (2015) Flow-induced deformation of a flexible thin structure as manifestation of heat transfer enhancement. Int J Heat Mass Transf 84:1070–1081

Tahoe is an open source C++ finite element solver, which was developed at Sandia National Labs

Turek S, Hron J (2006) Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and a laminar incompressible flow. Notes in computational science and engineering. Springer, Berlin

Williamson C, Govardhan R (2004) Vortex-induced vibrations. Annu Rev Fluid Mech 36:413–455

Yang J, Balaras E (2006) An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. J Comput Phys 215:12–40

Yang J, Preidikman S, Balaras E (2008) A strongly coupled, embedded-boundary method for fluid–structure interactions of elastically mounted rigid bodies. J Fluids Struct 24:167–182

Zheng X, Xue Q, Mittal R, Beilamowicz S (2010) A coupled sharp-interface immersed boundary-finite-element method for flow-structure interaction with application to human phonation. J Biomech Eng Trans Asme 132

Chapter 12
Study of Momentum and Thermal Wakes Due to Elliptic Cylinders of Various Axes Ratios Using the Immersed Boundary Method

Immanuvel Paul, Venkatesh Pulletikurthi, K. Arul Prakash, and S. Vengadesan

12.1 Introduction

Fluid flow over the bluff bodies is studied to understand the fundamentals of the fluid mechanics. Such fundamental understanding is necessary for flow control and to achieve heat transfer increase and drag reduction. Laminar flow over canonical bodies like circular cylinders is an orthodox problem to understand the balance between the inertial and viscous forces. Researchers show interest on non-canonical bodies like elliptic cylinders because of their closeness to the real-life applications. The ratio of minor axis to the major axis is given as axis ratio (AR). An elliptic cylinder of AR = 1 is the circular cylinder.

Besides axis ratio, Reynolds number (Re) and angle of attack (AOA) also determine the flow behavior behind the elliptical cylinders. With increase in Reynolds number, the flow changes from laminar to turbulence. In this study, we focus on the laminar flow over elliptic cylinder. In laminar flows, the interesting phenomenon is the vortex shedding. The imbalance between the momentum and viscous effects leads to the separation of the boundary layer. The shear layer thus formed between the high momentum and low momentum fluid forms a pair of vortices. These vor-

I. Paul
Stanford University, Stanford, CA, USA
e-mail: ipaul@stanford.edu

V. Pulletikurthi
School of Mechanical Engineering, Purdue University, West Lafayette, IN, USA
e-mail: vpulleti@purdue.edu

K. Arul Prakash (✉) · S. Vengadesan
Department of Applied Mechanics, IIT Madras, Chennai 600036, India
e-mail: arulk@iitm.ac.in

S. Vengadesan
e-mail: vengdes@iitm.ac.in

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_12

tices shed alternatively from either sides of the cylinder. The shedded vortices form a vortex pair street behind the cylinder called as von Karman vortex street (Fig. 12.1).

The imbalance between momentum and viscous forces is determined by the critical Reynolds number (Re_{cr}). The frequency of vortices is defined by the non-dimensional critical Strouhal number (St_{cr}). Taneda (1959) observed that the vortex streets which are convected downstream break down because of the instability of mean velocity profile. The shedded vortices rearrange themselves parallel to each other and shed even far downstream to form a secondary vortex shedding similar to von Karman vortex street Inoue and Yamazaki (1999). Aref and Siggia (1981), Cimbala et al. (1988), Najjar and Balachandar (1998) studied this secondary vortex street behind the circular cylinder and flat plate.

Johnson et al. (2001) used two-dimensional spectral element method to study vortex structures behind elliptical bodies for Reynolds number range of 30 to 200. With the decrease in axis ratio, the shedding behind the elliptic cylinder changed from steady Karman vortex shedding to a surprisingly different flow with two distinct regions. The first region situated directly behind the cylinder contained two rows of vortices rolling up from the cylinder with a region of relatively dead flow in between. Johnson et al. (2004) carried out direct numerical simulation (DNS) for elliptic cylinders (flat plate to circular cylinder) for Re range of 75–175. They found out that the inception point of the low-frequency vortices in the far wake occurs closer to elliptical cylinder with increasing Re or decreasing in axis ratio (AR). From these, they suggested the low-frequency unsteadiness behind normal flat plates does not need to be due to the vortex interaction, but, rather, can result from the presence of a two-dimensional instability of the mean wake. Saha (2007) carried out a direct numerical simulation for a plate of AR, 0.125 for Re 30–175. It was concluded that the flow with unsteady near wake and far wake at $Re = 35$ shows an unsteady near wake but steady far wake for $75 \leq Re \leq 140$. The flow in the far wake is seen to undergo a transition when the Reynolds number is increased to $Re = 145$ and results in the secondary vortex structures. At $Re = 150$, the low-frequency vortex street in the far wake undulates at a very low frequency, which increases with increasing the Reynolds number, giving rise to a tertiary vortex pattern.

Thompson et al. (2014) documented the critical Reynolds numbers and Strouhal frequencies for the initial transition to unsteady periodic flow which have been accurately determined for elliptical cylinders of different aspect ratios for the two-dimensional shedding regime. They observed that as the cylinder axis ratio is

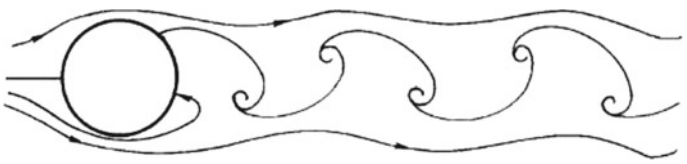


Fig. 12.1 A typical Karman vortex street behind an elliptic cylinder. Adapted from Kundu and Cohen (2008)

decreased, below 0.25, the wake deviates further from the standard Benard–von Karman state. For still smaller axis ratios, another three-dimensional quasi-periodic mode becomes unstable, leading to a different transition scenario. Ota et al. (1983) conducted experimental studies and observed that the average Nusselt number increases with Re and AR, but decreases with AOA. However, the relation between the momentum and thermal wakes is not clearly established. In this study, we aim to understand the relation between momentum and thermal wakes of an elliptic cylinder for various axis ratios using immersed boundary method. The reason for using the IBM is because of its ease in analyzing the flow for different axis ratios without much variation in the grid. Here, we show that the immersed boundary method effectively captures the flow physics not only for fluid structure interactions but also for flow over rigid bodies.

This chapter is organized as follows. Section 12.2 describes the immersed boundary method we used to simulate the flow and corresponding numerical details in subsections. The results are discussed in Sect. 12.3.

12.2 Immersed Boundary Method

Fluid flow past bluff bodies are simulated using the body-fitted grid methods (BFGM). The disadvantages of the BFGM are due to its time-consuming process in the grid generation and also the grid needs to be generated at every iteration for moving boundary problems. In order to address these issues, Cartesian grid methods (CGM) were introduced. In this method, the equations are solved in the Cartesian domain unlike BFGM. One such CGM is the immersed boundary method (IBM). This method was first used by Peskin (1972) to simulate cardinal flows. In IBM, Eulerian frame of reference is considered to solve the equations in fluid domain and the solid body is immersed on the Cartesian grid using Langrangian markers placed along the body. The effect of the solid body is incorporated using the forcing terms in the governing equations. The details of the equations and algorithm are given in Sect. 12.2.1. Though many variants of IBM are available now (Mittal and Iaccarino 2005; Pacheco et al. 2005; Su et al. 2007; Kim et al. 2001), the method adopted for this study uses a finite difference discretization on a staggered grid. An Eulerian grid is used for the fluid domain, and a set of Lagrangian marker points are used to represent the solid body. Interaction between Eulerian domain and Lagrangian marker points is achieved through a Dirac delta function. The solid body is modeled by a forcing term which is added to the governing equations of flow given as,

$$\nabla \cdot u = 0, \tag{12.1}$$

$$\frac{\partial u}{\partial \tau} + (u \cdot \nabla)u = -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\nabla^2 u + f, \tag{12.2}$$

$$\frac{\partial T}{\partial \tau} + (u \cdot \nabla)T = \frac{1}{\text{RePr}} + \nabla^2 T + q, \tag{12.3}$$

where f and q represent forcing terms for velocity and temperature for the solid boundary, τ—dimensionless time, Re—Reynolds number, Pr—Prandtl number = $\frac{c_p \mu}{k_t}$, k_t is the coefficient of thermal conductivity of fluid, c_p is the heat capacity, u and T are non-dimensional velocity vector and temperature, where $u = \frac{u}{u_\infty}$, $p = \frac{p}{\rho u_\infty^2}$, and $T = (t - t_\infty)/(t_w - t_\infty)$ for isothermal case, t is non-dimensionalized by $q_w d_H/\mu$, where u_∞—free stream velocity, ρ—density of fluid, μ—viscosity of fluid, and d_H is the hydraulic diameter of elliptic cylinder. Hydraulic diameter is taken as the characteristic length, which is defined as, $d_H = 4A/P$, where A is the area and P is the perimeter of the elliptic cylinder.

To impose boundary conditions on solid body, singular forces are applied on each Lagrangian marker points in such a way that, forcing will result in enforcement of required boundary condition. These singular forces are then distributed to the nearby Eulerian points so that the presence of solid body will be felt in the fluid domain. Thus, the singular forces applied on Lagrangian marker points determine the forcing term which is added to the governing equations.

12.2.1 Numerical Method

In the present study, an in-house solver based on immersed boundary method (IBM) is used to solve bluff body flow problem. The force due to the body is calculated using implicit method developed by Su et al. (2007). The Eulerian and Langrangian variables interaction is calculated using the 4-point regularized delta function. The algorithm and application of the Dirichlet boundary condition for one time step is as follows:

1. The momentum equation given in Eq. 12.2, was solved without considering the immersed boundary and the intermediate velocities, $u^*(x)$, at the Eulerian grid points are calculated as follows,

$$\frac{u^* - u^n}{\nabla \tau} = -\frac{3}{2}\nabla_h (uu)^n + \frac{1}{2}\nabla_h (uu)^{n-1} - \nabla_h P^n + \frac{1}{2Re}\nabla_h^2 (u^* + u^n). \tag{12.4}$$

   The convective and diffusive terms in momentum equation are discretized using Adams–Bashforth and Crank–Nicholson schemes, respectively, which provide overall second-order accuracy.
2. The velocities, $u^*(x_1)$, at the Lagrangian points are interpolated using nearby Eulerian points using the equation,

$$u_l^*(x_l) = \sum_x u^*(x)\delta_h(x - x_l)h^2. \tag{12.5}$$

3. The Langrangian forces, $f_l^*(x_l)$, are calculated as,

$$\sum_{j=1}^{M}\left(\sum_{x}\delta_h(x-x_l)\delta_h(x-(x_l)_j)\right)f_l^*(x_l)=\frac{u_l^{n+1}(x_l)-u_l^*(x_l)}{\Delta\tau}, \tag{12.6}$$

where M is the total number of Lagrangian points and is calculated from the prescribed boundary velocity, u_l^{n+1}, at each time step and the interpolated velocity is calculated in step 2.

4. The Lagrangian forces, $f_l^*(x_l)$, are distributed to nearby Eulerian points, $f^*(x)$,

$$f^*(x)=\sum_{k=1}^{M}f_l^*((x_l)_k)\delta_h(x-(x_l)_k)\Delta S, \tag{12.7}$$

where ΔS is the distance between adjacent Lagrangian points.

5. Intermediate velocities, $u^*(x)$, are corrected using the distributed force to get another intermediate velocity, u^{**},

$$\frac{u^{**}-u^*}{\Delta\tau}=f^*. \tag{12.8}$$

6. Solve the pressure Poisson equation

$$\nabla_h^2 p^*=\frac{(\nabla_h.u^{**})}{\nabla\tau} \tag{12.9}$$

The algebraic system of equations resulting from discretization of the pressure Poisson equation is solved using BiCGSTAB(2) (Van Der Vorst 2002), one of the most efficient elliptic equation solvers.

7. Correct the pressure, p, and velocities (Brown et al. 2001)

$$p^{n+1}=p^n+p^*-\frac{1}{2\mathrm{Re}}\nabla_h.u^{**} \tag{12.10}$$

$$\frac{u^{n+1}-u^n}{\nabla\tau}=-\nabla_h p^* \tag{12.11}$$

8. Now, the energy equation is solved using the velocities obtained from the above steps. The intermediate temperature, $T^*(x)$, at the Eulerian grid points (without considering immersed body boundary) are calculated as follows:

$$\frac{T^*-T^n}{\nabla\tau}=\frac{-3}{2}S_T^n+\frac{1}{2}S_T^{n-1}+\frac{1}{2\mathrm{RePr}}\nabla_h^2(T^*+T^n) \tag{12.12}$$

$$S_T=(u\cdot\nabla_h)T \tag{12.13}$$

Like momentum equations, the convective and diffusive terms in energy equations are discretized using Adams–Bashforth and Crank–Nicolson schemes, respectively, which provide overall second-order accuracy.

9. The temperature at the Langrangian points, $T_l^*(x_l)$, is interpolated using nearby Eulerian points

$$T_l^*(x_l) = \sum_x T^*(x)\delta_h(x - x_h)h^2 \tag{12.14}$$

10. The Lagrangian fluxes, $q_l^*(x_l)$, are calculated from prescribed boundary temperature, $T^{n+1}l(x_l)$, and interpolated temperature is calculated in step 9.

$$\sum_{j=1}^{M}\left[\sum_x \delta_h(x - (x_l)_k)\delta_h(x - x_l)\right] q_l^*(x_l) = \frac{T_l^{n+1}(x_l) - T_l^*(x_l)}{\nabla\tau} \tag{12.15}$$

11. The Lagrangian fluxes are distributed to nearby Eulerian points

$$q^*(x) = \sum_{k=1}^{M} q_l^*((x_l)_k)\delta_h(x - x_l)\Delta S \tag{12.16}$$

where ΔS is the distance between adjacent Lagrangian points.

12. Intermediate temperature, $T^*(x)$, is corrected using distributed flux.

$$\frac{T^{n+1} - T^*}{\nabla\tau} = q^* \tag{12.17}$$

In the above formulation, δ is the Dirac delta function which is employed to transfer the quantities between Eulerian and Lagrangian domains, effectively.

$$\delta = \frac{1}{h^2}\phi(\frac{x}{h})\phi(\frac{y}{h}) \tag{12.18}$$

where h is the Eulerian mesh width, x and y are Cartesian components of x. ϕ is the hat function, and we use 4-point delta function in our calculation Shin et al. (2008))

$$\phi(r) = \begin{cases} \frac{1}{8}(3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}) & \text{if} \quad 0 \le |r| \ge 1 \\ \frac{1}{8}(5 - 2|r| + \sqrt{-7 + 12|r| - 4r^2}) & \text{if} \quad 1 \le |r| \ge 2 \\ 0 & \text{otherwise} \end{cases} \tag{12.19}$$

For implementing the Neumann boundary condition, an additional layer of grid points is defined and placed one-grid spacing (h) distance outside the physical Lagrangian points (Zhang and Zheng 2007). The number of virtual points on this additional layer is same as the number of Lagrangian points on the body and

aligned in the surface normal direction. Therefore, for any pair of Lagrangian and virtual points,

$$Q = \frac{-\mathrm{d}T}{\mathrm{d}h} = \frac{T_l(x_k) - T_v(x_k)}{h} \tag{12.20}$$

T_v—temperature at the virtual point. First, the temperature at the virtual layer is interpolated using the method discussed in step 9, and then temperature at Lagrangian point is calculated by using the following expression

$$T_l^{n+1}(x_k) = -h\frac{\mathrm{d}T}{\mathrm{d}h} + T_v^n(x_k) \tag{12.21}$$

when a body is immersed in the fluid, aerodynamic forces act on the body. These forces are due to the distribution of pressure and shear stress over the body. Pressure acts normal to the body surfaces, whereas shear stress acts tangential to the body surface. In IBM, these aerodynamic forces are calculated by integrating the Lagrangian forces over the whole immersed boundary as given below:

$$F_L = -\int_0^L F_y ds, \quad F_D = -\int_0^L F_x ds \tag{12.22}$$

where F_L is the lift force acting on the body, F_D is the drag forces acting on the body, F_x is the horizontal component of Lagrangian force, F_y is the vertical component of Lagrangian force, ds is the distance between any two successive Lagrangian points, and L is the perimeter of the body.

12.2.2 Computational Domain and Grid Details

Figure 12.2 shows the computational domain used for simulation with the imposed boundary conditions. A uniform streamwise velocity profile and fully developed flow conditions are imposed at the inlet and outlet of the computational domain, respectively. Free-slip boundary conditions are imposed on the top and bottom walls. The elliptic cylinder is discretized with 315 Lagrangian marker points. The Eulerian domain is discretized with a non-uniform grid with 1271 and 845 grid points along x and y directions. A uniformly spaced grid is embedded around the elliptic cylinder for the effective use of forcing function and to resolve the shear layer. The size of the uniform grid-sized computational domain is $-1D \leq x \leq 1D$ and $-1D \leq y \leq 1D$ with the mesh size of $\Delta x = \Delta y = 0.003$. The size of the computational domain is chosen as $-8d_h \leq \mathrm{x} \leq 100d_h$ and $-8d_h \leq \mathrm{y} \leq 8d_h$. The detailed validation of the solver for several critical fluid and heat transfer parameters have been reported in Paul et al. (2013, 2014a, b, 2016), Paul (2013), and Pulletikurthi et al. (2014).

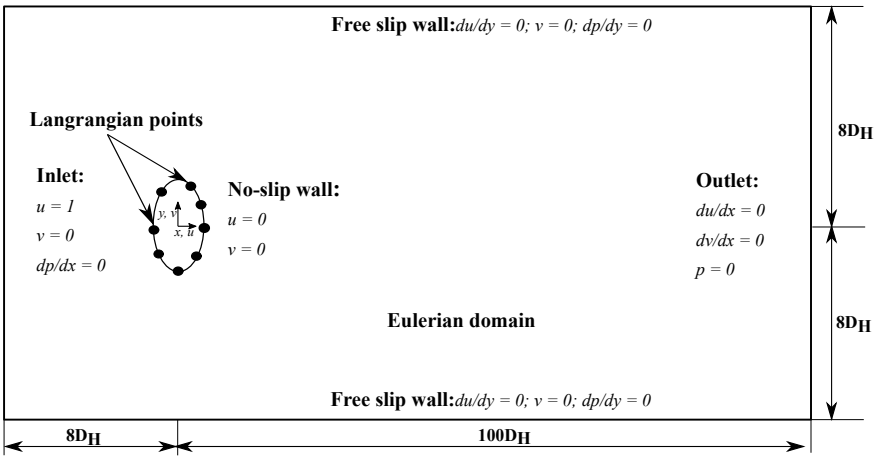


Fig. 12.2 Computational domain with boundary conditions

12.3 Results

This section presents the main results of this paper. First, we compute the critical Reynolds numbers at which flow separation and vortex shedding occur. Then, in the shedding regime, we analyze the different frequencies present in the wake. We show that the elliptic cylinder wakes exhibit low-frequency unsteadiness even in the near wake of the cylinder. Finally, we do thermal analysis of the thermal wake. We present an unusual mean temperature rise for the elliptic wake. This is accompanied with a secondary rise in the Nusselt number profile around the elliptic cylinder.

12.3.1 Computation of Laminar Separation Reynolds Number

As defined earlier, the critical laminar separation Reynolds number is at which the flow starts separating from the cylinder. There are various methods available in the literature to compute this quantity as discussed in Paul et al. (2014b). In this paper, we consider one of the methods called wake recirculation length method.

We know that when the flow separates, it forms a mean separation bubble at the lee surface of the cylinder. For instance, on the left image of Fig. 12.3, which is depicted for flow around a cylinder at Re = 6, there is no mean recirculation bubble is found as the flow does not separate at this Re. On the right image, however, we can clearly see the mean separation bubble for Re = 7 which means that the critical laminar separation bubble is in between 6 and 7.

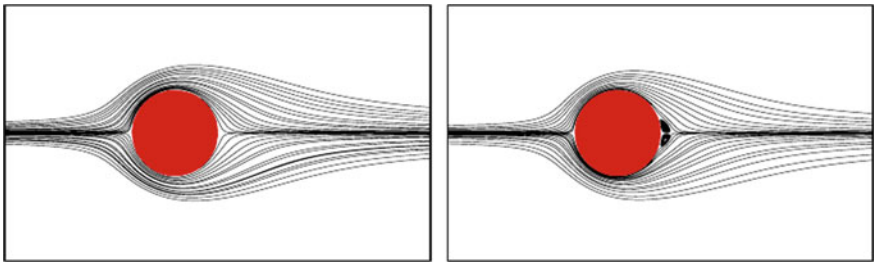


Fig. 12.3 Streamline contours of flow around a circular cylinder (AR = 1.0) at Re = 6 (left) and Re = 7 (right)

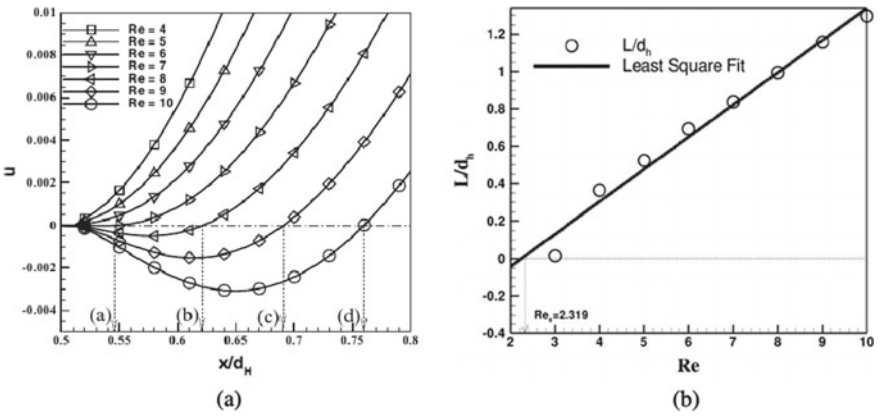


Fig. 12.4 **a** Measurement of wake lengths from u-velocity profiles for flow around circular cylinder, **b** Wake length method for circular cylinder (AR = 1.0)

In order to compute this critical Re accurately, we compute the mean recirculation length using the mean streamwise velocity profiles as shown in Fig. 12.4a. The mean recirculation length is the distance between the backward stagnation point and the streamwise distance at which the streamwise mean velocity becomes zero. This is illustrated in Fig. 12.4a. Once these values are computed for a range of Reynolds number, they are plotted against the Reynolds number as seen in Fig. 12.4b. The curve would usually be a straight line. Therefore, by plotting a linear square fit on the data we have, we can know the Reynolds number at which the mean recirculation length becomes larger than zero. For the circular cylinder, this critical Reynolds number is obtained as 6.27 which is in good agreement with the previous studies on circular cylinders (Paul et al. 2014b).

The critical laminar separation Reynolds numbers obtained for all axis ratios are tabulated in Table 12.1. As one can see, the flow separates at negligible Reynolds number when the axis ratio is small. Note that this is the first time such a set of values are reported in the literature. It seems that immersed boundary methods have

Table 12.1 Values of critical laminar separation and vortex shedding Reynolds numbers obtained for all simulations

Axis ratio	Critical laminar separation Re	Critical vortex shedding Re
0.1	0.90	23.48
0.4	2.31	28.50
0.6	3.11	31.34
0.8	5.08	36.26
1.0	6.27	48.34

the capacity to predict laminar separation accurately given that the mesh near the wall surface is properly resolved.

12.3.2 Estimation of Vortex Shedding Reynolds Number

It is well-known that the separation bubble that forms at the lee of the cylinder is unstable with respect to the Reynolds number. At a particular Reynolds number, Hopf bifurcation occurs and the separation bubble starts undulating and shed into the wake. This is widely known as von Karman vortex shedding in the literature. Having computed the Reynolds number at which the separation of flow occurs, this section focuses on computing the Reynolds number at which the vortex shedding starts in the wake.

There are different techniques available in the literature in order to compute this second critical Reynolds number. In this work, however, we will consider a method called saturation amplitude analysis. This analysis makes use of the lift

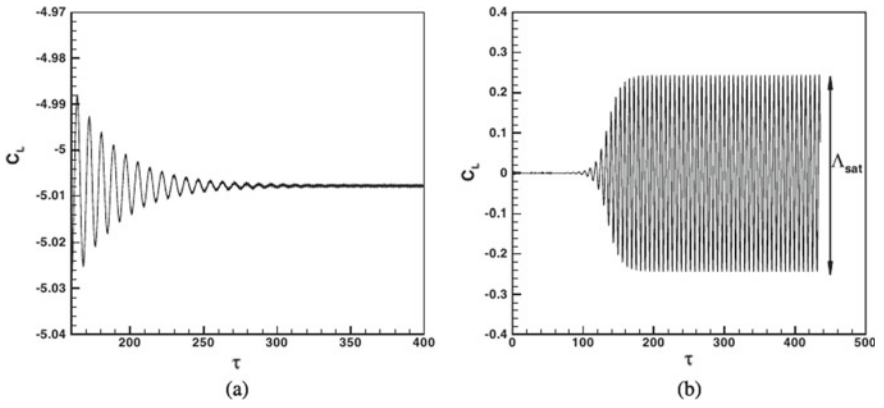


Fig. 12.5 **a** Signal with decaying amplitude. **b** Signal with saturated amplitude (Λ_{sat})

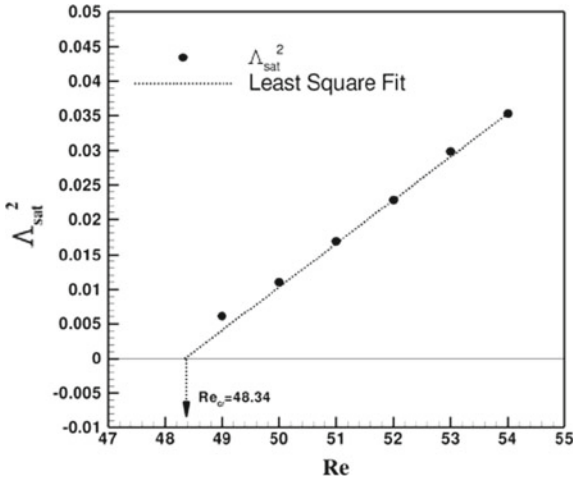


Fig. 12.6 Saturation amplitude analysis for circular cylinder

signal obtained from the cylinder surface. These signals show different behaviors according to whether there is a shedding or not. For non-shedding flows, the magnitude of the signal continually decreases in time and becomes zero at large time as shown in Fig. 12.5a. On the other hand, when there is vortex shedding, the magnitude of the lift curve increases from zero steadily to a constant value. The magnitude of lift signal at this saturation stage is called saturation magnitude (Λ_{sat}). A typical saturation magnitude is marked in Fig. 12.5b for the circular cylinder.

We simulate flow around cylinders for a fixed range of Reynolds number. More details on the way this range is fixed can be found in Paul et al. (2014b). Once the range is fixed, we compute Λ_{sat} for each case. Then, the square of these values (i.e., Λ_{sat}^2) is plotted against the Reynolds number. The curve Λ_{sat}^2—Re is a straight line, and therefore, we can perform the least square fit on the data as shown in Fig. 12.6. The least square curve passes the $y = 0$ line and the Re value associated at this juncture can be confidently called as the second critical Reynolds number. Our saturation amplitude analysis yields the critical vortex shedding Reynolds number for circular cylinder as 48.34 which is in good agreement with the literature.

We repeat the aforementioned procedure for all axis ratios considered in this study to compute the second critical Reynolds number. The obtained values are listed in Table 12.1. Again, the observations are similar to what was noted for the first critical Reynolds number. The vortex shedding starts at smaller Reynolds number for smaller axis ratio when the axis ration is kept perpendicular to the fluid flow. These results were never computed in the literature. We have demonstrated the capability of immersed boundary methods in predicting these values which are in general cumbersome to obtain accurately.

12.3.3 Low-Frequency Unsteadiness in the Near Wake

Having discussed the first and second critical Reynolds numbers of the flow, we turn our attention to the wake regimes where shedding is present. One of the important features of shedding wakes is the low-frequency unsteadiness. However, this phenomenon was noted only in the very far wake of a circular cylinder in the previous studies (Roshko 1954; Tritton 1959; Berger 1964; Inoue and Yamazaki 1999).

We first analyze the circular cylinder wakes. Since our computational domain is 100 diameters length in the streamwise direction from the back stagnation point, we could not note any low-frequency unsteadiness. Figure 12.7a, which represents flow around circular cylinder at Re = 180, shows the FFT of cross-stream velocity plotted along the different locations in the centerline of the wake. As can be seen from this figure, all the energy of flow is concentrated at one particular frequency (i.e., primary frequency of 0.13). This frequency is equivalent to the Strouhal number of the flow. The secondary low frequency was not detected here as it usually occurs for $x/D >$ 150 (Johnson et al. 2004).

Now, we analyze the data for flow around an elliptic cylinder of axis ratio 0.4 at Re = 100. As usual, we monitor the flow along the wake centerline at different locations. Figure 12.7b shows the result. In the near wake, we see something similar to the circular cylinder. That is, there is only one frequency which is equivalent to the shedding frequency. Things, however, change quite drastically even in the near wake of the elliptic cylinder. As can be seen in the figure, even around $x/D \approx 18$, we start seeing a secondary frequency, although at this location its m magnitude is very small. As we move further along the streamwise direction from this point, we start seeing this secondary low frequency becoming more prominent. In fact, in the

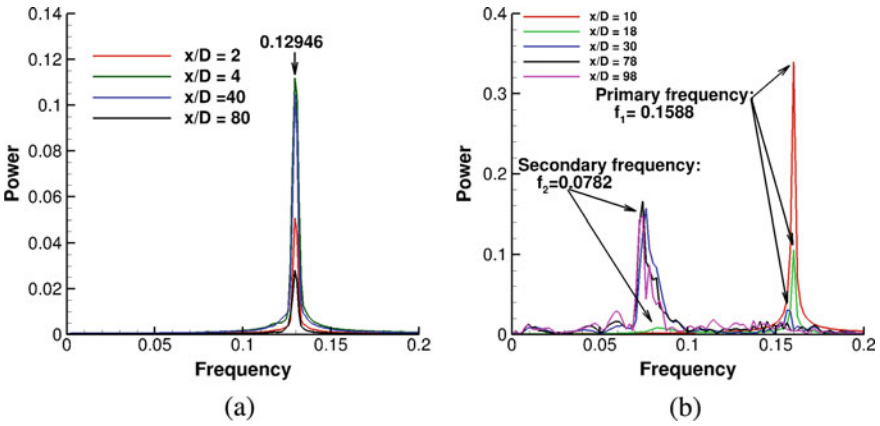


Fig. 12.7 FFT analysis of velocity signals taken at different downstream locations for **a** circular cylinder. **b** Elliptic cylinder

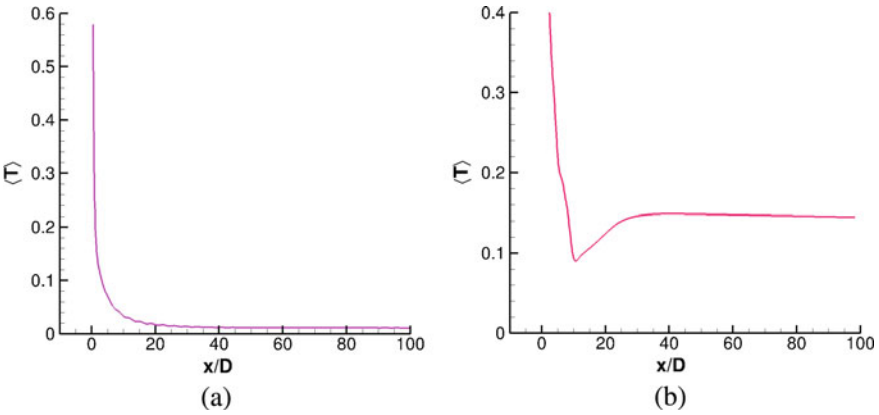


Fig. 12.8 Mean temperature profile along the wake centerline for **a** circular cylinder. **b** Elliptic cylinder

far wake (i.e., around $x/D > 70$), we have an absolute dominant low frequency. Thus, the elliptic cylinder wakes are unique in their characteristics that they exhibit low-frequency unsteadiness even in the near wake. Moreover, we have shown that the immersed boundary methods are capable of predicting such phenomenon.

12.3.4 Heat Transfer Characteristics of the Thermal Wake

Having studied the momentum wakes in the shedding regime, finally, we study the thermal wakes in the same shedding regime.

The target applications of this study include flow within a microprocessor or micromixing devices where cylindrical objects are placed to increase mixing. Thus, we are interested in the mean temperature evolution of the thermal wake.

The mean temperature profile downstream of the circular cylinder at Re = 180 is shown in Fig. 12.8a. As we heat the cylinder in this case, the temperature is higher near the wall. However, as the vortices transport heat in the wake, the temperature decreases drastically in the near wake and it reaches a near constant value further downstream. This is due to the diffusion process associated with the flow that transports heat.

On the other hand, the elliptic cylinder thermal wakes exhibit completely different behavior as reported in Fig. 12.8b. This Figure is plotted for axis ratio 0.4 and Re = 100. Here also, the temperature is maximum near the cylinder surface due to heating, and the magnitude of temperature falls sharply in the near wake. However, we note a peculiar trend in the mean temperature profile here. The temperature does not reach a saturation stage after drastically getting reduced in the near wake. Rather, it starts increasing in the wake and the magnitude of increase is roughly 20%. Such

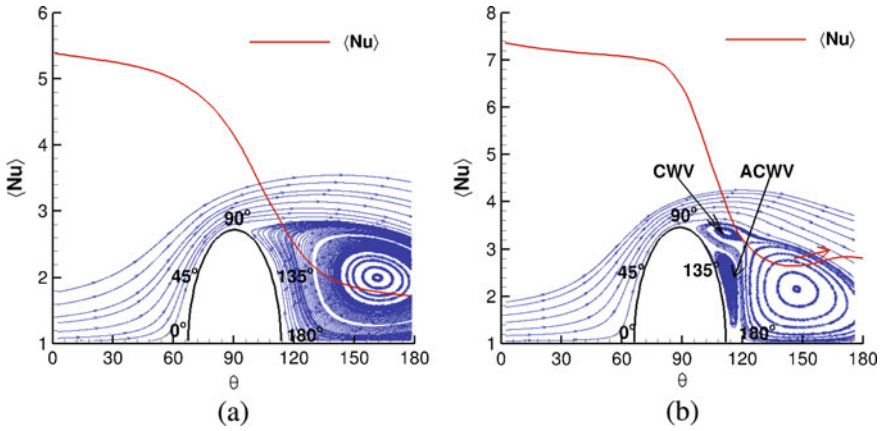


Fig. 12.9 Nusselt number distribution around **a** circular cylinder. **b** Elliptic cylinder. Mean streamlines are plotted in the background

a profile will significantly affect the mixing characteristics if an elliptic cylinder is employed in microfluidic devices. Paul et al. (2016) has shown that the reason for this increase in mean temperature within the wake is the different shedding type that occurs for elliptic cylinder of axis ration 0.4 at Re = 100 where the regular von Karman vortex street breaks down and forms into two parallel chain of same magnitude vortices. This process brings two vortices closer and thus causing an increase in mean temperature.

Finally, we study the Nusselt number distribution around the cylinder. The other target application of cylinders is the heat exchanger design where one is interested in exporting hot gas with less drag. Many studies have shown that the elliptic cylinders have smaller drag coefficient. Therefore, it is tempting to use them as heat exchanger tubes. For this reason, we must know the Nusselt number distribution around the cylinder tube.

Figure 12.9a shows the Nusselt number distribution for circular cylinder at Re = 100. Here, the maximum Nu is at the front stagnation point. This is because the cold flow touches first the cylinder at the front stagnation point, and therefore, it takes away much heat from the cylinder and thus resulting in higher heat transfer coefficient. This can also seen in the mean streamlines plot shown in Fig. 12.9a. Following this, now, a warmer fluid with temperature higher than the free stream temperature passes over the cylinder. Because of this, the heat transfer transfer rate decreases continuously from the front stagnation point. This decrease is aggravated further around 100° as at this juncture the flow separates to form a recirculation region which brings even more warmer fluid toward the cylinder. Therefore, for circular cylinder, there is only one maximum Nu point at the front stagnation region which is followed by a continual decrease.

The Nusselt number profile for heat transfer around an elliptic cylinder of axis ration 0.4 at Re = 100 is shown in Fig. 12.9b. We see an important difference here

compared with the Nu profile for the circular cylinder. Here, for the elliptic cylinder, we also note a secondary peak as noted as red colored arrow in Fig. 12.9b. This secondary peak is due to the nature of separation that elliptic cylinder wakes exhibit. Here, there are two counter-rotating vortices that are formed in the lee of the cylinder as seen in the mean streamlines plot of Fig. 12.9b. The implication of this is that, although the first vortex that lies between 95° and 120° transports more hot fluid toward the surface of the cylinder, the nearby vortex that lies between 125° and 180° is counter-rotating to the other vortex and thus it brings more cold fluids to the cylinder. As a result, the heat transfer rate increasing around 125° leading to a secondary maximum. Again, we have established that the immersed boundary methods can be effectively used to model thermal wakes and heat transfer from cylinders of different shapes.

12.4 Conclusion

The main objective of this work is to develop immersed boundary methods to model fluid flow and heat transfer around differently shaped cylinders. We considered elliptic cylinders of five different axis ratio whose major axes are kept perpendicular to the incoming flow direction.

We developed immersed boundary method based Navier–Stokes solver utilizing the direct forcing method. We have also shown the extension of the fluid solver to simulate thermal wakes behind the cylinder. The cylinder surface is modeled as a set of Lagrangian marker points. The information about the surface is then spread to the underlying Eulerian mesh through the use of Dirac delta function. We implemented both the constant temperature and constant heat flux boundary conditions. The solver has been validated extensively for various flow and thermal quantities.

We then applied the solver we developed to study fundamental physics of momentum and thermal wakes. We accurately computed the first and second critical Reynolds numbers at which the flow starts separating from the cylinder and at which the flow starts shedding, respectively. Then, we turned our attention to the flow regime where shedding was present. We showed that elliptic cylinder wakes are unique such that they exhibit low-frequency unsteadiness even in the near wake. We also studied the heat transfer characteristics. In particular, we demonstrated an unusual evolution of mean temperature in the wake of an elliptic cylinder. Finally, the elliptic cylinders are shown to have a secondary peak on the Nusselt number profiles due to their novel flow separation characteristics. In all these results, immersed boundary method was proven to be adequate to study flow and thermal physics.

Acknowledgements We thank the Aerodynamics Research and Development Board (ARDB) of India, and the Ministry of Human Resources India for funding this work.

References

Aref H, Siggia ED (1981) Evolution and breakdown of a vortex street in two dimensions. J Fluid Mech 109:435–463

Berger E (1964) The determination of the hydrodynamic parameters of a karman vortex street from hot wire measurements at low reynolds number. Z Flugwiss 12:41

Brown DL, Cortez R, Minion ML (2001) Accurate projection methods for the incompressible navier-stokes equations. J Comput Phys 168(2):464–499

Cimbala JM, Nagib HM, Roshko A (1988) Large structure in the far wakes of two-dimensional bluff bodies. J Fluid Mech 190:265–298

Inoue O, Yamazaki T (1999) Secondary vortex streets in two-dimensional cylinder wakes. Fluid Dyn Res 25(1):1

Johnson SA, Thompson MC, Hourigan K (2001, December) Flow past elliptical cylinders at low Reynolds numbers. In: Proceedings of 14th Australasian fluid mechanics conference, Adelaide University, South Australia, pp 9–14

Johnson SA, Thompson MC, Hourigan K (2004) Predicted low frequency structures in the wake of elliptical cylinders. Eur J Mech B/Fluids 23(1):229–239

Kim J, Kim D, Choi H (2001) An immersed-boundary finite-volume method for simulations of flow in complex geometries. J Comput Phys 171(1):132–150

Kundu P, Cohen I (2008) Fluid mechanics, San Diego: Elsevier. 4th edn

Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261

Najjar F, Balachandar S (1998) Low-frequency unsteadiness in the wake of a normal flat plate. J Fluid Mech 370:101–147

Ota T, Aiba S, TSURUTA T, Kaga M, (1983) Forced convection heat transfer from on elliptic cylinder of axis ratio 1: 2. Bull JSME 26(212):262–267

Pacheco J, Pacheco-Vega A, Rodić T, Peck R (2005) Numerical simulations of heat transfer and fluid flow problems using an immersed-boundary finite-volume method on nonstaggered grids. Numer Heat Transf Part B Fundam 48(1):1–24

Paul I (2013) Effect of axis ratio and incidence on fluid flow and heat transfer around elliptic cylinders. Master's thesis, Indian Institute of Technology Madras, Chennai

Paul I, Prakash K, Vengadesan S (2013) Forced convective heat transfer from unconfined isothermal and isoflux elliptic cylinders. Numer Heat Trans A Appl 64(8):648–675

Paul I, Prakash K,Vengadesan S (2014a) Numerical analysis of laminar fluid flow characteristics past an elliptic cylinder: A parametric study. Int J Numer Method H 24(7):1570–1594

Paul I, Prakash KA, Vengadesan S (2014b) Onset of laminar separation and vortex shedding in flow past unconfined elliptic cylinders. Phys Fluids 26(2):023601

Paul I, Prakash KA, Vengadesan S, Pulletikurthi V (2016) Analysis and characterisation of momentum and thermal wakes of elliptic cylinders. J Fluid Mech 807:303–323

Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10(2):252–271

Pulletikurthi V, Paul I, Prakash KA, Prasad B (2014) On the development of low frequency structures in near and far laminar wakes. Phys Fluids 26(2):023601

Roshko A (1954) On the development of turbulent wakes from vortex streets, NACA report, 1191

Saha AK (2007) Far-wake characteristics of two-dimensional flow past a normal flat plate. Phys Fluids 19(12):128110

Shin SJ, Huang WX, Sung HJ (2008) Assessment of regularized delta functions and feedback forcing schemes for an immersed boundary method. Int J Numer Meth Fluids 58(3):263–286

Su SW, Lai MC, Lin CA (2007) An immersed boundary technique for simulating complex flows with rigid boundary. Comput Fluids 36(2):313–324

Taneda S (1959) Downstream development of the wakes behind cylinders. J Phys Soc Jpn 14(6):843–848

Thompson MC, Radi A, Rao A, Sheridan J, Hourigan K (2014) Low-reynolds-number wakes of elliptical cylinders: from the circular cylinder to the normal flat plate. J Fluid Mech 751:570–600

Tritton DJ (1959) Experiments on the flow past a circular cylinder at low reynolds numbers. J Fluid Mech 6(4):547–567

Van Der Vorst HA (2002) Efficient and reliable iterative methods for linear systems. J Comput Appl Math 1:251–265

Zhang N, Zheng ZC (2007) An improved direct-forcing immersed-boundary method for finite difference applications. J Comput Phys 221(1):250–268

Chapter 13
Investigation of the Unsteady Aerodynamics of Insect Flight: The Use of Immersed Boundary Method

Srinidhi Nagarada Gadde, Y. Sudhakar, and S. Vengadesan

13.1 Introduction

The current chapter summarizes the insect flight research carried out in the group of Prof. S. Vengadesan at IIT Madras, with immersed boundary methods (IBM) as the research tool. While most of the insects employ a symmetric wing motion along a horizontal stroke plane (e.g., fruit-flies, bees, and beetles), a few insects (e.g., dragonflies and hover-flies) translate their wings asymmetrically along a more inclined stroke plane. Our work focuses on the unsteady aerodynamics involved in the inclined stroke plane motions, and we address the following aspects of such a flight with numerical simulations of idealized two-dimensional kinematics of insect wings:

- Mechanism of vertical force generation
- Influence of multiple wings and their relative kinematics on force generation
- The effect of ground on vortex dynamics and force generation.

S. N. Gadde
Physics of Fluids Group, University of Twente, 7500 AE Enschede, The Netherlands
e-mail: s.nagaradagadde@utwente.nl

Y. Sudhakar
School of Mechanical Sciences, Indian Institute of Technology Goa,
Ponda, Goa 403401, India
e-mail: sudhakar@iitgoa.ac.in

S. Vengadesan (✉)
Department of Applied Mechanics, IIT Madras, Chennai 600036, India
e-mail: vengades@iitm.ac.in

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods
in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_13

13.1.1 Brief Review of Insect Aerodynamics

The study of insect flight is fascinating in its own right due to the underlying unsteady aerodynamics. Moreover, the knowledge gained from such studies will greatly benefit the designing of micro-aerial vehicles (MAVs) which has potential applications in military reconnaissance, weather monitoring, and information gathering. Here, we provide a very brief review of the unsteady aerodynamics involved in the insect flight. Extensive details can be found in the comprehensive reviews available in the literature (Sane 2003; Platzer et al. 2008; Shyy et al. 2010).

By using high-speed photography, Ellington (1984a) found that a typical flapping flight of an insect consists of two translational (upstroke and downstroke) and two rotational (pronation and supination) motions. To explain the aerodynamics involved in the aforementioned complex kinematics, classical potential flow aerodynamics and quasi-steady-state theories have been proposed. Quasi-steady-state theories assume that instantaneous forces on a flapping wing are equivalent to those for steady motion at the same instantaneous velocity and angle of attack. By comparing the theoretical results with experimental observations, Ellington (1984b) proved that conventional quasi-steady-state theories are insufficient to explain the enhanced lift force observed in the flight of hovering insects. The failure of such theories strongly suggests that unsteady aerodynamic mechanisms play a key role in the flapping flight. Previous experimental and numerical studies have uncovered three important unsteady mechanisms in flapping flight:

- Delayed stall: Stable attached leading-edge vortices (LEVs) are formed over the insect wings even when their angle of attack (AoA) is as high as 40° (Ellington et al. 1996). These attached LEVs greatly enhance the lift force on flapping wings.
- Rotational circulation: The rotational motion of insect wings (pronation and supination) induce additional circulation around the wing, leading to large lift force (Dickinson et al. 1999).
- Wake capture: Insects interact favorably with the wake vortices that are shed in the earlier cycles of flapping and this leads to additional aerodynamic forces (Dickinson et al. 1999; Birch et al. 2004).

The aforementioned unsteady aerodynamic mechanisms are responsible for the observed high performance of flapping insect wings at low Reynolds numbers.

13.1.2 Tandem Wing Aerodynamics

In contrast to the aerodynamics of single-winged insects, flow structures involved in tandem winged fliers are more complex. Dragonflies, nature's most ubiquitous, agile, and highly maneuverable fliers have tandem wings (a forewing and a hindwing). They generally flap their wings in a stroke plane that is 60° relative to the horizontal (Norberg 1975). High maneuverability of dragonflies is due to the presence of fore and hindwings that move independently of each other (Alexander 1984).

Multiple wings cause wing–vortex and wing–wing interactions resulting in complex lift and drag variations. Lan and Sun (2001) studied the elliptical airfoils flapping in tandem at phase differences $\psi = 0°, 90°$, and $180°$ by solving 2D incompressible Navier–Stokes (N-S) equations on moving over-set grids. They reported that in-phase stroking ($\psi = 0°$) produces the maximum lift and $\psi = 90°$ phase difference produces the minimum lift. Furthermore, computational fluid dynamics (CFD) simulations of Wang and Sun (2005) show that the forewing–hindwing interaction results in reduced lift forces. In addition, the simulations of Wang and Russell (2007) show that a dragonfly uses out-of-phase flapping to minimize the power consumption during hovering; and in-phase flapping during take-offs which require maximum power. With the experiments on robotic wings, Usherwood et al. (2008) showed that dragonflies employ wing phasing to remove swirl and improve the efficiency. In general, the vortex wake contains swirl which reduces the aerodynamic efficiency of the wings and the forewing–hindwing interaction can be either beneficial or detrimental to the performance of the wings.

13.1.3 Ground Effect

Apart from the wing–wing and wing–vortex interactions, the presence of a wall can influence the vortical structures and the vortex-induced forces. Gao and Lu (2008) studied a model wing flapping in a horizontal stroke plane near the ground and reported three force regimes, viz. force enhancement, force reduction, and force recovery regimes with the conclusion that both shed and rebound vortices decide the variation of the lift and drag forces. Liu et al. (2009) extended the study for clap and fling kinematics, and De Rosis (2015) for symmetric wings flapping in tandem. By high-resolution digital particle image velocimetry (DPIV), van Truong et al. (2013) studied the vortical structures surrounding a beetle's wing during take-off and showed that the ground enhances the size and shape of the LEV. Recently, Kolomenskiy et al. (2016) with 3D CFD simulations using an immersed interface method studied the take-off of an insect. Interestingly, whether the ground effect increases or decreases the aerodynamic forces is dictated by the kinematics of flapping motion and further investigations are necessary to improve our understanding of the ground–vortex interactions.

13.1.4 Suitability of IBM to Study Insect Flight

The current state-of-the-art insect flight research involves two steps: (1) accurate measurement of insect wing kinematics and flow field using high-speed imaging (Ennos 1989; Altshuler et al. 2005; Fry et al. 2005) and (2) replication of these kinematics either in mechanical fliers or in a CFD simulation. While experiments give a reliable estimate of forces, they provide only a limited information of the flow and obtaining

the complete flow field information over the rapidly oscillating insect wings in a fully non-intrusive manner is extremely challenging.

Numerical studies of insect flight necessitate the simulation of flow over rapidly oscillating wings. Conventional CFD methods require the generation of high-quality body-fitted structured or unstructured grid over the immersed boundaries, which is a daunting task in itself while dealing with flow past complex geometries. A poor quality grid can negatively impact the accuracy, stability, and convergence properties of the solver. Often, over complex geometries, the task of grid generation is carried out by dividing the computational domain into various sub-domains and generating the grid separately in these domains. Besides increasing the complexity of the solution algorithm, the deterioration in grid smoothness at the interface of the sub-domains can affect the stability of the solver. When the finite difference method is employed on a structured grid, the transformation of governing equations from the physical domain into the computational domain increases the per-grid-point operation count (Mittal and Iaccarino 2005).

While simulating the moving boundary problems with the help of a body-fitted grid, one encounters two difficulties:

- Transient re-meshing strategies are compulsory to accommodate the change in the shape or orientation of the body in fluid flow.
- A stable algorithm is necessary to project the old solution onto the new grid.

In addition to increasing the computational cost, these steps restrict the maximum time-step size that can be used for stable computations. IBM (Mittal and Iaccarino 2005) can be used to circumvent the aforementioned problems. IBM was first proposed by Peskin (1972) to study the flow around heart valves; numerous modifications have been proposed to the method since then (Goldstein et al. 1993; Fadlun et al. 2000; Kim and Choi 2006). In the past, IBM has been successfully used to simulate flows with complex moving boundaries such as flapping wings (Gilmanov and Sotiropoulos 2005; De Rosis 2014; Sudhakar and Vengadesan 2010a; Srinidhi and Vengadesan 2017a, b). IBM is particularly suited for flapping wing simulations due to the ease with which the kinematics can be imposed on the wings, high accuracy, and the computational advantage it provides. There are many variants of continuous forcing IBM available in the literature, and we make use of the immersed boundary projection method (IBPM) proposed by Taira and Colonius (2007).

In Sect. 13.2, the governing equations, the methodology of IBM solver, and a brief note on the multi-processor implementation of the IBM solver are detailed. In the subsequent sections, the IBM will be used to study the mechanism of vertical force generation in inclined stroke plane kinematics of insect flight, the effect of wing interference in the case of tandem wings, and the effect of ground on vortex dynamics of the flapping flight.

13.2 Governing Equations and the Numerical Method

We use the IBPM proposed by Taira and Colonius (2007) to develop a parallelized IBM solver. IBPM considers boundary force as a Lagrange multiplier to satisfy the no-slip condition; this is similar to the pressure acting as a Lagrange multiplier to satisfy the divergence-free constraint. Poisson equation for the pressure is modified to incorporate both divergence-free constraint as well as the no-slip constraint on the body. In IBM, the N-S equations are solved on a non-body conforming grid called Eulerian grid, $\mathscr{D}$, and a set of discrete Lagrangian points, ξ_k, represent the surface of a body, $\mathscr{B}$. Similar to most IBM, the incompressible flow is initially solved on an Eulerian grid, and the intermediate velocities are interpolated onto the Lagrangian points using an interpolation operator. The interpolated velocities are used to calculate the forces at the Lagrangian points and the forces are redistributed (regularized) to the nearby Eulerian grid points. In IBM, the Lagrangian points do not necessarily coincide with the underlying Eulerian grid. Hence, to interpolate the quantities to the Lagrangian points, discrete delta functions are used to exchange information between the Eulerian grid and the Lagrangian points.

The governing equations used are:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \Delta \mathbf{u} + \int_S \mathbf{f}(\boldsymbol{\xi}(s,t))\delta(\boldsymbol{\xi} - \mathbf{x})\mathrm{d}s, \tag{13.1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{13.2}$$

$$\mathbf{u}(\boldsymbol{\xi}(s,t)) = \int_S \mathbf{u}(\mathbf{x})\delta(\mathbf{x} - \boldsymbol{\xi})d\mathbf{x} = \mathbf{u}_B(\boldsymbol{\xi}(s,t)), \tag{13.3}$$

where $\mathbf{x} \in \mathscr{D}$, $\boldsymbol{\xi}(s,t) \in \mathscr{B}$, $\mathbf{u}$ represent the velocity vector, p is the pressure, ν is the kinematic viscosity, ∇ is the gradient operator, Δ is the Laplacian operator, and $\mathbf{f}$ represents the immersed boundary force. The boundary $\mathscr{B}$ is parameterized by s and moves at the velocity, $\mathbf{u}_B(\boldsymbol{\xi}(s,t))$. The governing equations are solved on staggered grids with pressure at the center of the cell and velocities located on the cell faces. The viscous terms are discretized with implicit Crank-Nicholson scheme and the second-order Adams-Bashforth scheme is used to discretize the nonlinear advective terms. The schemes yield a formal second-order accuracy in space and first-order accuracy in time.

The discretized governing equations can be written as:

$$\begin{pmatrix} A & G & -H \\ D & 0 & 0 \\ E & 0 & 0 \end{pmatrix} \begin{pmatrix} u^{n+1} \\ \phi \\ f \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \\ u_B^{n+1} \end{pmatrix} + \begin{pmatrix} bc_1 \\ -bc_2 \\ 0 \end{pmatrix}, \tag{13.4}$$

where Hf corresponds to the last term in Eq.(13.1) which is the regularization operation. ϕ represents the pressure. The interpolation operator E is used to enforce the no-slip condition [Eq. (13.2)]; where, $Eu^{n+1} = u_B^{n+1}$. A, D, and G represent the implicit operator for velocity, discrete divergence, and gradient. r^n, bc_1, and bc_2 are

the explicit terms in the momentum equation, inhomogeneous terms resulting from the boundary condition of Laplacian operator and from the divergence operator, respectively. H and E represent the regularization and interpolation operators used to exchange information between the Eulerian and Lagrangian grid points. The operators are constructed using discrete delta function proposed by Roma et al. (1999). The present delta function is supported over three cells and has the form:

$$d(r) = \begin{cases} \frac{1}{6\Delta r}\left[5 - 3\frac{|r|}{\Delta r} - \sqrt{-3\left(1 - \frac{|r|}{\Delta r}\right)^2 + 1}\right] & \text{for} \quad 0.5\Delta r \le |r| \le 1.5\Delta r, \\ \frac{1}{3\Delta r}\left[1 + \sqrt{-3\left(\frac{r}{\Delta r}\right)^2 + 1}\right] & \text{for} \quad |r| \le 0.5\Delta r, \\ 0 & \text{otherwise}, \end{cases} \tag{13.5}$$

where Δr is the cell width. The delta function can only be used in uniform grids, so the extent of the domain in which the bodies move is discretized uniformly, stretched grids are used in the rest of the domain (Fig. 13.1).

Convolution of Eulerian background velocities with the delta function gives the velocities at the Lagrangian points,

$$\mathbf{u}(\boldsymbol{\xi}) = \int_x \mathbf{u}(\mathbf{x})\delta(\mathbf{x} - \boldsymbol{\xi})\mathbf{dx}, \tag{13.6}$$

the convolution yields:

$$u_k = \Delta x \Delta y \sum_i u_i d(x_i - \xi_k) d(y_i - \eta_k), \tag{13.7}$$

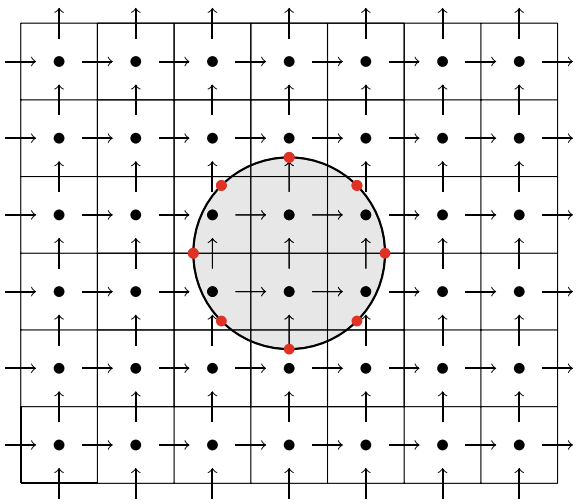


Fig. 13.1 Body $\mathscr{B}$, is represented by the shaded object immersed in a 2D domain $\mathscr{D}$ discretized by a staggered grid. Horizontal and vertical arrows ($\rightarrow$, $\uparrow$) denote the u and v velocity nodes, respectively. Pressure is located at the center of each cell depicted by circles ($\bullet$). Lagrangian points, $\boldsymbol{\xi}_k = (\xi_k, \eta_k)$, are shown by red circles

If $\gamma = \Delta x \Delta y$, Eq. (13.7) can be simplified as,

$$E_{k,i} = \gamma d(x_i - \xi_k) d(y_i - \eta_k), \tag{13.8}$$

The regularization operator H is also obtained by the convolution of Lagrangian values with the delta function, and it is equal to $-E^T$. We can formulate G and D such that $D = -G^T$.

$$\begin{pmatrix} A & G & E^T \\ G^T & 0 & 0 \\ E & 0 & 0 \end{pmatrix} \begin{pmatrix} u^{n+1} \\ \phi \\ f \end{pmatrix} = \begin{pmatrix} r^n \\ 0 \\ u_B^{n+1} \end{pmatrix} + \begin{pmatrix} bc_1 \\ -bc_2 \\ 0 \end{pmatrix}, \tag{13.9}$$

Considering both ϕ and f as Lagrange multipliers, we get:

$$Q \equiv [G, E^T], \quad \lambda \equiv \begin{pmatrix} \phi \\ f \end{pmatrix}, \quad r_1 \equiv r^n + bc_1, \quad r_2 \equiv \begin{pmatrix} -bc_2 \\ u_B^{n+1} \end{pmatrix}. \tag{13.10}$$

Using Eqs. (13.10), (13.9) can be simplified as below:

$$\begin{pmatrix} A & Q \\ Q^T & 0 \end{pmatrix} \begin{pmatrix} q^{n+1} \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}, \tag{13.11}$$

where scaling factors have been used to convert velocities, u^{n+1}, to fluxes, q^{n+1}, at cell faces. Thus, the steps in immersed boundary projection method are:

$$Aq^* = r_1 \qquad \text{(Solve for intermediate velocity)}, \tag{13.12}$$

$$Q^T B^N Q\lambda = Q^T q^* - r_2 \qquad \text{(Solve the modified Poisson equation)}, \tag{13.13}$$

$$q^{n+1} = q^* - B^N Q\lambda \qquad \text{(Projection step)}, \tag{13.14}$$

where B^N is an approximation of A^{-1}. Additional details about the algorithm and its implementation can be found in Taira and Colonius (2007). Major difference and advantage of IBPM are that it calculates the body forces and pressure values in a single step by solving Eq. (13.13).

Our code is based on the open-source code PetIBM developed by Krishnan (2015). We modified the code and added the capability to perform simulations with moving bodies. The C++ based code is parallelized using the open-source parallel programming library PETSc (Balay et al. 2019). Basic parallelization strategies and the necessary details of the implementation are given in Krishnan (2015). Here, we focus on the implementation details of the moving boundary simulations. In vector r_2 of Eq. (13.10), $u_B^{n+1} = 0$ for the flow over stationary bodies, and $u_B^{n+1} \neq 0$ for the flow over moving boundaries. We created a parallel array to distribute Lagrangian points to different processors. In simulations involving moving boundaries, the elements in the modified Poisson matrix $Q^T B^N Q$ changes due to the change in the position of Lagrangian points. This necessitates the modification of the matrix at the end of every time step. The matrix $Q^T B^N Q$ and the parallel distribution vector corresponding to the immersed boundary are destroyed and recreated after every time step. As the

creation of a parallelized matrix in PETSc is time consuming; the aforementioned step is the major bottleneck in the present implementation. Modification of Poisson matrix in IBPM increases the condition number of the matrix system. To solve the ill-conditioned system, we use Krylov sub-space iterative solvers with multigrid preconditioner.

13.3 The Mechanism of Force Generation in Flapping Flight

Of the three unsteady aerodynamic mechanisms presented in Sect. 13.1, delayed stall, i.e., enhanced force generation due to the attached LEVs is the most significant lift generation mechanism for insect flight. For a fruit-fly, which uses horizontal wing motion, the delayed stall generates enough force to support more than 85% of its total weight (Wu and Sun 2004). However, the functional significance of delayed stall in hovering insects which oscillate their wings along an inclined stroke plane is still less evident. While horizontal stroke plane motions rely on lift, drag on the wings in inclined stroke plane motions makes a significant contribution to support the weight (Wang 2004). Given the different strategies of weight support inherent in these horizontal plane and inclined plane wing motions, it is natural to expect that aerodynamic force generation mechanism in lift-dependent horizontal stroke plane wing motions is different than in drag-dependent inclined stroke plane wing motions.

In this work, we consider the following idealized wing kinematics for the flapping motion of insect wings which is schematically shown in Fig. 13.5.

Translational velocity,

$$\mathrm{v}(\tau) = -\sin\left(2\frac{c}{A_0}\tau\right), \tag{13.15}$$

Angular velocity confined to stroke reversal,

$$\omega(\tau) = \bar{\omega}\left[1 - \cos\left(\frac{2\pi(\tau - \tau_r)}{\Delta\tau_r}\right)\right]; \quad \tau_r \le \tau \le (\tau + \Delta\tau_r), \tag{13.16}$$

where τ is the non-dimensional time, c is the chord length of the wing, A_0 is the stroke amplitude, τ_r is the time at which the rotation starts, $\Delta\tau_r$ is the time required to perform the rotation, $\bar{\omega} = \frac{\Delta\theta}{\Delta\tau}$ is the average rotational speed, and $\Delta\theta$ is the change in AoA achieved in wing rotation.

Non-dimensional period of wing beat cycle (τ_c) can be found by the following relation,

$$2\frac{c}{A_0}\tau_c = 2\pi. \tag{13.17}$$

A flat plate of 2% thickness to chord ratio is used to model the wing cross section. The plate is discretized with 102 Lagrangian points ($\Delta s = 0.02$). The size of the rectangular computational domain chosen is ($-30c \le x \le 30c$, $-30c \le y \le 30c$).

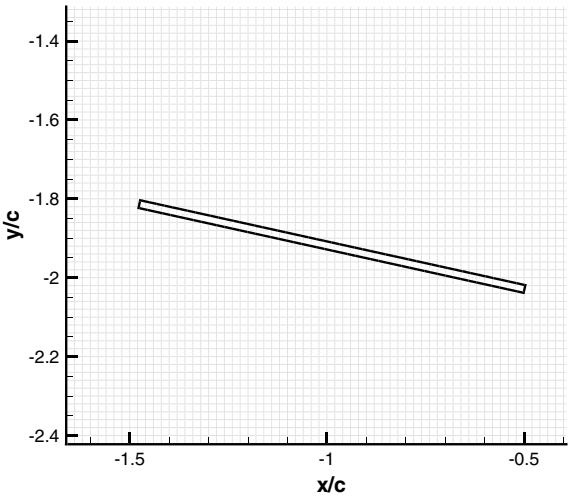
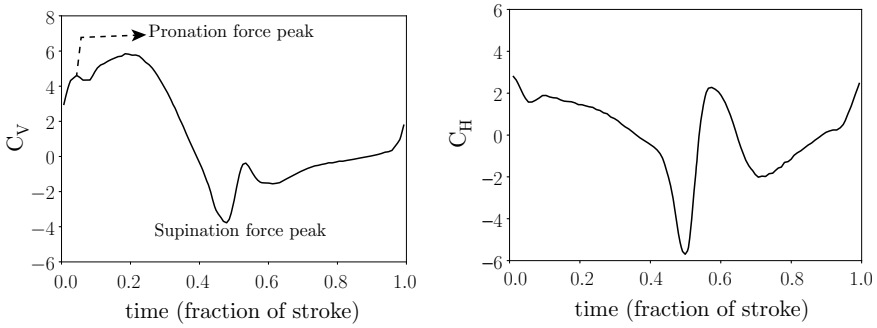


Fig. 13.2 Close up view of non-body conformal grid around the wing

A small area within the computational domain is discretized with uniform grid of $\Delta x = \Delta y = 0.02$. The size of this area is chosen in such a way that the wing is immersed within the uniform grid region throughout the stroke. The final size of the Eulerian grid in x- and y-direction for 2.5 chord lengths travel is 397 and 438, respectively. A picture of the wing immersed in the non-body conformal Cartesian grid is shown in Fig. 13.2. Every flapping cycle is discretized with 2000 time steps ($\Delta\tau_r = \tau_c/2000$). All the results presented in the subsequent sections are for the wing during its tenth cycle of flapping, by which time the forces and the flow reach a periodic state. The instantaneous forces are non-dimensionalized with $0.5\rho \mathrm{v}_{\mathrm{rms}}^2 c$. It has been confirmed that the results presented here are grid- as well as time-step independent.

We simulate the flapping wing with the following typical kinematic parameters: Reynolds number, $\mathrm{Re}(= \mathrm{v}_{\mathrm{max}} c/\nu) = 150$ where ν is the kinematic viscosity, stroke amplitude, $A_0 = 2.5c$, rotational period is 20% of the period of wing beat cycle ($\Delta\tau_r = 0.2\tau_c$), and stroke plane angle, $\beta = 62.8°$. The AoA during downstroke and upstroke are 50.6° and 15°, respectively. All these details are for the dragonfly hovering, similar to the simulations of Wang (2004), except that in our study a flat plate is used to model the cross section of the wing and the wing rotation in our study is confined to stroke reversal.

The time history of vertical force coefficient $\mathrm{C_V} = \frac{\mathrm{F_V}}{\frac{1}{2}\rho \mathrm{v_{rms}}^2 c}$ and horizontal force coefficient $\mathrm{C_H} = \frac{\mathrm{F_H}}{\frac{1}{2}\rho \mathrm{v_{rms}}^2 c}$ for one complete stroke is shown in Fig. 13.3; here, $\mathrm{F_V}$ and $\mathrm{F_V}$ are the vertical and horizontal forces on the wing, respectively. The stroke averaged horizontal force coefficient, $\overline{\mathrm{C_H}}$ is almost zero, which confirms that the simulation is for hovering motion. Since $\overline{\mathrm{C_H}}$ is almost zero in other simulations also, only the time history of $\mathrm{C_V}$ is presented in the subsequent sections. As has been



(a) Vertical force coefficient (b) Horizontal force coefficient

Fig. 13.3 Time history of force coefficients at Re = 150. The time between 0–0.5 is downstroke and 0.5–1 is upstroke

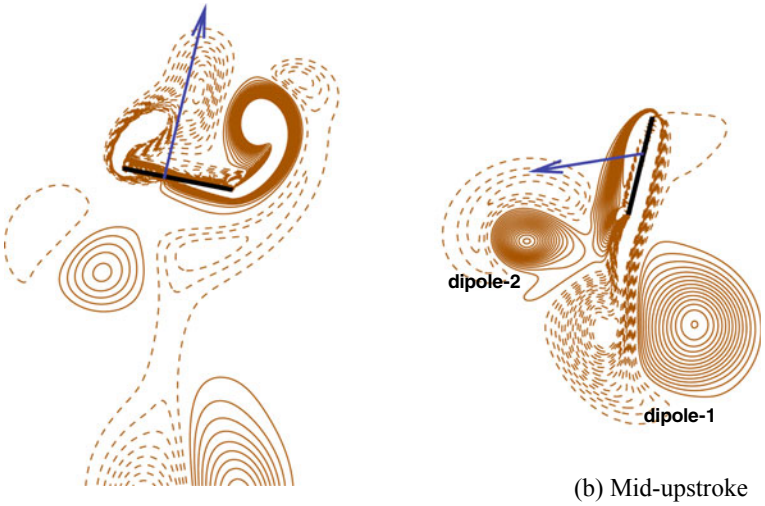
explained, downstroke induces positive C_V, and in upstroke negative C_V is generated over the flapping wing. The figure also shows the force peaks during the wing rotation (pronation and supination).

In the downstroke, two attached vortices are formed on either side of the wing as shown in Fig. 13.4a. The flow field is analogous to the flow past a bluff body in a laminar steady-state regime. The pressure drag formed over the wing is very high, so that the vertical force production is also high. The net aerodynamic force produced during the downstroke is almost perpendicular to the wing (Fig. 13.4a), implying the dominance of pressure forces over viscous forces. During upstroke, attached shear layers are formed over the wings without vorticity roll-up. The aerodynamic force is not perpendicular to the wing, but is more inclined to the wing surface (Fig. 13.4b), implying that viscous forces are also important in upstroke. It is clear from Fig. 13.4 that the upward component of aerodynamic force produced during the downstroke is much higher than the downward component of aerodynamic force generated during the upstroke.

The above analysis reveals a remarkable feature of the inclined stroke plane kinematics: insects utilize their tiny wing as a bluff body during downstroke, producing enormous pressure drag and as a streamlined body during upstroke, producing low skin-friction drag and this difference in drag helps insects to hover. Additional analyses (not discussed here) have confirmed that the delayed stall, which is the most important aerodynamic mechanism in horizontal stroke plane motions has marginal significance in inclined stroke plane kinematics (Sudhakar and Vengadesan, 2010b).

13.4 Wing Interference Effects

We discussed the mechanism of vertical force generation considering a single flapping wing in the previous section. Insects like dragonflies have two wings in tandem. To study the effect of wing interference, we consider tandem wings hovering in a qui-



(a) Mid-downstroke

(b) Mid-upstroke

Fig. 13.4 Contours of vorticity and the instantaneous force acting on the wing during the middle of the half-strokes for Re = 150

escent fluid. The study demonstrates the feasibility of IBM in the study flows involving multiple moving bodies. The computational domain, boundary conditions, and the kinematics used in the study are shown in Fig. 13.5. We have used the following kinematics proposed by Wang (Wang, 2004):

$$[\mathrm{x}(t), \mathrm{y}(t)] = \frac{A_0}{2c}\cos(2\pi ft + \psi)(\cos\beta \sin\beta), \tag{13.18}$$

$$\alpha(t) = \alpha_0 - \alpha_m \sin(2\pi ft + \phi + \psi), \tag{13.19}$$

$$\mathrm{C_H} = \frac{\mathrm{F_H}}{\frac{1}{2}\rho \mathrm{U}^2 c}, \qquad \mathrm{C_V} = \frac{\mathrm{F_V}}{\frac{1}{2}\rho \mathrm{U}^2 c}, \tag{13.20}$$

where $[\mathrm{x}(t), \mathrm{y}(t)]$ is the position of the center of chord of the wing, $\alpha(t)$ is the angle made by the chord with the stroke plane, β is the stroke plane angle, ϕ is the phase difference between translation and rotation, f is the frequency of flapping, and A_0/c and α_m are the amplitudes of translation and rotation, respectively.

Velocity scale, $\mathrm{U} = \pi(A_0/c)f$, is related to oscillating translation. Reynolds number, $\mathrm{Re} = \mathrm{U}c/\nu = \pi f A_0 c/\nu$, is based on the maximum velocity of translation and the chord length. $T = 1/f$ is the time period of flapping. $\mathrm{C_H}$ and $\mathrm{C_V}$ represent the horizontal and vertical force coefficients respectively and ψ is the phase difference between flapping of forewing and hindwing. The size of the computational domain is $20c \times 20c$. The wing is immersed in a uniform grid of size $\Delta\mathrm{x}, \Delta\mathrm{y} = 0.01c$ and stretched everywhere else. The corresponding grid size is 992×992. We employ vorticity contours and backward finite-time Lyapunov exponent (FTLE) ridges to explain the time-varying forces resulting from the vortex dynamics.

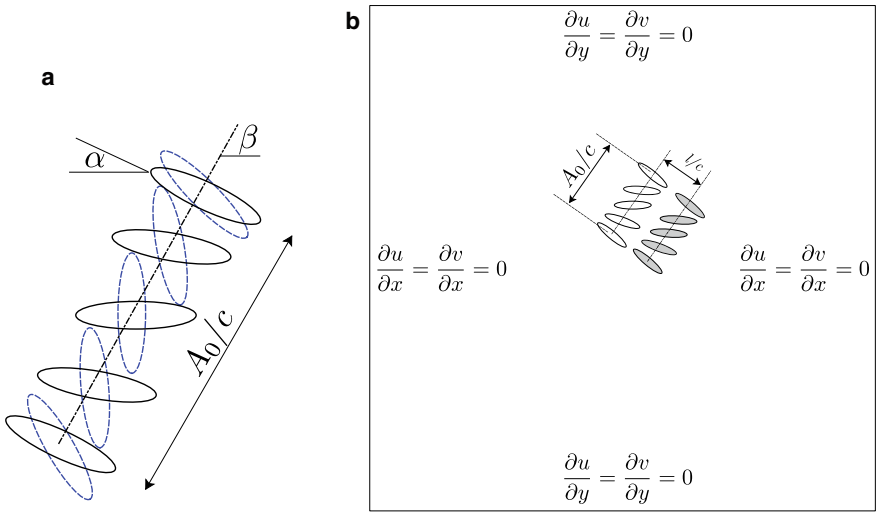


Fig. 13.5 **a** Positions of the elliptical foil in a flapping cycle is shown here. Solid lines represent the positions in downstroke and dashed lines represent the positions during upstroke. $\beta = \pi/3$ is the stroke plane angle, $A_0/c = 2.5$ stroke length, $\alpha_0 = \alpha_m = \pi/4$ is the maximum angle of attack, and Re $= 100$ in the study. **b** Boundary conditions and the computational domain used in the study are given here. Forewing and hindwing are represented by white and gray foils, respectively. The perpendicular distance between the stroke planes of the two wings is represented by l/c

The visualization of hovering flapping wings are presented in the following section to emphasize the importance of Lagrangian coherent structures (LCS) in the study of unsteady vortex dynamics. Further information about the calculation of backward FTLE can be found in Srinidhi and Vengadesan (2017b). The Reynolds number Re is 100 and the ratio of minor axis to major axis of the ellipse is 0.25. LCS in Fig. 13.7 show the attracting dynamic structures in the flow which entrain the surrounding fluid. Time-dependent behavior of individual vortices like stretching and merging can be kept track of LCS.

13.4.1 Force Variation and Vortical Structures in Tandem Wing Hovering

To study the effect of wing interference, the inter-wing distance and phase difference are varied. Inter-wing distances of $l = 1.1c$, $1.2c$, $1.3c$, $1.5c$, $1.7c$, $1.9c$ $2.1c$ and phase differences $\psi = 0°$ and $180°$ were considered in the study. The variation of C_V reaches a periodic state in 4–5 flapping cycles and the forces are averaged over a flapping cycle after the tenth cycle. The time-averaged vertical and horizontal force coefficients are denoted by $\overline{C}_V$ and $\overline{C}_H$, respectively. In hovering, the weight of the insect is supported by the vertical force and as such we focus further discussions

only on the variation of C_V. In this section, the variation of time-averaged force with inter-wing distance, the effect of wing kinematics on the force variation in a flapping cycle and the effect of phase difference between the forewing and the hindwings are presented.

13.4.1.1 In-Phase Stroking, $\psi = 0°$

Figure 13.6 shows that $\overline{C}_V$ of the forewing is larger in magnitude than the $\overline{C}_V$ of a single wing. The presence of the forewing results in the decrease of the vertical force on the hindwing; whereas, the converse is true in the case of the forewing and the effects of the wing interference reduce as the inter-wing distance increases. The vertical force generated by the flapping wings depends on the unsteady vortex dynamics. The evolution of vorticity and the corresponding variation of C_V at $l = 1.1c$ for the same are shown in Figs. 13.7 and 13.8, respectively.

Hindwing constantly operates in the wake of the forewing; due to the effect of wake on the LEV generation, C_V of the hindwing is less compared to the forewing. In the downstroke, the presence of trailing edge vortex of the forewing (TEV_F, Fig. 13.7b) has a detrimental effect on the growth of the LEV of the hindwing. In the upstroke, the hindwing is nearly vertical and it constantly moves in the downwash created by the forewing. The downwash increases the drag on the surface of the hindwing, consequently, C_V of the hindwing is less than a single-wing flapping system.

Figures 13.8a, b show the time variation of C_V at different inter-wing distances. For the sake of comparison, C_V variation of single-wing flapping is also plotted (dashed line). From Fig. 13.6, it is clear that forewing generates more force than a single flapping wing system, this shows that the presence of the hindwing enhances the force generation of the forewing. In Fig. 13.8a, b, the initial peak in vertical force at $t/T \approx 0.05$ is due to the reaction force provided by the fluid due to acceleration

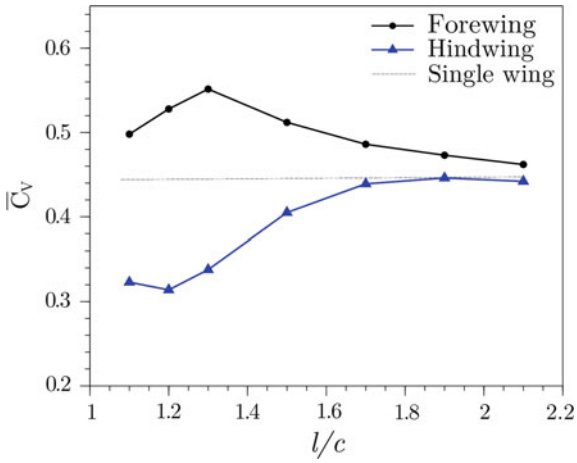
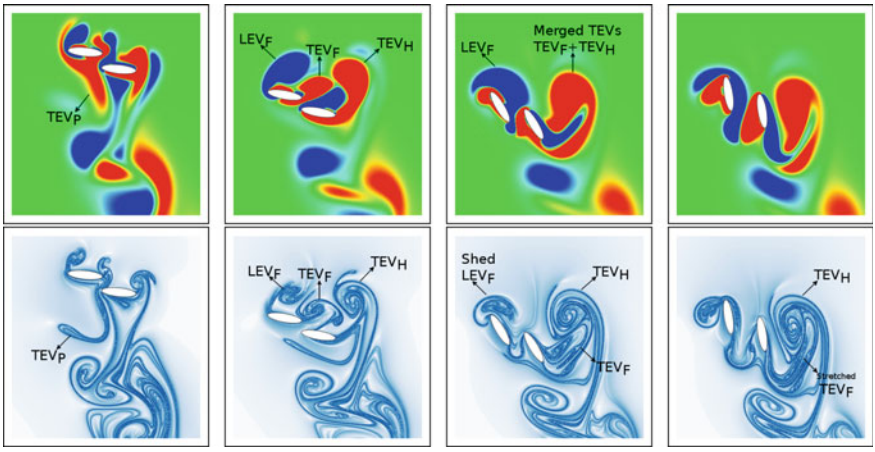


Fig. 13.6 $\overline{C}_V$ versus l/c of both fore and hindwing



(a) $t/T = 0.1$ (b) $t/T = 0.3$ (c) $t/T = 0.5$ (d) $t/T = 0.7$

Fig. 13.7 Evolution of vorticity with time and LCS for $l = 1.1c$. Vorticity and LCS are plotted in alternating rows. Subscripts F and H represent fore and hindwings, respectively. Subscript P corresponds to the residual vorticity from the previous stroke or the shed vortex in the wake

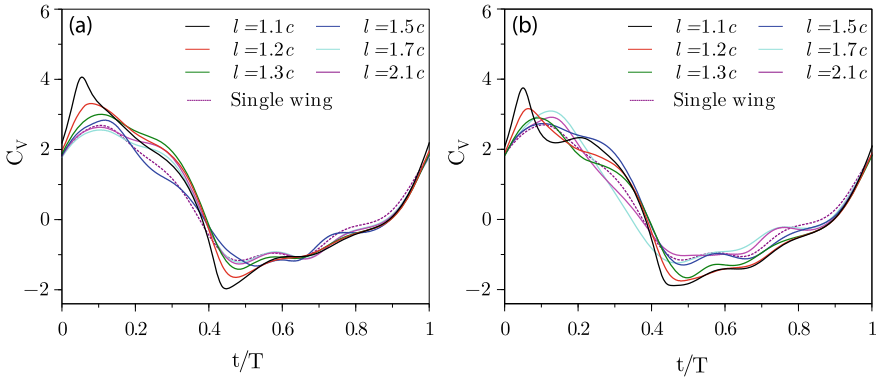


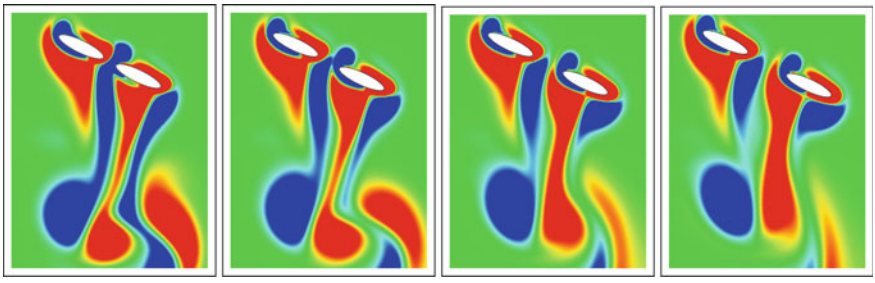
Fig. 13.8 **a** and **b** Time-varying C_V of forewing and hindwing, respectively

of the wings and rapid pitch-down rotation of the wings (Meng and Sun, 2016). Also, the counterclockwise (CCW) wake vortex represented by TEV_P (Fig. 13.8a, subscript P represents vorticity from the previous cycle) interacts with the wing and transfers momentum. This interaction, the so-called wake capture, also enhances the force. As the wings continue their downstroke, clockwise (CW) LEV is developed at the leading edge of the wings in accordance with the delayed stall mechanism (Fig. 13.7c). TEV_P is captured accurately in the LCS plot of Fig. 13.7a. At $t/T \approx 0.3$, TEV_P interacts with the hindwing, this corresponds to the local maxima in $\overline{C}_V$ of the hindwing (Fig. 13.8b). LEVs and TEVs of the forewing and the hindwing are represented by LEV_F, TEV_F and LEV_H, TEV_H, respectively. At $t/T \approx 0.5$, TEV

of the forewing is shed (TEV_F in Fig. 13.7c), and it interacts with the hindwing. As the wings start pitching up, due to the deceleration of the wings, $\overline{C}_V$ reaches its minimum value. The shed TEV of the forewing merges with the TEV of the hindwing ($TEV_F + TEV_H$ in Fig. 13.7d). The phenomenon of vortex merging is captured with finesse in the LCS contours. TEV_F gets sheared, stretched, and ultimately merges with TEV_H. As the wings continue with the upstroke, merged TEVs, and shed LEVs entrain surrounding fluid and transfer momentum. The jet created by the counter-rotating vortices forms a part of the total vertical force in the upstroke and the beginning of the downstroke.

Effect of inter-wing distance on force generation:

Figure 13.6 shows the effect of inter-wing distance on the cycle averaged vertical force. $\overline{C}_V$ of the hovering single wing is 0.446 (dashed line). Figure 13.9 presents the vorticity contours at various inter-wing distances. As the inter-wing distance increases, the effect of the forewing on the LEV generation of the hindwing decreases and $\overline{C}_V$ of the hindwing increases and reaches the $\overline{C}_V$ value of single flapping wing. When the wings are very close to each other, they act as a single system and the added mass effect which depends on the shape of the body and the acceleration of the fluid is prominent. As the separation between the wings increases, the added mass effect decreases and consequently the initial peak in C_V decreases (Fig. 13.8a, b). Figure 13.6 shows that $\overline{C}_V$ of fore and hindwings asymptotically reach $\overline{C}_V$ of the single-wing values at large enough inter-wing distances. In the downstroke of the wings, the maximum influence of the delayed stall mechanism on LEV generation occurs between $t/T = 0$ and 0.5. As the inter-wing distance increases, the effect of forewing downwash on the LEV generation of the hindwing decreases. Consequently, LEV of the hindwing grows in size and the vertical force generated by the hindwing increases. For $l < 1.3c$, the width of the wake increases as the l increases. For $l > 1.3c$, the width of the wake decreases as the inter-wing distance increases. Decrease in the width of the wake reduces the force generation. It is clear from Fig. 13.7 that LCS reveals structures which are otherwise hidden in vorticity plots.



(a) $l = 1.3c$ (b) $l = 1.5c$ (c) $l = 1.7c$ (d) $l = 1.9c$

Fig. 13.9 Vorticity contours for different l at the end of the downstroke. As l increase size and strength of LEV and TEV increases

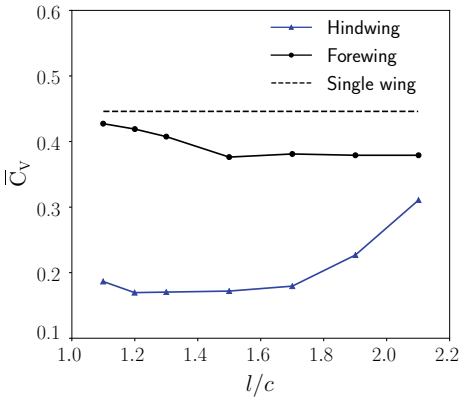


Fig. 13.10 $\overline{C}_V$ of both forewing and hindwing

13.4.1.2 Counter-Stroking, $\psi = 180°$

Figure 13.10 shows that the phase difference of $\psi = 180°$ reduces the variation of time-averaged vertical with the increasing inter-wing distance. Figures 13.11a, b represent the time variation of C_V for different l/c, for $\psi = 180°$. From Fig. 13.11a, it is clear that the presence of hindwing has a marginal effect on the C_V of the forewing. Hindwing in its downstroke generates lesser force compared to a single flapping wing, as it operates in a lower pressure area created by the shed TEV of the forewing. As the inter-wing distance increases, the effect of forewing on the force generation of hindwing decreases and consequently, for $l > 1.5c$ vertical force generation of the hindwing increases rapidly (Fig. 13.11b). Typical variation of C_V is explained for $l = 1.1c$. Figure 13.12 shows the evolution of vorticity and LCS contours over time.

Hindwing is at the beginning of its upstroke when the forewing is at the beginning of its downstroke (Fig. 13.12a). The initial peak in C_V is reduced because of the presence of the LEV of the hindwing (Fig. 13.11a). At $t/T = 0.3$, TEV of the forewing interacts with the CCW vorticity of the hindwing, creating a low-pressure region near the lower surface of the forewing (Fig. 13.12b). This corresponds to the minima in C_V of the forewing at $t/T \approx 0.3$. As the forewing moves away from the hindwing (Fig. 13.12c), the pressure on the lower surface of the wing increases and consequently C_V increases (Fig. 13.12c) and reaches a local maxima at $t/T = 0.3$. At $t/T = 0.3$, TEV of the forewing and CCW vorticity of the hindwing merge together and form a region of low pressure between the two wings (Fig. 13.12c). As the hindwing starts its downstroke, it interacts with the merged CCW vortex, this wake capture increases the vertical force generation of the hindwing. In Fig. 13.12c, it is clear that the C_V maxima at $t/T = 0.6$ is greater in magnitude than C_V of a single-wing flapping due to the interaction with the merged vortex (Fig. 13.12d).

After the wake capture, as the hindwing continues its downstroke, the downwash of the forewing and the LEV shed by the forewing result in a sudden fall in the vertical force. For the rest of its downstroke, the hindwing operates in a low-pressure

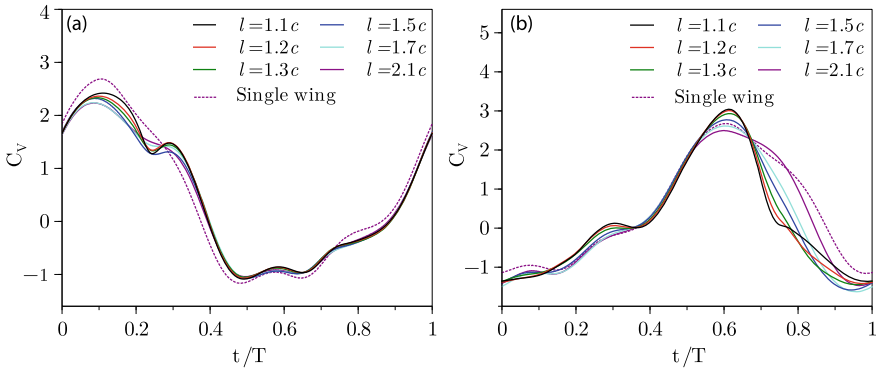
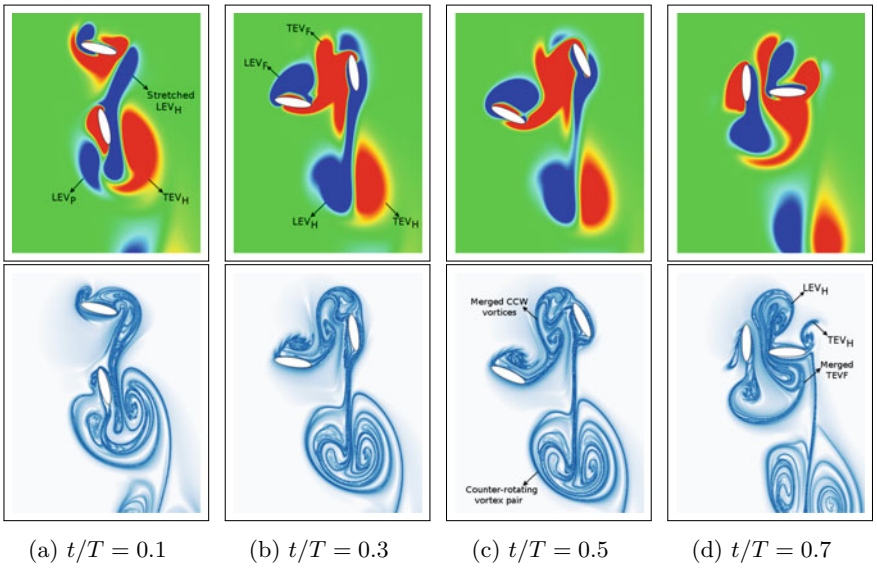


Fig. 13.11 **a** and **b** Time-varying C_V of forewing and hindwing, respectively



(a) $t/T = 0.1$ (b) $t/T = 0.3$ (c) $t/T = 0.5$ (d) $t/T = 0.7$

Fig. 13.12 Evolution of vorticity with time and LCS for $l = 1.1c$. Vorticity and LCS are plotted in alternating rows

region created by the shed LEV of the forewing, the merged vortex and the growing LEV. This results in the sudden drop in the vertical force generation of the hindwing for $t/T > 0.7$. In comparison with a single-wing flapping where the delayed stall mechanism generates much of the vertical force. Besides, a part of the shed LEV of the forewing and the CW shear layers merge with the LEV of the hindwing (Fig. 13.12d) and enhance the delayed stall effect. This results in a further reduction of pressure around the wing and is the major reason for the decreased vertical force generation of the hindwing. A pair of counter-rotating vortices are shed in every flapping cycle. The

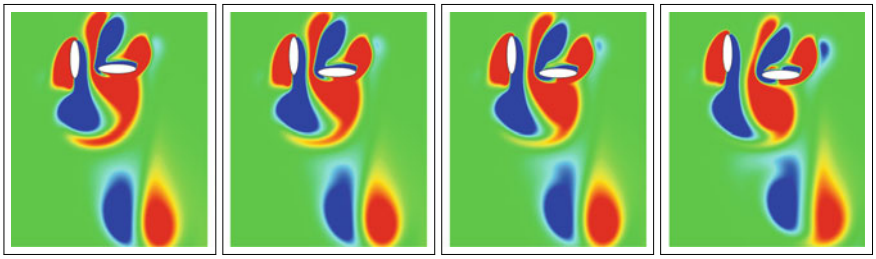


Fig. 13.13 Vorticity contours for different l at the end of the downstroke. As l increase size and strength of LEV and TEV increases

vortex pair has low swirl component. This wake with predominant vertical velocity increases the stability of the body in hovering.

Effect of inter-wing distance:

Figure 13.13 presents the vorticity contours at different inter-wing distances. $\overline{C}_V$ of the forewing slightly decreases initially, for $l < 1.5c$ and remains nearly constant at greater distances. The effect is apparent as the vortical structures of the forewing look similar. $\overline{C}_V$ of the hindwing increases slowly for $l < 1.5c$ and increases rapidly for $l > 1.5c$. For $l \geq 1.7c$, there is no formation of the merged vortex, this reduces the peak C_V of the hindwing at $t/T = 0.6$ (Fig. 13.11b) as the effect of wake capture is diminished. For $l > 1.5c$, due to the reduced wing–wing interactions, and decreased effect of merged vortex in reducing the pressure around the hindwing, the delayed stall mechanism becomes more effective in the force generation (Fig. 13.11b).

13.5 Effect of Ground on the Force Generation

In this section, we study the effect of ground on the vortex dynamics and the force production of a single flapping wing. The study is carried out in a domain of size: $-20c \leq x \leq 20c$, $-1c \leq y \leq 20c$ on a grid with uniform grid dimensions of $0.01c$. Figure 13.14 represents the details of the computational domain and boundary conditions used in the study.

Figure 13.15a represents the $\overline{C}_V$ versus the ground clearance D/c. Figure 13.15b shows the time-varying vertical force C_V versus the non-dimensionalized time plotted at different heights from the ground. For the sake of clarity, only C_V variations pertaining to $D/c = 0.5$, 1, 2, 3, 5, and without ground effect cases are plotted.

Similar to Gao and Lu (2008), the variation of $\overline{C}_V$ can be grouped into three regimes: force enhancement, force reduction, and force recovery regimes. For $D/c < 2$, as the wing moves closer to the ground, force generation increases, and the ground effect is dominant; the force behavior lies in the force enhancement regime. For $2 < D/c < 4$, $\overline{C}_V$ is lesser than the values for the case with $D/c = 0.5$ as well as

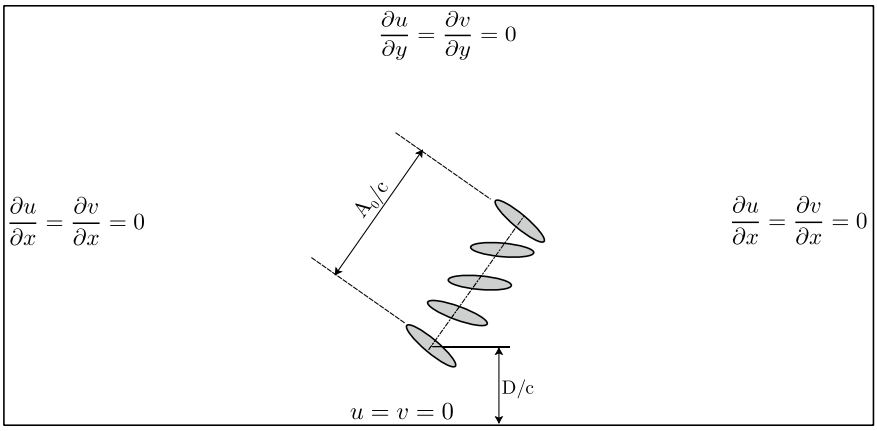


Fig. 13.14 Computational domain and the boundary conditions used in the study are represented here. The non-dimensional clearance from the ground D/c is the vertical distance between the center of the wing and the ground when the wing is at the end of its downstroke

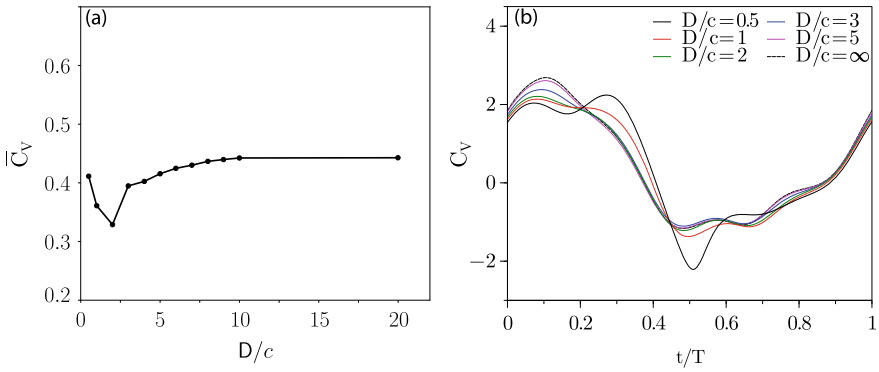
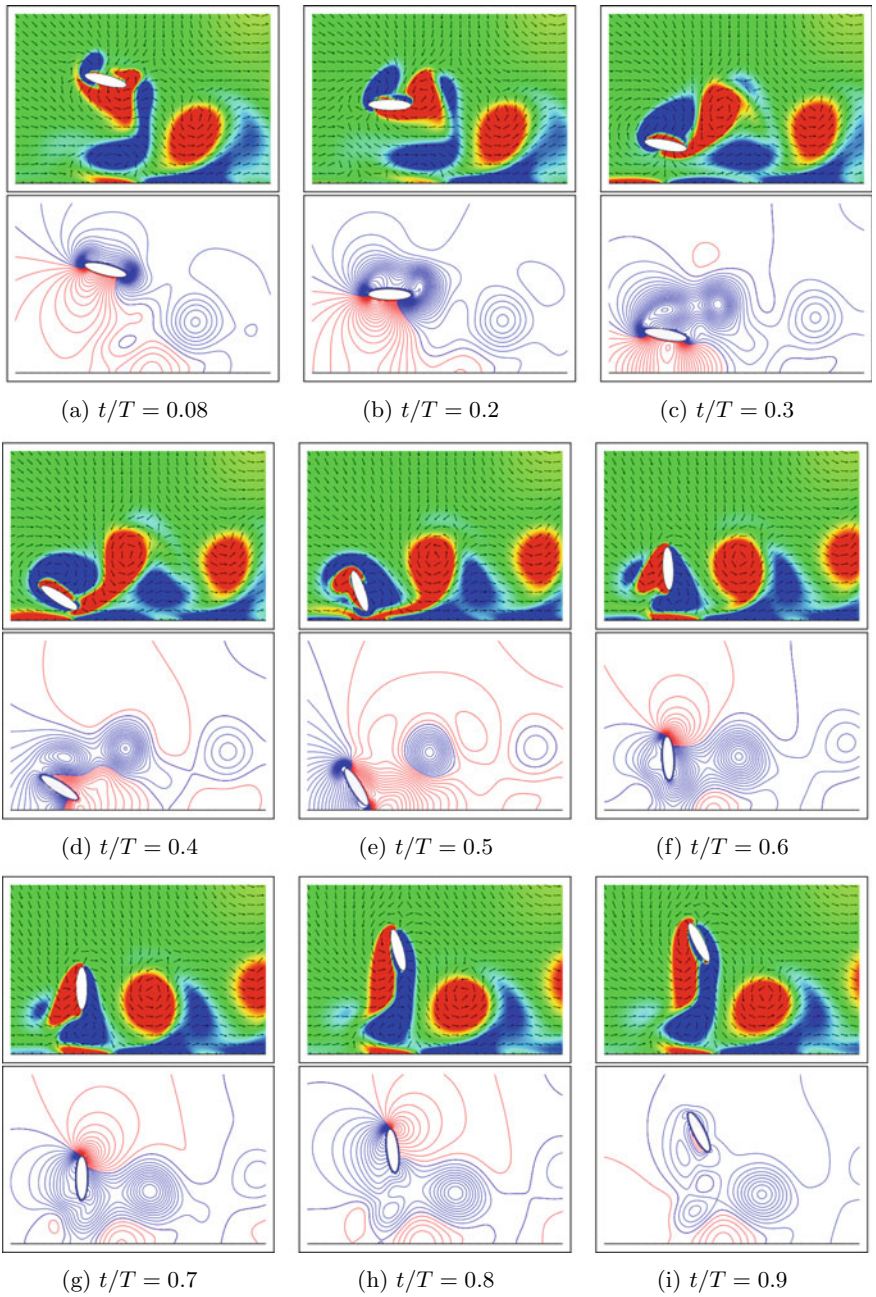


Fig. 13.15 **a** Time-averaged vertical force versus ground clearance. **b** Vertical force versus time

$\overline{C}_{V_\infty}$, and the force behavior changes from force enhancement to force recovery regime. This regime is called force reduction regime. In the force recovery regime ($D/c > 4$), the wing experiences increased vertical forces due to the reverse Kármán vortex shedding and $\overline{C}_V$ slowly reaches $\overline{C}_{V_\infty}$. Figure 13.16 shows the evolution of vortical structures with time, along with the corresponding pressure contours for $D/c = 0.5$. Velocity vectors are superimposed on vorticity plots to visualize the interaction between fluid and the ground.

In Fig. 13.15b, C_V initially increases and reaches its maximum at $t/T \approx 0.08$. This initial peak is due to the acceleration of the wing, and the rapid pitch-down rotation. After the initial peak, C_V slowly decreases and starts increasing at $t/T \approx 0.2$. In Fig. 13.16a, b, the vortex which is near the lower surface of the wing adds a CW circulation to the fluid displaced by the wing, this reduces the effect of the ground on



(a) $t/T = 0.08$ (b) $t/T = 0.2$ (c) $t/T = 0.3$

(d) $t/T = 0.4$ (e) $t/T = 0.5$ (f) $t/T = 0.6$

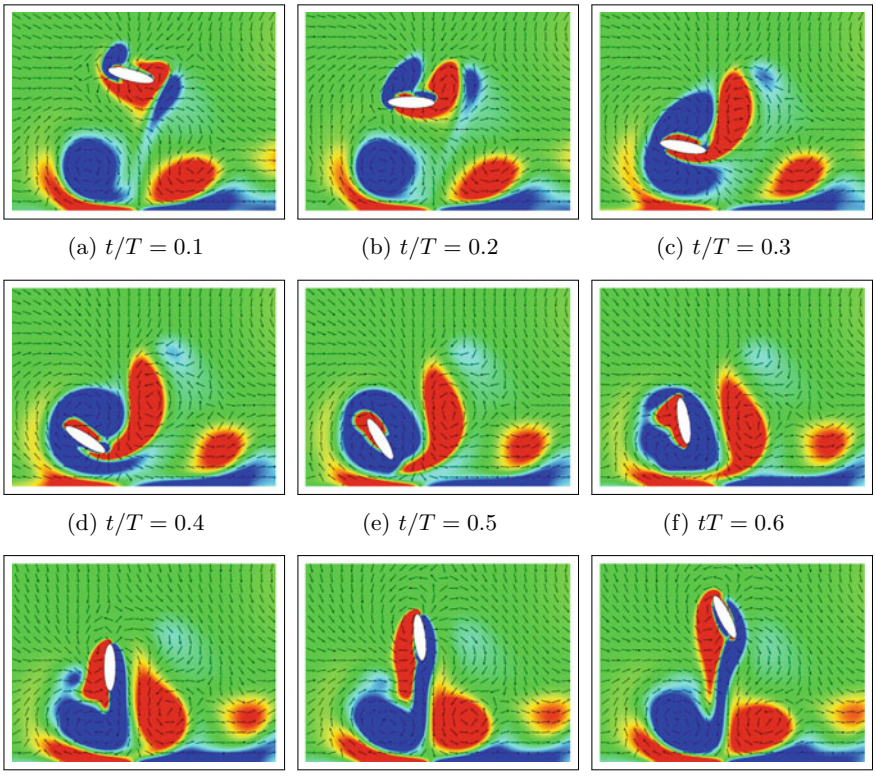
(g) $t/T = 0.7$ (h) $t/T = 0.8$ (i) $t/T = 0.9$

Fig. 13.16 Evolution of vorticity and pressure with time at $D/c = 0.5$. Velocity vectors are superimposed on vorticity. Vorticity and pressure are plotted in alternating rows. Red represents CCW vorticity and positive pressure, and blue represents CW vorticity and negative pressure. Contour levels for both vorticity and pressure are from -1 to 1

the generation of vertical force. As the wing continues its downstroke, the CW vortex advects downstream and away from the wings facilitating cushion effect; this results in the direct impingement of the fluid on the ground. The increasing pressure on the lower surface of the wing causes an increase in C_V. This effect is analogous to a jet impinging on a surface. Along with the cushion effect, the formation of leading-edge vortex (LEV) in accordance with the delayed stall mechanism creates a low-pressure region on the upper surface of the wing. A counterclockwise (CCW) trailing edge vortex (TEV) is also created at the trailing edge of the wing. LEV coupled with the cushion effect is the reason for the high $\overline{C}_V$ observed in the near ground cases. The flow generated by the wing creates shear layers on the ground. As the wing starts pitching up, LEV and TEV of the wing are shed (Fig. 13.16c) causing a total loss of lift. At $t/T \approx 0.5$, the wing interacts with the shear layer on the ground and disrupts it (Fig. 13.16e). Due to the severe gradients created, C_V becomes minimum. At $t/T \approx 0.5$, the wing starts its upstroke and the lift slowly starts increasing because of the induced velocity of the jet created by the shed LEV and the shed TEV. For most part of the upstroke, the wing is vertical and is surrounded by a low-pressure region (pressure plots of Fig. 13.16f–h), as a result, the lift generated in the upstroke is less compared to the downstroke.

Figure 13.17 shows the evolution of vortical structures at $D/c = 1$. Important flow features in the present case are the rebound vortices. LEV and TEV shed in the previous stroke strike the ground and rebound, and the shed LEV forms a low-pressure region below the lower surface of the wing (Fig. 13.17a). Between $t/T = 0.2$ and 0.4, the wing interacts with the CW rebound vortex, (Fig. 13.17b–d). The presence of the rebound vortex reduces the cushion effect on the wing, as a result, C_V decreases compared to $D/c = 0.5$. The LEV shed at the end of the downstroke strikes the ground and forms a new rebound vortex. An important observation we made is the change in the effective angle of attack (AoA) of the wing caused by the flow created by the CW rebound vortex. When the wing is in its downstroke, the circulation added by the rebound vortex to the surrounding fluid changes the AoA of the wing. Depending on the size and strength of the CW rebound vortex, the vertical force generated by the wing may either increase or decrease.

At $D/c = 2$, the induced velocity of the jet created by the rebound vortices generates most of the force. The prominent flow feature at this ground clearance is the presence of a sustained CW rebound vortex (Fig. 13.18a). LEV shed by the wing at the end of the downstroke feeds the rebound vortex from the previous stroke. Shed TEV of the wing gets stretched by the shear layer at the ground, loses its strength and eventually dissipates. As the ground clearance increases, the induced velocity of the jet created by the shed vortices dominates the force generation. The flow structures at the beginning of the downstroke for $D/c = 2$, 5, and out-of-ground effect cases are shown in Fig. 13.18. The figures show that the effect of ground is negligible for $D/c > 5.0$ as flow structures are almost similar (Fig. 13.18b, c).

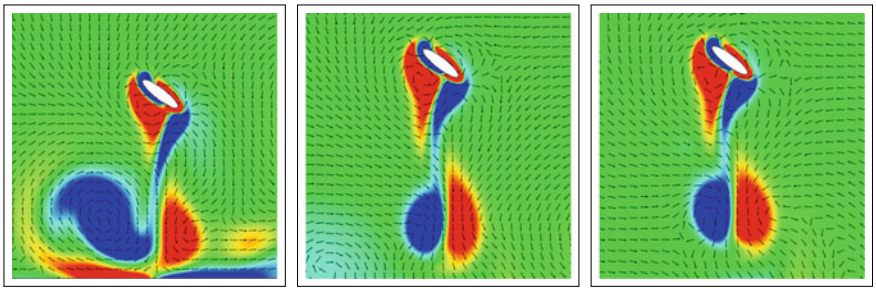


(a) $t/T = 0.1$ (b) $t/T = 0.2$ (c) $t/T = 0.3$

(d) $t/T = 0.4$ (e) $t/T = 0.5$ (f) $tT = 0.6$

(g) $t/T = 0.7$ (h) $t/T = 0.8$ (i) $t/T = 0.9$

Fig. 13.17 Evolution of vorticity with time for $D/c = 1$. Velocity vectors are superimposed on vorticity to visualize ground effect



(a) $D/c = 2$ (b) $D/c = 5$ (c) $D/c = \infty$

Fig. 13.18 Vorticity contours at $t/T = 0.1$. Velocity vectors are superimposed on vorticity to visualize ground effect

13.5.1 A Note on Three-Dimensionality and Wing Flexibility

In this article, we have not considered two important effects relevant to insect flight: (1) finite aspect ratio of the wings and (2) wing flexibility. Various researchers have employed IBM, due to its versatility, to study the influence of these two and related parameters on the insect aerodynamics. These studies focused on the effect of the following parameters on the force production by flapping insect wings: three-dimensional wingtip vortices on finite aspect ratio wings (Moriche et al. 2016), effect of wing kinematic parameters (Han et al. 2018), complex maneuvering (Bode-Oke et al. 2018), wing flexibility (Shahzad et al. 2018), fluid–structure–acoustics interaction (Wang and Tian 2019), and complete wing-body models (Minami et al. 2014).

13.6 Conclusion

In this chapter, we presented the application of immersed boundary projection method to study the unsteady aerodynamics of insect flight. Following a brief review of the unsteady aerodynamic mechanisms involved in the insect flight, we presented the numerical implementation of the moving-body and multi-processor implementation of the present IBPM algorithm. The unsteady flow structures and aerodynamic forces acting on an idealized 2D dragonfly model wing were studied by numerically solving the N-S equations with the IBPM formulation. The chapter covered the kinematics and flow physics of the flapping flight with the focus on three major aspects of the flight: (1) the mechanism of vertical force generation, (2) the forewing–hindwing interaction in the case of tandem wings, and (3) the effect of ground on force generation. Spatio-temporal dynamics of vorticity field and Lagrangian coherent structures are used to understand the physics behind the force variation in inclined stroke plane kinematics. Our results suggest that the delayed stall mechanism is not the dominant lift generation mechanism in the case of such kinematics. Insects using inclined stroke kinematics use their wings as a bluff body in downstroke, and as a streamlined body during upstroke; this difference in operation helps in large vertical force generation. In the presence of tandem wings, in-phase stroking of the wings produces maximum vertical force and the out-of-phase stroking generates the least vertical force. Furthermore, the ground effect can be grouped into three regimes: force enhancement, force reduction, and force recovery regimes, depending on the non-dimensional ratio of distance between the ground to chord length.

It is worth emphasizing here that the immersed boundary method would be the ideal choice to simulate fluid flow over rapidly oscillating insect wings. Moreover, the study of interference effect and ground effect require handling multiple bodies coming very close to each other. IBM is instrumental in simulating flows around multiple bodies with complex kinematics. With conventional body-fitted methods,

it would be practically impossible to handle such situations without re-meshing operations and excessive human intervention. This situation is directly dealt with using immersed boundary methods, thus enabling us to easily study the physics in such scenarios.

References

Alexander DE (1984) Unusual phase relationships between the forewings and hindwings in flying dragonflies. J Exp Biol 109(1):379–383

Altshuler DL, Dickson WB, Vance JT, Roberts SP, Dickinson MH (2005) Short-amplitude high-frequency wing strokes determine the aerodynamics of honeybee flight. Proc Natl Acad Sci 102:18213–18218

Balay S, Abhyankar S, Adams MF, Brown J, Brune P, Buschelman K, Dalcin L, Dener A, Eijkhout V, Gropp WD, Kaushik D, Knepley MG, May DA, McInnes LC, Mills RT, Munson T, Rupp K, Sanan P, Smith BF, Zampini S, Zhang H, Zhang H (2019) PETSc web page

Birch JM, Dickson WB, Dickinson MH (2004) Force production and flow structure of the leading edge vortex on flapping wings at high and low Reynolds numbers. J Exp Biol 207(7):1063–1072

Bode-Oke AT, Zeyghami S, Dong H (2018) Flying in reverse: kinematics and aerodynamics of a dragonfly in backward free flight. J R Soc Interface 15(143):20180102

De Rosis A (2014) On the dynamics of a tandem of asynchronous flapping wings: lattice Boltzmann-immersed boundary simulations. Phys A Stat Mech Appl 410:276–286

De Rosis A (2015) Ground-induced lift enhancement in a tandem of symmetric flapping wings: lattice Boltzmann-immersed boundary simulations. Comput Struct 153:230–238

Dickinson MH, Lehmann FO, Sane SP (1999) Wing rotation and the aerodynamic basis of insect flight. Science 284(5422):1954–1960

Ellington CP (1984a) The aerodynamics of hovering insect flight. III. Kinematics. Philos Trans R Soc Lond B Biol Sci 305(1122):41–78

Ellington CP (1984b) The aerodynamics of hovering insect flight. VI. Lift and power requirements. Philos Trans R Soc Lond B Biol Sci 305(1122):145–181

Ellington CP, van den Berg C, Willmott AP, Thomas ALR (1996) Leading-edge vortices in insect flight. Nature 384(6610):626

Ennos R (1989) The kinematics and aerodynamics of the free flight of some Diptera. J Exp Biol 142:49–85

Fadlun EA, Verzicco R, Orlandi P, Mohd Yusof J (2000) Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161(1):35–60

Fry SN, Sayaman R, Dickinson MH (2005) The aerodynamics of hovering flight in Drosophila. J Exp Biol 208:2303–2318

Gao T, Lu XY (2008) Insect normal hovering flight in ground effect. Phys Fluids 20(8):087101

Gilmanov A, Sotiropoulos F (2005) A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies. J Comput Phys 207(2):457–492

Goldstein D, Handler R, Sirovich L (1993) Modeling a no-slip flow boundary with an external force field. J Comput Phys 105(2):354–366

Han J, Yuan Z, Chen G (2018) Effects of kinematic parameters on three-dimensional flapping wing at low Reynolds number. Phys Fluids 30(8):081901

Kim D, Choi H (2006) Immersed boundary method for flow around an arbitrarily moving body. J Comput Phys 212(2):662–680

Kolomenskiy D, Maeda M, Engels T, Liu H, Schneider K, Nave JC (2016) Aerodynamic ground effect in fruitfly sized insect takeoff. PLoS One 11(3):e0152072

Krishnan A (2015) Towards the study of flying snake aerodynamics, and an analysis of the direct forcing method. Ph.D. thesis
Lan SL, Sun M (2001) Aerodynamic properties of a wing performing unsteady rotational motions at low Reynolds number. Acta Mech 149(1–4):135–147
Liu Y, Liu N, Lu X (2009) Numerical study of two-winged insect hovering flight. Adv Appl Math Mech 1(4):481–509
Meng X, Sun M (2016) Wing kinematics, aerodynamic forces and vortex-wake structures in fruit-flies in forward flight. J Bionic Eng 13(3):478–490
Minami K, Suzuki K, Inamuro T (2014) Free flight simulations of a dragonfly-like flapping wing-body model using the immersed boundary-lattice Boltzmann method. Fluid Dyn Res 47(1):015505
Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261
Moriche M, Flores O, García-Villalba M (2016) Three-dimensional instabilities in the wake of a flapping wing at low Reynolds number. Int J Heat Fluid Flow 62:44–55
Norberg RÅ (1975) Hovering flight of the dragonfly *Aeschna Juncea* L., kinematics and aerodynamics. In: Swimming and flying in nature. Springer, Boston, pp 763–781
Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10(2):252–271
Platzer MF, Jones KD, Young J, Lai JCS (2008) Flapping wing aerodynamics: progress and challenges. AIAA J 46(9):2136–2149
Roma AM, Peskin CS, Berger MJ (1999) An adaptive version of the immersed boundary method. J Comput Phys 153(2):509–534
Sane SP (2003) The aerodynamics of insect flight. J Exp Biol 206(23):4191–4208
Shahzad A, Tian F-B, Young J, Lai JCS (2018) Effects of Hawkmoth-like flexibility on the aerodynamic performance of flapping wings with different shapes and aspect ratios. Phys Fluids 30(9):091902
Shyy W, Aono H, Chimakurthi SK, Trizila P, Kang CK, Cesnik CES, Liu H (2010) Recent progress in flapping wing aerodynamics and aeroelasticity. Prog Aerosp Sci 46(7):284–327
Srinidhi NG, Vengadesan S (2017a) Ground effect on tandem flapping wings hovering. Comput Fluids 152:40–56
Srinidhi NG, Vengadesan S (2017b) Lagrangian coherent structures in tandem flapping wing hovering. J Bionic Eng 14(2):307–316
Sudhakar Y, Vengadesan S (2010a) Flight force production by flapping insect wings in inclined stroke plane kinematics. Comput Fluids 39(4):683–695
Sudhakar Y, Vengadesan S (2010b) The functional significance of delayed stall in insect flight. Numer Heat Transf Part A Appl 58:65–83
Taira K, Colonius T (2007) The immersed boundary method: a projection approach. J Comput Phys 225(2):2118–2137
Usherwood JR, Lehmann FO (2008) Phasing of dragonfly wings can improve aerodynamic efficiency by removing swirl. J R Soc Interface 5(28):1303–1307
van Truong T, Kim J, Kim MJ, Park HC, Yoon KJ, Byun D (2013) Flow structures around a flapping wing considering ground effect. Exp Fluids 54(7):1575
Wang ZJ (2004) The role of drag in insect hovering. J Exp Biol 207(23):4147–4155
Wang ZJ, Russell D (2007) Effect of forewing and hindwing interactions on aerodynamic forces and power in hovering dragonfly flight. Phys Rev Lett 99(14):148101
Wang JK, Sun M (2005) A computational study of the aerodynamics and forewing-hindwing interaction of a model dragonfly in forward flight. J Exp Biol 208(19):3785–3804
Wang L, Tian F-B (2019) Numerical study of flexible flapping wings with an immersed boundary method: fluid-structure-acoustics interaction. J Fluids Struct 90:396–409
Wu JH, Sun M (2004) Unsteady aerodynamic forces of a flapping wing. J Exp Biol 207:1137–1150

Chapter 14
Hybrid Lagrangian–Eulerian Method-Based CFSD Development, Application, and Analysis

Namshad Thekkethil and Atul Sharma

14.1 Introduction

Fluid–structure dynamics (*FSD*)—a coupled interaction between fluid dynamics and structure dynamics—is one of the complex phenomena observed in nature and has led to the development of biomimetic-based engineering systems. Analysis of the FSD in the natural systems could lead to a better design of the biomimetic systems. Since experimental methods have several limitations with regard to physical model for the complex FSD phenomenon, computational methods can take the lead in the analysis of the experimentally challenging FSD problems. For the *computational fluid–structure dynamics* (*CFSD*), there are various types of methods that are based on independent advancements in *computational fluid dynamics* (*CFD*) and *computational structure dynamics* (*CSD*) along with a coupling between the CFD and CSD that can be either one-way or two-way. For a *one-way coupled CFSD*, the structure is rigid and subjected to a forced motion that is independent of fluid dynamic forces acting on the structure while the fluid flow depends on the kinematic conditions of the structure. For a *two-way coupled CFSD*, the fluid flow and motion and/or deformation of flexible/rigid structure are dependent on each other; the motion/deformation of the structure is caused by the fluid dynamic forces.

N. Thekkethil · A. Sharma (✉)
Indian Institute of Technology Bombay, Mumbai 400076, India
e-mail: atulsharma@iitb.ac.in

N. Thekkethil
e-mail: namshad.th@gmail.com

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_14

14.1.1 CFSD Development, Application, and Analysis

Computational fluid–structure dynamics involves the development of a software, its application for a fluid–structure dynamics problem to obtain scientifically exciting and engineering-relevant results, and analysis of the results for a unified cause-and-effect study (Sharma 2017). Historically, the Eulerian approach-based finite volume method (FVM) is usually preferred in CFD while CSD prefers Lagrangian approach-based finite element method (FEM). However, for CFSD, various combinations of Eulerian and Lagrangian methods are considered that are broadly classified into two approaches: monolithic and partitioned. The *monolithic approach* considers fluid and structure as a continuum and uses either Eulerian or Lagrangian approach throughout the domain, whereas the *partitioned approach* solves the fluid flow and the structure motion or/and deformation separately along with a coupling condition at the fluid–solid interface. The partitioned approach is further classified into three types: fully Lagrangian, arbitrary Lagrangian–Eulerian (ALE), and *hybrid Lagrangian–Eulerian* (*HLE*) methods.

Fully Lagrangian method (Belytschko and Kennedy 1975; Donea et al. 1976) considered both the fluid dynamics and structural dynamics in the Lagrangian system and was the first choice for CFSD. However, the Lagrangian method for fluid flow is limited to almost stationary fluid since the fluid flow leads to a distortion of the mesh. ALE method considers body-fitted mesh and involves dynamic meshing without the mesh distortion problems (Noh 1963). ALE methods are efficient for many classes of FSI problems; however, it is limited by the need to re-mesh and gets into the trouble of the mesh distortion at a larger deformation of the structure. The HLE methods are the best choice for large deformation of CFSD problems. It uses the Eulerian approach for CFD and the Lagrangian approach for CSD. The HLE method presented here was proposed in our recent work (Thekkethil and Sharma 2019) for both one-way and two-way coupled CFSD problems. The HLE method considers a physical law-based FVM (Sharma 2017) and a *level-set function-based immersed boundary method* (LS-IBM) for CFD and geometric nonlinear Galerkin FEM for CSD along with direct implementation of coupling conditions at the fluid–solid interface.

14.1.2 Immersed Boundary Method

A historical development for CFD simulation of flow across immersed complex-shaped body started with a finite difference method-based solution on a Cartesian grid that approximates the curved body as a stepped one. Later, a finite volume method-based solution on a body-fitted grid was proposed initially for a structured curvilinear grid and later for an unstructured grid. The FVM-based solution continued for many years; however, the progress in CFD application from flow across a stationary structure to a moving and/or deforming structure led to various numerical challenges in generating a *time-wise varying* body-fitted structured/unstructured

grid. This led to a renewed interest in the application of Cartesian grid although with a new form—non-body-fitted Cartesian grid.

Immersed boundary method (*IBM*) (Peskin 2002) is a numerical methodology for finite difference method (FDM) or FVM-based CFSD development on a *non-body-fitted* and *fixed* Cartesian grid that involves a special treatment for implementation of fluid–solid interface boundary conditions and also for the CFD solution on the partially filled fluid cells. IBM gained popularity during the last few decades. The motion/deformation of the immersed structure results in certain Cartesian fluid cells (near the fluid–solid interface) to be partially or entirely filled with the solid at certain time instants. Depending on the numerical method to handle the change in the fluid cells near the moving interface, many IBMs are available in the literature that can be broadly classified into two types (Mittal and Iaccarino 2005): continuous forcing IBM and discrete forcing IBM (Mittal and Iaccarino 2005). The discrete forcing IBM is further classified based on the direct or indirect implementation of fluid–solid interface boundary conditions. A *sharp-interface* IBM (Udaykumar et al. 2001; Mittal et al. 2008) considers the physically realistic sharp fluid–solid interface, while a numerically diffused fluid–solid interface is considered in a *diffused interface* IBM (Pan 2006; Patel and Natarajan 2018). Depending on the strategy used for the application of fluid–solid interface boundary conditions, various sharp-interface methods are available in the literature, such as ghost-cell-based IBM (Majumdar et al. 2001; Mittal et al. 2008) and cut-cell-based IBM (Udaykumar et al. 2001). Both methods use a certain type of interpolation for the application of fluid–solid interface boundary conditions.

14.1.3 Outline of the Chapter

In this chapter, we present an HLE method-based CFSD development in Sect. 14.3 and its application for analysis of various types of one-/two-way coupled CFSD problems in Sect. 14.4. The HLE method (Thekkethil and Sharma 2019) involves FVM and LS-IBM for fluid dynamics and geometric nonlinear Galerkin FEM for structural dynamics and is based on a partitioned approach. The associated conservation laws and the fluid–solid coupling conditions are presented in Sect. 14.2.

14.2 CFSD: Conservation Laws and Fluid–Solid Coupling Conditions

For any FSD problem, conservation laws for fluid flow and structure dynamics need to be satisfied along with a continuity of stress and kinematics as the coupling condition at the fluid–solid interface. For fluid flow, mass and momentum conservation laws are considered in the Eulerian form while a Lagrangian form of momentum conservation law is considered for motion as well as deformation of the structure.

14.2.1 Mass and Momentum Conservation Laws for a Fluid Control Volume: Eulerian Form

For an *incompressible* fluid control volume (CV) with volume Ω^v and surface Γ^v (Fig. 14.1a), the Eulerian form of unsteady mass and momentum conservation laws is given for a negligible body force as

$$\text{Mass: } \mathrm{M}^v_{\text{out}} - \mathrm{M}^v_{\text{in}} = 0 \tag{14.1}$$

$$\text{Momentum: } \frac{\partial}{\partial t}\left(\mathcal{M}\overrightarrow{u}\right)^v + A^v_{\text{out}} - A^v_{\text{in}} = \overrightarrow{\mathrm{F}}^{\,v}_{s} \tag{14.2}$$

where M^v_{in} and $\mathrm{M}^v_{\text{out}}$ are the mass flow rates while A^v_{in} and A^v_{out} are the momentum flow rates entering and leaving the CV, respectively. Furthermore, $\mathcal{M}$ is the mass, $\overrightarrow{u}$ is the velocity, and $\overrightarrow{\mathrm{F}}^{\,v}_{s}$ is the surface force acting on the surface Γ^v of the control volume.

14.2.2 Momentum Conservation Law for a Solid Control Mass: Lagrangian Form

For a solid control mass with volume Ω^m and surface Γ^m (Fig. 14.1b), the Lagrangian form of momentum conservation law is given for a negligible body force as

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathcal{M}\overrightarrow{u}\right)^m = \overrightarrow{\mathrm{F}}^{\,m}_{s} \text{ where } \overrightarrow{u} = \frac{\mathrm{d}\,\overrightarrow{\mathrm{d}}}{\mathrm{d}t} \tag{14.3}$$

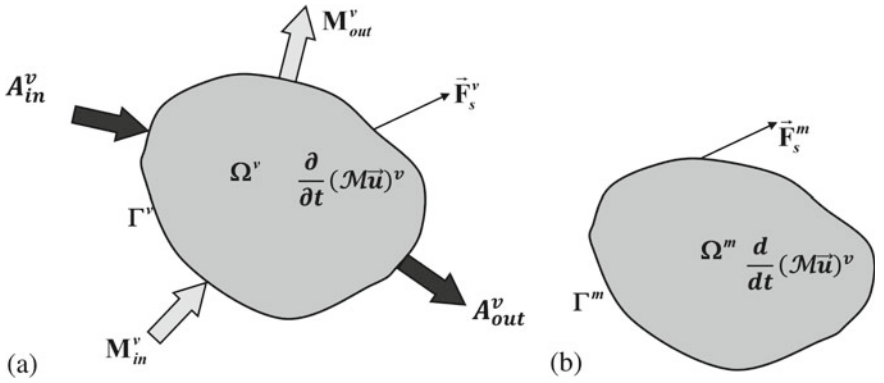


Fig. 14.1 **a** Mass and momentum conservation on a fluid *control volume* Ω^v with control surface Γ^v and **b** momentum balance in a solid *control mass* Ω^m with control surface Γ^m

Here, $\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathcal{M}\overrightarrow{u}\right)^m$ is the rate of change of instantaneous momentum of the control mass, $\overrightarrow{\mathrm{F}}_s^m$ is the surface force acting on the surface Γ^m, and $\overrightarrow{d}$ is the displacement vector of the control mass.

14.2.3 Fluid–Solid Coupling Conditions

Coupled fluid dynamics and structural dynamics govern the fluid–solid interface dynamics. The coupling is obtained by continuity of kinematics and stress field at the interface, given as

$$\overrightarrow{u}_{f,\mathrm{int}} = \overrightarrow{u}_{s,\mathrm{int}} \text{ and } \sigma_{s,\mathrm{int}}.\hat{n} = \sigma_{f,\mathrm{int}}.\hat{n} \tag{14.4}$$

Here, the subscripts f and s represent the fluid and structure, respectively, and int represents the fluid–solid interface. The vector $\hat{n}$ represents the unit normal vector at the interface.

14.3 HLE Method-Based CFSD Development: Hybrid FEM-FVM-Based Numerical Methodology

The present HLE method (Thekkethil and Sharma 2019) uses a form of conservation law that is Lagrangian for structure dynamics and Eulerian for fluid dynamics, presented in the previous subsection. Furthermore, the derivation of the algebraic formulations for the present HLE method-based CFSD development considers a physical law-based finite volume method (Sharma 2017) and a Galerkin finite element method (Zienkiewicz et al. 1977) for the fluid and structure dynamics, respectively. The physical law-based FVM starts with a discrete form of conservation laws, proposed by Sharma (2017) in a recent textbook on CFD as compared to starting with the partial differential equations (PDEs) in almost all the other FVM books on CFD (Patankar 2018; Versteeg and Malalasekera 2007). Both the physical law-based FVM and the PDE-based FVM use the same approximations and, thus, result in the same algebraic formulation for CFD.

Numerical methodology for the FVM-based CFD development and FEM-based CSD development and the associated coupling for HLE method-based CFSD development are presented in separate subsections below.

14.3.1 CFD Development: Physical Law-Based FVM and Level-Set Function-Based Immersed Boundary Method

CFD development consists of five steps (Sharma 2017): grid generation, FVM-based algebraic formulation, solution methodology, computation of engineering parameters, and testing. The first three steps of the CFD development are presented in separate subsections below for the present level-set function-based immersed boundary method (LS-IBM). The present LS-IBM involves a level-set function-based direct implementation of fluid–solid interface boundary condition (Shrivastava et al. 2013); thus, it avoids any interpolation for the interfacial boundary conditions.

14.3.1.1 Cartesian Grid Generation

For the development of a CFD solver, the present LS-IBM considers a fixed Cartesian grid, as shown in Fig. 14.2. For flow across a non-Cartesian or complex-shaped structure, as seen in Fig. 14.2, the non-body-fitted Cartesian grid results in certain partially filled fluid *control volumes* (CVs) that require special treatment to ensure mass and momentum conservation laws and no-slip boundary conditions. The figure shows the various types of CVs for the Cartesian grid.

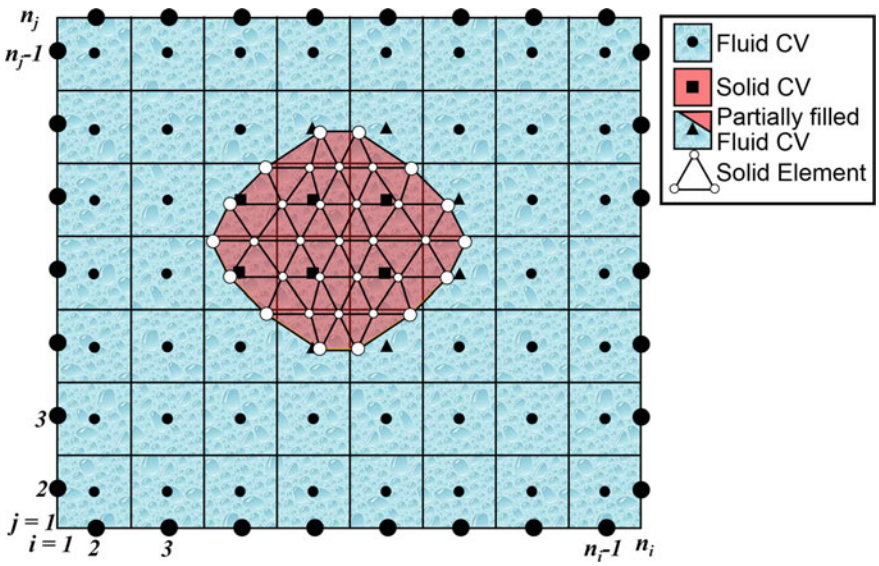


Fig. 14.2 Computational domain for a 2D FSI problem with a Lagrangian triangular mesh for structure immersed in the Eulerian Cartesian mesh in the complete domain

14.3.1.2 Physical Law-Based FVM

Figure 14.3a shows a computational stencil for a CV P whose neighbouring CVs are also fluid CVs. Considering the computational stencil, for a 2D incompressible flow, a mass and momentum conservation law-based FVM results in discrete mathematics-based approximated algebraic formulation, given (Sharma 2017) as

$$\left(m_{x,e}^{n+1} - m_{x,w}^{n+1}\right) \triangle y_P - \left(m_{y,n}^{n+1} - m_{y,s}^{n+1}\right) \triangle x_P = 0 \tag{14.5}$$

$$\rho_f \frac{\phi_P^{n+1} - \phi_P^n}{\triangle t} \triangle V_P + A_{\phi,P}^{n+1} = D_{\phi,P}^{n+1} + S_{\phi,P}^{n+1}$$

$$\text{where } \mathrm{A}_{\phi,P} = \left[\left(m_{x,e}^{+}\phi_e^{+} + m_{x,e}^{-}\phi_e^{-}\right) - \left(m_{x,w}^{+}\phi_w^{+} + m_{x,w}^{-}\phi_w^{-}\right)\right] \triangle y_P$$

$$\left[\left(m_{y,n}^{+}\phi_n^{+} + m_{y,n}^{-}\phi_n^{-}\right) - \left(m_{y,s}^{+}\phi_s^{+} + m_{y,s}^{-}\phi_s^{-}\right)\right] \triangle x_P$$

$$D_{\phi,P} = \mu_f \left[\left(\frac{\phi_E - \phi_P}{\delta x_e} - \frac{\phi_P - \phi_W}{\delta x_w}\right) \triangle y_P + \left(\frac{\phi_N - \phi_P}{\delta y_n} - \frac{\phi_P - \phi_S}{\delta y_s}\right) \triangle x_P\right]$$

$$S_{u,P} = (p_w - p_e) \triangle y_P, S_{v,P} = (p_s - p_n) \triangle x_P \tag{14.6}$$

where m_x and m_y are the components of mass flux in x- and y-directions, respectively, and the superscript $n + 1$ represents the time instant $(t + \triangle t)$.

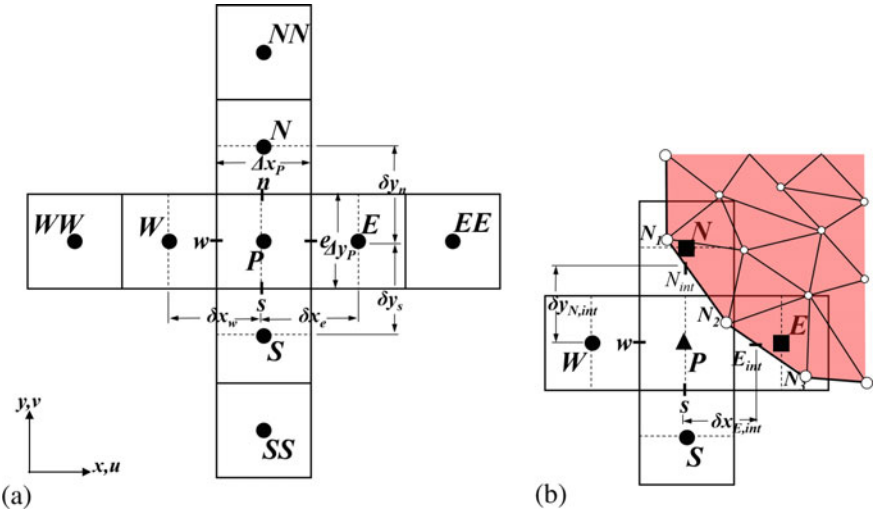


Fig. 14.3 Computational stencil for a fluid control volume with **a** all the neighbouring cells in the fluid and **b** north and east neighbours in the structure

For a *semi-implicit* solution methodology considered here, momentum equation considers the implicit time-level $(n+1)$ for all the terms (advection A, diffusion D, source S). Furthermore, $\phi = u$ and $\phi = v$ in Eq. (14.6) correspond to x-momentum and y-momentum conservations, respectively. The advection terms $A_{u,P}$ and $A_{v,P}$ are the advection of x-momentum flow rate and y-momentum flow rate, while the diffusion terms $D^{\chi}_{u,P}$ and $D^{\chi}_{v,P}$ are the viscous forces in x- and y-directions, respectively. Furthermore, ρ_f is the density of the fluid, μ_f is the viscosity of the fluid, and p is the pressure acting on the surface of the CVs. Also, in the advection term, the mass flux in the positive and negative directions $m^+ = \max(m, 0)$ and $m^- = \min(m, 0)$, and u/v velocity at the face centre $\phi_{f=e,w,n,s} = w_D\phi_D + w_U\phi_U + w_{UU}\phi_{UU}$ is obtained using an advection scheme (Sharma 2017). Here, D, U, and UU correspond to the downstream, upstream, and upstream-of-upstream values, respectively, and the weights w_D, w_U, and w_{UU} for the first-order upwind (FOU), second-order upwind (SOU), and quadratic upstream interpolation for convective kinematics (QUICK) schemes are obtained from a distance-based extrapolation/interpolation scheme (Sharma 2017).

14.3.1.3 Solution Methodology: Semi-implicit Pressure Projection Method

A *semi-implicit pressure projection method* (SIPPM) on a co-located grid system is used for the unsteady solution of the algebraic formulation—Eq. (14.5) for mass and Eq. (14.6) for momentum conservation. The velocity field $\overrightarrow{u}_P^{n+1}$ at a new time-level $(n+1)$ is obtained from the momentum conservation equation, while the mass conservation equation is converted into an algebraic equation for pressure (presented below), using a predictor–corrector method in the SIPPM. The predictor step involves prediction (represented by $*$ values) of velocity at cell centre $\overrightarrow{u}_P^{*}$ as well as normal velocities of mass fluxes at the face centres (u_e^*, u_w^*, v_n^*, and v_s^*). The predicted mass fluxes at the face centres are used to obtain the pressure p_P^{n+1} at the new time-level $(n+1)$ from the pressure equation. Finally, the pressure field is used to obtain the velocity correction (represented by $'$ values) at the cell centres $\overrightarrow{u}_P^{'}$ and then obtain $\overrightarrow{u}_P^{n+1} = \overrightarrow{u}_P^{*} + \overrightarrow{u}_P^{'}$. Formulation of the algebraic equations for $\overrightarrow{u}_P^{*}$, u_f^*/v_f^*, p_P^{n+1}, and $\overrightarrow{u}_P^{'}$ is presented below for the SIPPM.

Original Proposition:

Using Eq. (14.6) with $\phi = u$ or v, the velocity at the cell centre u_P^{n+1} and that at the east face centre u_e^{n+1} are given as

$$\begin{aligned}
\rho_f \frac{u_P^{n+1} - u_P^n}{\triangle t} \triangle V_P + A_{u,P}^{n+1} &= D_{u,P}^{n+1} + \left(p_w^{n+1} - p_e^{n+1}\right) \triangle y_P \\
\rho_f \frac{u_e^{n+1} - u_e^n}{\triangle t} \triangle V_e + A_{u,e}^{n+1} &= D_{u,e}^{n+1} + \left(p_P^{n+1} - p_E^{n+1}\right) \triangle y_P
\end{aligned} \tag{14.7}$$

Predictor step:

The projection method results in a velocity predictor equation for u_P^*/u_e^* obtained from the above equation, after dropping the pressure term, given as

$$\begin{aligned}\rho_f \frac{u_P^* - u_P^n}{\triangle t}\triangle V_P + A_{u,P}^* &= D_{u,P}^* \\ \rho_f \frac{u_e^* - u_e^n}{\triangle t}\triangle V_e + A_{u,e}^* &= D_{u,e}^*\end{aligned} \tag{14.8}$$

From the above implicit equation, u_P^* is obtained after an iterative solution while u_e^* is approximated by linear interpolation of the neighbouring cell-centre predicted velocity, i.e. $u_e^* = \overline{u_P^*, u_E^*}$; similarly, $u_w^* = \overline{u_P^*, u_W^*}$, $v_n^* = \overline{v_P^*, v_N^*}$, and $v_s^* = \overline{v_P^*, v_S^*}$ where v_P^* is obtained from equation similar to Eq. (14.8). Note that u_e^* is not obtained from the above implicit equation.

Corrector step:

Subtracting Eq. (14.8) for u_e^* from Eq. (14.7) for u_e^{n+1}, we get an approximate velocity correction as

$$u_e^{n+1} - u_e^* \approx \frac{\triangle t}{\rho_f}\frac{\left(p_E^{n+1} - p_P^{n+1}\right)}{\delta x_e} \tag{14.9}$$

$$\Rightarrow m_{x,e}^{n+1} \approx m_{x,e}^* - \triangle t\frac{\left(p_E^{n+1} - p_P^{n+1}\right)}{\delta x_e} \tag{14.10}$$

The approximations in the above equation correspond to neglecting the velocity correction corresponding to the advection and diffusion terms—resulting in the semi-implicit equation (Patankar 2018) although the original proposition is fully implicit [Eq. (14.6)].

Algebraic formulation for pressure:

Equations similar to Eq. (14.10) can be obtained for the mass fluxes at the other face centres, and substituting from these equations to the mass conservation Eq. (14.5), we obtain the pressure equation as

$$\begin{aligned}a_P p_P^{n+1} &= a_E p_E^{n+1} + a_W p_W^{n+1} + a_N p_N^{n+1} + a_S p_S^{n+1} + b \\ &\text{where } a_E = \frac{\triangle t \triangle y_P}{\delta x_e}, a_W = \frac{\triangle t \triangle y_P}{\delta x_w}, a_N = \frac{\triangle t \triangle x_P}{\delta y_n}, a_S = \frac{\triangle t \triangle x_P}{\delta y_s}, \\ a_P &= a_E + a_W + a_N + a_S, b = -S_{m,P}^* \\ &= -\left[\left(m_{x,e}^* - m_{x,w}^*\right)\triangle y_P + \left(m_{y,n}^* - m_{y,s}^*\right)\triangle x_P\right]\end{aligned} \tag{14.11}$$

Special Treatment for Partially Filled Fluid CVs

Figure 14.3b shows a control volume P with its east and neighbouring north cells in the structure. The solution procedure is same as other cells except for the computation of advection, diffusion, and pressure terms on the faces whose adjoining CV is a solid CV. For the control volume P in Fig. 14.3b, considering the east and north interfaces as horizontal and vertical lines, the advection and diffusion fluxes at the east and north sides are computed at E_{int} and N_{int}, respectively (instead of e and f), using the values of velocity at the solid boundary. The advection and diffusion fluxes at E_{int} and N_{int} are given as

$$\begin{aligned} a^{n+1}_{\phi x,E,\text{int}} &= m^{n}_{x,E,\text{int}}\phi^{n+1}_{E,\text{int}},\ a^{n+1}_{\phi y,N,\text{int}} = m^{n}_{y,N,\text{int}}\phi^{n+1}_{N,\text{int}} \\ d_{\phi x,E,\text{int}} &= \mu_f \frac{\phi^{n+1}_{E,\text{int}} - \phi^{n+1}_{P}}{\delta x_{E,\text{int}}},\ d_{\phi y,N,\text{int}} = \mu_f \frac{\phi^{n+1}_{N,\text{int}} - \phi^{n+1}_{P}}{\delta y_{E,\text{int}}} \end{aligned} \tag{14.12}$$

where $\delta x_{E,\text{int}}$ and $\delta y_{N,\text{int}}$ are shown in Fig. 14.3b. The interface velocities ($\phi_{E,\text{int}}$ and $\phi_{N,\text{int}}$) and the mass fluxes at the east and north sides are obtained from the neighbouring solid grid points (at the interface) by linear interpolation, given as

$$\begin{aligned} \phi_{E,\text{int}} &= \overline{\phi_{N_2}, \phi_{N_3}},\ \phi_{N,\text{int}} = \overline{\phi_{N_1}, \phi_{N_2}} \\ m_{x,E,\text{int}} &= \rho_f u_{E,\text{int}},\ m_{y,N,\text{int}} = \rho_f v_{N,\text{int}} \end{aligned} \tag{14.13}$$

where N_1, N_2, and N_3 are solid nodes (defined for FEM), as shown in Fig. 14.3b. Similar linear interpolation is used to obtain $v_{E,\text{int}}$ and $v_{N,\text{int}}$, and the resulting $\overrightarrow{u}_{E,\text{int}}$ and $\overrightarrow{u}_{N,\text{int}}$ are used to obtain the advection fluxes [Eq. (14.12)] without using any advection scheme. The interface distances in the diffusion flux, $\delta x_{E,\text{int}}$ and $\delta y_{E,\text{int}}$ [Eq. (14.12) and Fig. 14.3b], are obtained using the level-set function ψ, given as

$$\delta x_{E,\text{int}} = \frac{x_E - x_P}{|\psi_E - \psi_P|}\,|\psi_P|\,,\ \delta y_{N,\text{int}} = \frac{y_N - y_P}{|\psi_N - \psi_P|}\,|\psi_P| \tag{14.14}$$

For the mass balance, the mass fluxes at east and north faces that corresponds to the interface values [Eq. (14.13)] are directly used; resulting mass balance equation, for the partially filled CV "P" (Fig. 14.3b), is given as

$$\begin{aligned} a_P p^{n+1}_P &= a_W p^{n+1}_W + a_S p^{n+1}_S + b \\ \text{where } a_P &= a_W + a_S,\ b = -S^{*}_{m,P} \\ &= -\left[\left(m^{n+1}_{x,E,\text{int}} - m^{*}_{x,w}\right)\triangle y_P + \left(m^{n+1}_{y,N,\text{int}} - m^{*}_{y,s}\right)\triangle x_P\right] \end{aligned} \tag{14.15}$$

The interface pressures ($P_{E,\text{int}}$ and $P_{N,\text{int}}$) are obtained from the pressure gradient boundary condition at the interface, given as

$$\frac{\partial p}{\partial n} = -\rho_f a_n \implies \nabla p.\hat{n} = -\rho_f \overrightarrow{a}\,.\hat{n} \tag{14.16}$$

where a_n is the normal acceleration at the interface. Using the level-set function at the interface, it is given (Shrivastava et al. 2013) as

$$\frac{\partial p}{\partial x}\frac{\partial \psi}{\partial x} + \frac{\partial p}{\partial y}\frac{\partial \psi}{\partial y} = -\rho_f \left(a_{x,\text{int}}\frac{\partial \psi}{\partial x} + a_{y,\text{int}}\frac{\partial \psi}{\partial y} \right) \tag{14.17}$$

where $a_{x,\text{int}}$ and $a_{y,\text{int}}$ are the accelerations in the x- and y-directions at the solid surface. Considering the solid boundary as horizontal and vertical lines, the above equation results in an approximated boundary condition for the pressure, given as

$$\frac{\partial p}{\partial x} \approx -\rho_f a_{x,\text{int}}, \quad \frac{\partial p}{\partial y} \approx -\rho_f a_{y,\text{int}} \tag{14.18}$$

Thus, the pressure boundary conditions at the east and north faces of the cell P are given as

$$\begin{aligned} p_{E,\text{int}} &= p_P - \delta x_{E,\text{int}} \rho_f a_{x,E,\text{int}} \\ p_{N,\text{int}} &= p_P - \delta y_{N,\text{int}} \rho_f a_{y,N,\text{int}} \end{aligned} \tag{14.19}$$

where $a_{x,E,\text{int}}$ and $a_{y,N,\text{int}}$ are obtained by linear interpolation from the solid grid points [similar to Eq. (14.13)].

Calculation of Level-Set Function: A Geometric Method

Level-set function ψ (Sethian 1999) is a normal distance function with a change in sign across an interface. The sign of ψ is used to detect the Cartesian CVs that are in the fluid and also the partially filled CVs. Furthermore, its magnitude is used to calculate the diffusion fluxes [Eq. (14.14)] in the partially filled CVs. The sign of the level-set function is distinguished by using a winding number algorithm, and its magnitude is obtained by a minimum distance algorithm, proposed in our recent work (Thekkethil and Sharma 2019). The algorithms are presented here for a 2D solid body; however, it can be extended to 3D geometries also.

The surface of the structure Γ_s is divided into n_{ss} number of line segments Γ_i, as shown in Fig. 14.4, where $i = 1, 2, \ldots, n_{ss}$. For the endpoints in the line segments, the position vectors are specified as $\left[\overrightarrow{x}_1, \overrightarrow{x}_2, \ldots, \overrightarrow{x}_{n_{ss}}\right]$. For any cell P in the Cartesian domain (Fig. 14.4), the sign of the level-set function is computed using the winding number algorithm, which is better than many other methods in this category such as line tracing algorithm, that fails if the shape of the structure is complex. In the winding number algorithm, the counterclockwise angle subtended by each line segment with the cell P is added to obtain the winding number. For the cell P with position vector $\overrightarrow{x}$ in Fig. 14.4, the angle subtended by the line segment Γ_i is given as

$$\theta_i = \cos^{-1}\left[\frac{\left(\overrightarrow{x}_i - \overrightarrow{x}\right).\left(\overrightarrow{x}_{i+1} - \overrightarrow{x}\right)}{\left|\overrightarrow{x}_i - \overrightarrow{x}\right|\left|\overrightarrow{x}_{i+1} - \overrightarrow{x}\right|} \right] \tag{14.20}$$

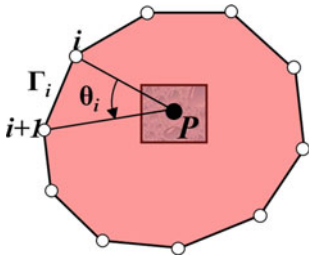


Fig. 14.4 For a fluid cell in the Cartesian system, computation of angle between two consecutive grid points on the structure surface

The winding number ω_n for the cell P is computed as

$$\omega_{n,P} = \frac{1}{2\pi} \sum_{i=1}^{n_{ss}} \theta_i \tag{14.21}$$

For any cell P inside the structure, $\omega_{n,P}$ will be 1 while it will be 0 for a cell P in the fluid.

For finding the magnitude of the level-set function, the minimum distance algorithm is used. For each Cartesian cell P, the shortest distance from each line segment Γ_i (Fig. 14.4) is calculated as

$$\begin{aligned} |\psi|_{P,i} &= \left|\vec{x}_d\right| \\ \text{where } \vec{x}_d &= \begin{cases} \vec{x} - \vec{x}_i & \text{if } t < 0 \\ \vec{x} - \vec{x}_{i+1} & \text{if } t > 1 \\ \vec{x} - \left(\vec{x}_i + t\left[\vec{x}_{i+1} - \vec{x}_i\right]\right) & \text{if } 0 \le t \le 1 \end{cases} \\ t &= \frac{\left(\vec{x} - \vec{x}_i\right).\left(\vec{x}_{i+1} - \vec{x}_i\right)}{\left|\vec{x}_{i+1} - \vec{x}_i\right|^2} \end{aligned} \tag{14.22}$$

The magnitude of the level-set function is computed as the minimum of the shortest distance from each line segment, given as

$$|\psi|_P = \min\left(|\psi|_{P,1}, |\psi|_{P,2}, \ldots, |\psi|_{P,n_{ss}}\right) \tag{14.23}$$

For a 3D geometry, a 3D winding algorithm (Jacobson et al. 2013) can be used to find whether the point lies inside or outside the structure. The 3D geometry surface can be divided into a finite number of elements. For each Cartesian cell, similar to θ_i in 2D, the solid angle can be calculated for each element on the surface. Summation of the solid angles with all the elements gives a measure of the winding number. For finding the magnitude of the level-set function, similar to $|\psi|_{P,i}$ in 2D, the shortest distance from each element can be computed, and the minimum of the shortest distance gives the magnitude of the level-set function.

Solution algorithm:

For the present HLE method, the CFD solution is obtained by solving the equations presented in Sect. 14.3.1.3, for the completely as well as partially filled CVs. The solution algorithm to obtain the velocity and pressure at a new $(n+1)$th time step from the old nth time instant is given as follows:

1. Initialise the velocity, pressure, and level-set field in the domain as per the initial configuration variables.
2. Compute the level-set function ψ using Eqs. (14.21) and (14.23).
3. Solve Eq. (14.8) for u^* and v^*.
4. Predict $S^*_{m,P}$ [Eq. (14.11)] required for the pressure equation.
5. Solve the pressure equation, Eq. (14.11).
6. Calculate the corrected mass flux m_f^{n+1} [Eq. (14.10)].
7. Solve Eq. (14.7) for u_P^{n+1} and v_P^{n+1} using mass conserving mass flux $m_{f^{n+1}}$ and linearly interpolated pressure ($p_w = \overline{p_P, p_W}$ and $p_e = \overline{p_P, p_E}$).
8. Set $n = n + 1$ and repeat steps $2 - 8$ for the next time step. Continue up to certain stopping criterion of the transient simulation.

14.3.2 CSD Development: Geometric Nonlinear Galerkin FEM-Based Numerical Methodology for Structural Dynamics

For the structural dynamics involving large deformation, a geometric nonlinear Galerkin FEM-based algebraic formulation is used here to convert the momentum conservation equation [Eq. (14.3)] to a system of linear algebraic equations for displacement vector $\vec{\mathrm{d}}$. Similar to the CFD development in the previous section, CSD development is presented below in separate subsections for grid generation, FEM-based algebraic formulation, and solution methodology.

14.3.2.1 Unstructured Grid Generation

For the development of a CSD solver, the present geometric nonlinear Galerkin FEM-based numerical methodology considers a fixed body-fitted unstructured grid, as shown in Fig. 14.2. The figure shows that the grid generation involves dividing the solid into several control masses, considered triangular here. The control masses are called as *elements,* and the grid points at the vertices of the elements are called as *nodes* in FEM, represented by unfilled circles in Fig. 14.2. Although the triangular elements, along with the nodes, move substantially during large deformation, a fixed node/element is considered in the geometric nonlinear Galerkin FEM presented in the next subsection. This involves defining the deformation vector $\vec{\mathrm{d}}$ at the various solid nodes, with reference to the initial ($t = 0$) node configuration, shown in Fig. 14.2.

14.3.2.2 Geometric Nonlinear Galerkin Finite Element Method

In order to solve the Lagrangian form of the momentum conservation law [Eq. (14.5)] for the deformation vector $\overrightarrow{\mathrm{d}}$ at the various nodes (Fig. 14.2), the algebraic formulation for a three-node triangular element is presented here. For the volume Ω^e and surface Γ^e of the element, the instantaneous momentum conservation [Eq. (14.5)] for the element e is given as

$$\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\int_{\Omega^e}\rho_s\,\overrightarrow{\mathrm{d}}^{\,e}\mathrm{d}\Omega^e\right)^{n+1} = \int_{\Gamma_c^e}\sigma_c^{n+1}.\hat{n}\mathrm{d}\Gamma_c^e = \int_{\Gamma_{\mathrm{in}}^e}\left(\mathbb{D}^{e,n+1}\mathrm{S}^{e,n+1}\right).\hat{n}\mathrm{d}\Gamma_{in}^e \tag{14.24}$$

where the L.H.S of the above equation corresponds to the unsteady term $\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathcal{M}\overrightarrow{u}\right)$ and R.H.S to the surface force term $\overrightarrow{\mathrm{F}}_s^e$. Also, note that the above equation for the force is first represented with reference to the current (instantaneous) configuration $\overrightarrow{\mathrm{F}}_{s,c}^e$ and then with reference to the initial ($t = 0$) configuration $\overrightarrow{\mathrm{F}}_{s,\mathrm{in}}^e$. Here, ρ_s is the density of the solid and $\overrightarrow{\mathrm{d}}^{\,e}$ is the displacement vector of the element. $\overrightarrow{\mathrm{F}}_{s,c}^e$ is presented above as the surface integral of the Cauchy stress stress σ that is with reference to the deformed or current configuration. Its conversion with reference to the initial configuration (with surface area Γ_{in}^e) results in a product of $\mathbb{D}^e$ and S^e and corresponds to *deformation gradient* and *second Piola–Kirchhoff's stress*, respectively. They are given in 2D Cartesian coordinate system as

$$\mathbb{D}^e = I + \left(\nabla\overrightarrow{\mathrm{d}}^{\,e}\right)^T = \begin{bmatrix} 1+d_{x,x}^e & d_{x,y}^e \\ d_{y,x}^e & 1+d_{y,y}^e \end{bmatrix} \text{ and } \mathrm{S}^e = \begin{bmatrix} S_{xx}^e & S_{xy}^e \\ S_{yx}^e & S_{yy}^e \end{bmatrix} \tag{14.25}$$

Here, the components of S^e are presented in a matrix form as $\mathbf{S^e} = \mathrm{D}\mathbf{E}^e$, where $\mathbf{S^e}$ is the *element stress matrix,* D is the *stress–strain relationship matrix*, and $\mathbf{E}^e$ is the element *Green strain matrix*. Using St. Venant–Kirchhoff's model for a plain-strain case, the matrix form of stress–strain relationship is given as

$$\mathbf{S^e} = \mathrm{D}\mathbf{E}^e \Rightarrow \begin{bmatrix} S_{xx}^e \\ S_{yy}^e \\ S_{xy}^e \end{bmatrix} = \begin{bmatrix} C_1 & C_2 & 0 \\ C_2 & C_1 & 0 \\ 0 & 0 & C_3 \end{bmatrix} \begin{bmatrix} d_{x,x}^e + 0.5\left[\left(d_{x,x}^e\right)^2 + \left(d_{y,x}^e\right)^2\right] \\ d_{y,y}^e + 0.5\left[\left(d_{x,y}^e\right)^2 + \left(d_{y,y}^e\right)^2\right] \\ d_{x,y}^e\left(1+d_{x,x}^e\right) + d_{y,x}^e\left(1+d_{y,y}^e\right) \end{bmatrix} \tag{14.26}$$

where $C_1 = {}^{E(1-\nu_s)}/_{(1+\nu_s)(1-2\nu_s)}$, $C_2 = {}^{E\nu_s}/_{(1+\nu_s)(1-2\nu_s)}$, and $C_3 = {}^{E}/_{2(1+\nu_s)}$. Here, E is Young's modulus and ν_s is Poisson's ratio of the solid material. Further, the suffix after the comma for d^e above represents the derivative, i.e. $d_{x,x}^e = \frac{d}{dx}d_x^e$.

Using the Gauss divergence theorem, Eq. (14.24) is given as

$$\rho_{s,\text{in}} \left(\frac{\mathrm{d}^2}{\mathrm{d}t^2} \int\limits_{\Omega^e_{in}} \overrightarrow{\mathrm{d}}^{\,e} d\Omega^e_{\text{in}} \right)^{n+1} = \int\limits_{\Omega^e_{\text{in}}} \nabla . \left(\mathbb{D}^{e,n+1} \mathrm{S}^{e,n+1} \right) d\Omega^e_{\text{in}} \tag{14.27}$$

Using a bilinear interpolation with $\overrightarrow{\mathrm{d}}^{\,e} = a\overrightarrow{x} + b\overrightarrow{y} + c$ for the element e, the constants a, b, and c are obtained as $f\left(\overrightarrow{x}_1, \overrightarrow{x}_2, \overrightarrow{x}_3, \overrightarrow{\mathrm{d}}^{\,e}_1, \overrightarrow{\mathrm{d}}^{\,e}_2, \overrightarrow{\mathrm{d}}^{\,e}_3\right)$, which after certain rearrangements results in a function form of the displacement vector for the element e as

$$\overrightarrow{\mathrm{d}}^{\,e} = N^e_1 \overrightarrow{\mathrm{d}}^{\,e}_1 + N^e_2 \overrightarrow{\mathrm{d}}^{\,e}_2 + N^e_3 \overrightarrow{\mathrm{d}}^{\,e}_3$$

$$\text{where, } \begin{bmatrix} N^e_1 \\ N^e_2 \\ N^e_3 \end{bmatrix} = \frac{1}{\begin{vmatrix} 1 & x^e_1 & y^e_1 \\ 1 & x^e_2 & y^e_2 \\ 1 & x^e_3 & y^e_3 \end{vmatrix}} \begin{bmatrix} x^e_2 y^e_3 - x^e_3 y^e_2 + \left(y^e_2 - y^e_3\right) x + \left(x^e_3 - x^e_2\right) y \\ x^e_3 y^e_1 - x^e_1 y^e_3 + \left(y^e_3 - y^e_1\right) x + \left(x^e_1 - x^e_3\right) y \\ x^e_1 y^e_2 - x^e_2 y^e_1 + \left(y^e_1 - y^e_2\right) x + \left(x^e_2 - x^e_1\right) y \end{bmatrix} \tag{14.28}$$

Here, N^e_1, N^e_2, and N^e_3 are called as *shape functions* of the element e with respect to the nodes 1, 2, and 3, respectively (Fig. 14.5). Substituting $\overrightarrow{\mathrm{d}}^{\,e}$ from Eqs. (14.28) to (14.27), we get

$$\rho_{s,\text{in}} \left[\frac{\mathrm{d}^2}{\mathrm{d}t^2} \int\limits_{\Omega^e_{\text{in}}} \left(N^e_1 \overrightarrow{\mathrm{d}}^{\,e}_1 + N^e_2 \overrightarrow{\mathrm{d}}^{\,e}_2 + N^e_3 \overrightarrow{\mathrm{d}}^{\,e}_3 \right) \mathrm{d}\Omega^e_{\text{in}} \right]^{n+1} = \left[\int\limits_{\Omega^e_{\text{in}}} \nabla . \left(\mathbb{D}^e \mathrm{S}^e \right) \mathrm{d}\Omega^e_{\text{in}} \right]^{n+1} \tag{14.29}$$

Nodal Equations:

For conversion of the above element equation to a nodal equation, the geometric nonlinear Galerkin FEM (as compared to other FEMs) involves multiplication of the

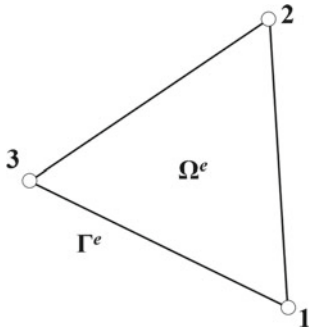


Fig. 14.5 A triangular element with three nodes considered for the geometric nonlinear Galerkin FEM-based algebraic formulation for an element "e"

integrand of the above equation with the shape functions of the corresponding nodes. Using this operation, Eq. (14.29) results in a nodal for node i as

$$\rho_{s,\text{in}}\left[\frac{\mathrm{d}^2}{\mathrm{d}t^2}\int_{\Omega^e_{\text{in}}}\left(N_1^e\overrightarrow{\mathrm{d}}_1^e+N_2^e\overrightarrow{\mathrm{d}}_2^e+N_3^e\overrightarrow{\mathrm{d}}_3^e\right)N_i^e\mathrm{d}\Omega^e_{\text{in}}\right]^{n+1}$$
$$=\left[\int_{\Gamma^e_{\text{in}}}\left(\mathbb{D}^e\mathrm{S}^e\right)N_i^e.\hat{n}\mathrm{d}\Gamma^e_{\text{in}}\right]^{n+1}-\left[\int_{\Omega^e_{\text{in}}}\left(\mathbb{D}^e\mathrm{S}^e\right).\nabla\left(N_i^e\right)\mathrm{d}\Omega^e_{\text{in}}\right]^{n+1} \tag{14.30}$$

where the first surface integral term (obtained after applying the Gauss divergence theorem) on the R.H.S above represents the surface force acting on node i, $\overrightarrow{F}_i^e$. Furthermore, substituting $\mathbb{D}^e$ and S^e from Eq. (14.25), the integrand of the second term in the R.H.S of the above equation is presented in matrix form. They are given as

$$\left[\int_{\Gamma^e_{\text{in}}}\left(\mathbb{D}^e\mathrm{S}^e\right)N_i^e.\hat{n}\mathrm{d}\Gamma^e_{\text{in}}\right]^{n+1}=\begin{bmatrix}F^e_{xi}\\F^e_{yi}\end{bmatrix},$$
$$\left(\mathbb{D}^e\mathrm{S}^e\right).\nabla\left(N_i^e\right)=\begin{bmatrix}1+d^e_{x,x} & d^e_{x,y}\\ d^e_{y,x} & 1+d^e_{y,y}\end{bmatrix}\begin{bmatrix}S^e_{xx} & S^e_{xy}\\ S^e_{yx} & S^e_{yy}\end{bmatrix}\begin{bmatrix}N^e_{i,x}\\N^e_{i,y}\end{bmatrix}$$
$$=\left(\left[\mathbf{B}_i^{L,e}\right]^T+\left[\mathbf{B}_i^{NL,e}\right]^T\right)\mathbf{S}^e \tag{14.31}$$

where $\mathbf{B}_i^{L,e}$ and $\mathbf{B}_i^{NL,e}$ are the *linear* and *nonlinear deformation matrices* for node i, given as

$$\mathbf{B}_i^{L,e}=\begin{bmatrix}N^e_{i,x} & 0\\ 0 & N^e_{i,y}\\ N^e_{i,y} & N^e_{i,x}\end{bmatrix},\ \mathbf{B}_i^{NL,e}=\begin{bmatrix}N^e_{i,x}d^e_{x,x} & N^e_{i,x}d^e_{y,x}\\ N^e_{i,y}d^e_{x,y} & N^e_{i,y}d^e_{y,y}\\ N^e_{i,x}d^e_{x,y}+N^e_{i,y}d^e_{x,x} & N^e_{i,x}d^e_{y,y}+N^e_{i,y}d^e_{y,x}\end{bmatrix} \tag{14.32}$$

Here, the suffix after the comma for N_i^e above represents the derivative, i.e. $N^e_{i,x}=\frac{d}{dx}N_i^e$. Further, the element stress matrix $\mathbf{S}^e$ in Eq. (14.31) is represented in terms of $\mathbf{B}^{L,e}$ and *modified deformation matrix* $\mathbf{Bm}^e$, by substituting Eq. (14.28) for $\overrightarrow{\mathrm{d}}^e$ into Eq. (14.26) for $\mathbf{S}^e$, given as

$$\mathbf{S}^e=\mathrm{D}\left(\mathbf{B}^{\mathrm{L,e}}+\mathbf{Bm}^{\mathrm{e}}\right)\mathbf{d}^e$$
$$\text{where }\mathbf{B}^{L,e}=\left[\mathbf{B}_1^{L,e}\ \mathbf{B}_2^{L,e}\ \mathbf{B}_3^{L,e}\right]\text{ and }\mathbf{Bm}^e=\left[\mathbf{Bm}_1^e\ \mathbf{Bm}_2^e\ \mathbf{Bm}_3^e\right] \tag{14.33}$$

where $\mathbf{Bm}_i^e$ and $\mathbf{d}^e$ (nodal displacement vector of the element e) are given as

$$\mathbf{Bm}_i^e = 0.5 \begin{bmatrix} N_{i,x}^e d_{x,x}^e & N_{i,x}^e d_{y,x}^e \\ N_{i,y}^e d_{x,y}^e & N_{i,y}^e d_{y,y}^e \\ 2N_{i,y}^e d_{x,x}^e & 2N_{i,x}^e d_{y,y}^e \end{bmatrix}, \mathbf{d}^e = \begin{bmatrix} d_{x1}^e \\ d_{y1}^e \\ d_{x2}^e \\ d_{y2}^e \\ d_{x3}^e \\ d_{y3}^e \end{bmatrix} \tag{14.34}$$

Substituting Eqs. (14.31)–(14.30), taking all the terms to the R.H.S, and using *generalised Newmark algorithm* (Zienkiewicz et al. 1977) [with a second-degree polynomial approximation for time variation and second-order accuracy (GN22)], the *residual vector* $\mathbf{\Psi}$ of an element e with respect to node i for $(n+1)$th time instant is given as

$$\mathbf{\Psi}_i^{e,n+1} = \begin{bmatrix} F_{xi}^e \\ F_{yi}^e \end{bmatrix} - \left[\int_{\Omega_{\text{in}}^e} \left(\left[\mathbf{B}_i^{L,e} \right]^T + \left[\mathbf{B}_i^{NL,e} \right]^T \right) \mathbf{S}^e d\Omega_{\text{in}}^e \right]^{n+1} - \frac{2\rho_{s,\text{in}}}{\triangle t^2} \int_{\Omega_{\text{in}}^e} \begin{bmatrix} N_1^e & 0 & N_2^e & 0 & N_3^e & 0 \\ 0 & N_1^e & 0 & N_2^e & 0 & N_3^e \end{bmatrix} N_i^e d\Omega_{\text{in}}^e \left[\mathbf{d}^{e,n+1} - \mathbf{d}^{e,n} + \triangle t \times \mathbf{u}^{e,n} \right] = 0 \tag{14.35}$$

where $\mathbf{u}^{e,n}$ is the nodal velocity vector of the element e at nth time instant. Since the above equation is nonlinear, an iterative method is used for the solution. Using the *Newton–Raphson method*, the residual vector at $(k+1)$th iterative step is obtained by a Taylor series expansion, given as

$$\mathbf{\Psi}_i^{e,n+1,k+1} \approx \mathbf{\Psi}_i^{e,n+1,k} + \frac{\partial \mathbf{\Psi}_i^{e,n+1,k}}{\partial \mathbf{d}^{e,n+1}} d\mathbf{d}^{e,n} = 0 \tag{14.36}$$

where $d\mathbf{d}^{e,n}$ is the increment to the displacement vector, given as

$$d\mathbf{d}^{e,n} = \mathbf{d}^{e,n+1,k+1} - \mathbf{d}^{e,n+1,k} \tag{14.37}$$

Thus, the final equation for the three nodes of a triangular element—called as *nodal equation*—is given as

$$\mathbf{K}_i^e d\mathbf{d}^{e,n} = \mathbf{\Psi}_i^{e,n+1k} \text{ where } \mathbf{K}_i^e = -\frac{\partial \mathbf{\Psi}_i^{e,n+1,k}}{\partial \mathbf{d}^{e,n+1}} \text{ and } i = 1, 2, 3 \tag{14.38}$$

Here, $\mathbf{K}_i^e$ is the *element stiffness matrix* for the element e with respect to node i and the associated derivative of $\mathbf{\Psi}_i^{e,n+1,k}$ [Eq. (14.35)] with respect to $\mathbf{d}^{e,n+1}$, given as

$$\mathbf{K}_i^e = 0 + \left(\mathbf{K}_{m,i}^e + \mathbf{G}_i^e \right) + \frac{2\mathbf{M}_i^e}{\triangle t^2} \tag{14.39}$$

where the derivative of Eq. (14.35) with respect to $\mathbf{d}^{e,n+1}$ for the surface force $\overrightarrow{F}_i^e$ is zero, differentiation in parts for the stress term results in the terms shown above inside the bracket, and that for the unsteady term results in $2\mathbf{M}_i^e/\triangle t^2$. $\mathbf{K}_{m,i}^e$, $\mathbf{G}_i^e$, and $\mathbf{M}_i^e$ in the above equation are called as *material tangent matrix*, *geometric stiffness matrix*, and *mass matrix*, respectively. The material tangent matrix is given as

$$\mathbf{K}_{m,i}^e = \int_{\Omega_{\text{in}}^e} \left(\left[\mathbf{B}_i^{L,e} \right]^T + \left[\mathbf{B}_i^{NL,e} \right]^T \right)^{n+1} \frac{d}{d\mathbf{d}^{e,n+1}} \mathbf{S}^{e,n+1} d\Omega_{\text{in}}^e \tag{14.40}$$

Substituting $\mathbf{S}^e$ from Eq. (14.33) and using $\frac{d}{d\mathbf{d}^{e,n+1}} \left(\left[\mathbf{B}_i^{L,e} \right]^T \right) = 0$, we get

$$\begin{aligned} \frac{d}{d\mathbf{d}^{e,n+1}} \mathbf{S}^{e,n+1} &= \mathrm{D}\mathbf{B}^{e,n+1} \text{ where } \mathbf{B}^e = \mathbf{B}^{L,e} + \mathbf{B}^{NL,e} \\ \implies \mathbf{K}_{m,i}^e &= \left(\left[\mathbf{B}_i^e \right]^T \right)^{n+1} \mathrm{D}\mathbf{B}^{e,n+1} \Omega_{\text{in}}^e \end{aligned} \tag{14.41}$$

Further, using $\frac{d}{d\mathbf{d}^{e,n+1}} \left(\left[\mathbf{B}_i^{L,e} \right]^T \right) = 0$, the geometric stiffness matrix $\mathbf{G}_i^e$ is given as

$$\begin{aligned} \mathbf{G}_i^e &= \int_{\Omega_{\text{in}}^e} \frac{d}{d\mathbf{d}^{e,n+1}} \left(\left[\mathbf{B}_i^{L,e} \right]^T + \left[\mathbf{B}_i^{NL,e} \right]^T \right)^{n+1} \mathbf{S}^{e,n+1} d\Omega_{\text{in}}^e \\ \Rightarrow \mathbf{G}_i^e &= \begin{bmatrix} G_{i,1}^e & 0 & G_{i,2}^e & 0 & G_{i,3}^e & 0 \\ 0 & G_{i,1}^e & 0 & G_{i,2}^e & 0 & G_{i,3}^e \end{bmatrix} \Omega_{\text{in}}^e \text{ where} \\ G_{i,j}^e &= N_{i,x}^e S_{xx}^{e,n+1} N_{j,x}^e + N_{i,x}^e S_{xy}^{e,n+1} N_{j,y}^e + N_{i,y}^e S_{xy}^{e,n+1} N_{j,x}^e + N_{i,y}^e S_{yy}^{e,n+1} N_{j,y}^e \end{aligned} \tag{14.42}$$

The mass matrix $\mathbf{M}_i^e$ is given as

$$\mathbf{M}_i^e = \rho_{s,\text{in}} \int_{\Omega_{\text{in}}^e} \begin{bmatrix} N_1^e & 0 & N_2^e & 0 & N_3^e & 0 \\ 0 & N_1^e & 0 & N_2^e & 0 & N_3^e \end{bmatrix} N_1^e d\Omega_{\text{in}}^e \tag{14.43}$$

Element equations:

Combining the nodal equations for all the three nodes [Eq. (14.38)], the system of equations for an element e is presented in matrix form as

$$\mathbf{K}^e d\mathbf{d}^{e,n,k} = \mathbf{\Psi}^{e,n+1,k} \text{ where } \mathbf{K}^e = \begin{bmatrix} \mathbf{K}_1^e \\ \mathbf{K}_2^e \\ \mathbf{K}_3^e \end{bmatrix} \text{ and } \mathbf{\Psi}^e = \begin{bmatrix} \mathbf{\Psi}_1^e \\ \mathbf{\Psi}_2^e \\ \mathbf{\Psi}_3^e \end{bmatrix} \tag{14.44}$$

Here, $\mathbf{K}^e$ is the *element stiffness matrix* and $\mathbf{\Psi}^{e,n+1,k}$ is the residual of the element e for $(n+1)$th time step, given as

$$\boldsymbol{\Psi}^{e,n+1,k} = \begin{bmatrix} F^e_{x1} \\ F^e_{y1} \\ F^e_{x2} \\ F^e_{y2} \\ F^e_{x3} \\ F^e_{y3} \end{bmatrix} - \left(\left[\mathbf{B}^e\right]^T\right)^{n+1} \mathbf{S}^{e,n+1}\Omega^e_{\text{in}} - \frac{2\mathbf{M}^e}{\triangle t^2}\left[\mathbf{d}^{e,n+1} - \mathbf{d}^{e,n} + \triangle t \times \mathbf{u}^{e,n}\right],$$

$$\mathbf{K}^e = \mathbf{K}^e_m + \mathbf{G}^e + \frac{2\mathbf{M}^e}{\triangle t^2} \tag{14.45}$$

For the element e, the various matrices in the above equation are given as

$$\mathbf{K}^e_m = \left(\left[\mathbf{B}^e\right]^T\right)^{n+1} \mathrm{D}\left[\mathbf{B}^e\right]^{n+1} \Omega^e_{\text{in}},$$

$$\mathbf{G}^e = \Omega^e_{\text{in}} \begin{bmatrix} G^e_{1,1} & 0 & G^e_{1,2} & 0 & G^e_{1,3} & 0 \\ 0 & G^e_{1,1} & 0 & G^e_{1,2} & 0 & G^e_{1,3} \\ G^e_{2,1} & 0 & G^e_{2,2} & 0 & G^e_{2,3} & 0 \\ 0 & G^e_{2,1} & 0 & G^e_{2,2} & 0 & G^e_{2,3} \\ G^e_{3,1} & 0 & G^e_{3,2} & 0 & G^e_{3,3} & 0 \\ 0 & G^e_{3,1} & 0 & G^e_{3,2} & 0 & G^e_{3,3} \end{bmatrix}, \mathbf{M}^e = \frac{\rho_{s,\text{in}}\Omega^e_{\text{in}}}{12} \begin{bmatrix} 2&0&1&0&1&0 \\ 0&2&0&1&0&1 \\ 1&0&2&0&1&0 \\ 0&1&0&2&0&1 \\ 1&0&1&0&2&0 \\ 0&1&0&1&0&2 \end{bmatrix} \tag{14.46}$$

14.3.2.3 Solution Methodology: Global Equations

For CSD development, solution methodology corresponds to solution of a global system of linear algebraic equations that is obtained by a summation of the element equation [Eq. (14.44) for n_e number of elements], given as

$$\sum_{e=1}^{n_e} \mathbf{K}^e d\mathbf{d}^{e,n,k} = \sum_{e=1}^{n_e} \boldsymbol{\Psi}^{e,n+1,k} \Rightarrow \mathbf{K}^G d\mathbf{d}^{G,n,k} = \boldsymbol{\Psi}^{G,n+1,k} \tag{14.47}$$

where $\mathbf{K}^G$ is the *global stiffness matrix*, $d\mathbf{d}^G$ is the *global displacement-increment vector*, and $\boldsymbol{\Psi}^G$ is the *global residual vector*. The global $\mathbf{K}^G$, $d\mathbf{d}^G$, and $\boldsymbol{\Psi}^G$ are expressed in terms of the respective elemental equations, given as

$$K^G_{2i+p-2,2j+q-2} = \sum_{e=1}^{n_e} K^e_{2l+p-2,2m+q-2}, \quad \text{if } R^{e,l} = i;\, R^{e,m} = j;$$
$$i = 1, 2, \ldots, n_s; j = 1, 2, \ldots, n_s$$
$$\text{for} \quad l = 1, 2, 3; m = 1, 2, 3 \tag{14.48}$$
$$p = 1, 2; q = 1, 2$$

$$d\mathrm{d}^{G,n}_{2i+p-2} = d\mathrm{d}^{e,n}_{2l+p-2}, \text{ if } i = R_{e,l} \text{ , for } \begin{matrix} e = 1, 2, \ldots, n_e \\ i = 1, 2, \ldots, n_s \\ l = 1, 2, 3 \\ p = 1, 2 \end{matrix} \tag{14.49}$$

$$\Psi^{G,n+1,k}_{2i+p-2} = \mathrm{f}^{G}_{2i+p-2} - \sum_{e=1}^{n_e} \Psi^{e,n+1,k}_{2l+p-2} \text{ if } i = R^{e,l} \text{ , for } \begin{matrix} i = 1, 2, \ldots, n_s \\ l = 1, 2, 3 \\ p = 1, 2 \end{matrix} \tag{14.50}$$

where f^{G}_{2i+p-2} represents the external forces acting on the node i (in x-direction for $p = 1$ and y-direction for $p = 2$), i.e. $\mathrm{f}^{G}_{2i+p-2} = \sum_{e=1}^{n_e} f^{e}_{2l+p-2}$. Here, $\mathbf{K}^G$, $d\mathbf{d}^G$, and $\mathbf{\Psi}^G$ are computed with respect to the global node numbering from 1 to n_s. In order to relate global node numbering with the element node numbering, a node relationship matrix is defined as

$$R^{e,l} = 3e - 3 + l \text{ where } e = 1, 2, \ldots, n_e; l = 1, 2, 3 \tag{14.51}$$

The global displacement vector $\mathbf{d}^G$ is also related to the element displacement vector $\mathbf{d}^e$ in the same way. For the present iterative step, $\mathbf{d}^{G,n+1,k+1}$ is obtained as

$$\mathbf{d}^{G,n+1,k+1} = \mathbf{d}^{G,n+1,k} + d\mathbf{d}^{G,n,k} \tag{14.52}$$

Solution algorithm:

1. Generate unstructured mesh, resulting in the position vector of all the nodes and global as well as local node numbering.
2. Compute node relationship matrix $R^{e,l}$ [Eq. (14.51)].
3. Initialise $\mathbf{d}^G = 0$ and compute the geometric parameters.
4. Assume $\mathbf{d}^{G,n+1,k} = \mathbf{d}^{G,n}$ and set $k = 1$.
5. Update the displacement matrix $\mathbf{d}^{G,n+1,k+1} = \mathbf{d}^{G,n+1,k}$ and set $k = k + 1$.
6. Compute global stiffness matrix $\mathbf{K}^G$ [Eq. (14.48)] and global residual vector $\mathbf{\Psi}^{G,n+1,k}$ [Eq. (14.50)].
7. Solve Eq. (14.47) to obtain $d\mathbf{d}^{G,n,k}$ and update $\mathbf{d}^{G,n+1,k+1}$ using Eq. (14.52).
8. Check for the convergence. If $\left|\Psi^{n+1,k+1}\right| < \varepsilon \left|\Psi^{n+1,1}\right|$, continue to next time step and go to step 2, else set $k = k + 1$ and go to step 2.

14.3.3 Implicit Coupling Between CFD and CSD Solvers

The fluid dynamics and structural dynamics co-occur in an FSI problem. Thus, both CFD solver and CSD solver are coupled. The coupling is achieved by using the continuity condition presented in Sect. 14.2.3, which can be either explicit or implicit. Explicit coupling leads to a time lag between the fluid and structural solver, while

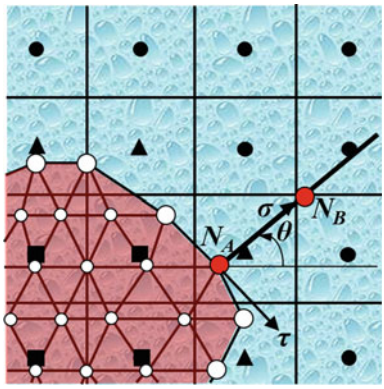


Fig. 14.6 Normal and shear stress on a node N_A on the solid surface

an iterative procedure is used in the implicit coupling to remove the time lag. The implicit coupling has better numerical stability characteristics and is essential for large deformation; thus, the implicit coupling is used in the present HLE method (Thekkethil and Sharma 2019) and presented below.

For the fluid domain, the body velocity obtained from the CSD solver is used as the boundary condition for the two-way coupled CFSD problem. For the CSD solver, the fluid dynamic forces obtained from the CFD solver are used as the boundary condition on the surface of the solid, for the two-way coupled CFSD problem. Figure 14.6 shows the fluid dynamics forces acting on a node N_A on the solid surface. The normal and shear stresses acting on the node are given as follows:

$$\sigma = \left[-p + \mu_f \frac{\partial u_n}{\partial n}\right]_{N_A}; \tau = \left[\mu_f \frac{\partial u_\tau}{\partial n}\right]_{N_A} \tag{14.53}$$

Here, the pressure p at node N_A is computed using quadratic interpolation from the neighbouring nodes. The normal derivatives of normal and tangential velocities at node N_A are computed as

$$\left[\frac{\partial u_n}{\partial n}\right]_{N_A} = \frac{u_{n,N_B} - u_{n,N_A}}{\delta}; \left[\frac{\partial u_\tau}{\partial n}\right]_{N_B} = \frac{u_{\tau,N_B} - u_{\tau,N_A}}{\delta} \tag{14.54}$$

where N_B is a point along the normal at the node N_A at a distance δ, which is equal to the finest grid size considered in the fluid domain. The normal and tangential velocities at N_B, u_{n,N_B}, and u_{τ,N_B} are computed using quadratic interpolation from the nearest fluid cells. From the normal and tangential stresses, the stresses along x- and y-directions are obtained as

$$\sigma_x = \sigma \cos\theta + \tau \sin\theta; \sigma_y = -\sigma \sin\theta + \tau \cos\theta \tag{14.55}$$

where θ is the angle of the normal at node N_A, given as

$$\theta = \tan^{-1}\left[\frac{\left(\frac{\partial \psi}{\partial y}\right)}{\left(\frac{\partial \psi}{\partial x}\right)}\right] \tag{14.56}$$

For the implicit coupling between the fluid and structural solvers, the solution is obtained iteratively until a convergence criterion is achieved for the interface variables, i.e. x-position x_{int}, y-position y_{int}, x-velocity u_{int}, y-velocity v_{int}, x-acceleration $a_{x,\text{int}}$, and y-acceleration $a_{y,\text{int}}$. The convergence criteria correspond to maximum of residual that is given as

$$\mathbf{R} = \max\left(R_{x_{\text{int}}}^{n+1,\text{new}}, R_{y_{\text{int}}}^{n+1,\text{new}}, R_{u_{\text{int}}}^{n+1,\text{new}}, R_{v_{\text{int}}}^{n+1,\text{new}}, R_{a_{x,\text{int}}}^{n+1,\text{new}}, R_{a_{y,\text{int}}}^{n+1,\text{new}}\right) < \varepsilon \tag{14.57}$$

Here, $R_{\chi_{\text{int}}}^{n+1,\text{new}}$ is the root mean square of the residuals of all the interface nodes for the present iteration, given as

$$R_{\chi_{\text{int}}}^{n+1,\text{new}} = \sqrt{\frac{1}{n_s}\sum_{i=1}^{n_{s,\text{int}}}\left(r_{\chi_{int,i}}^{n+1,\text{new}}\right)^2}, \text{ where } r_{\chi_{int,i}}^{n+1,\text{new}} = \chi_{int,i}^{n+1,\text{new}} - \chi_{int,i}^{n+1,\text{old}} \tag{14.58}$$

Here, the superscripts represent the new and old iterations. For each iteration, the interface variables are updated using an under-relaxation factor to ensure convergence. For faster convergence, *Aitken's acceleration method* (Degroote et al. 2010) is used for the under-relaxation factor after certain (three here) iterative steps, given as

$$\omega_{\chi_{\text{int}}}^{n+1,\text{new}} = \omega_{\chi_{\text{int}}}^{n+1,\text{old}} \frac{\left(\mathbf{r}_{\chi_{\text{int}}}^{n+1,\text{old}}\right)^T \left(\mathbf{r}_{\chi_{\text{int}}}^{n+1,\text{new}} - \mathbf{r}_{\chi_{\text{int}}}^{n+1,\text{old}}\right)}{\left\|\mathbf{r}_{\chi_{\text{int}}}^{n+1,\text{new}} - \mathbf{r}_{\chi_{\text{int}}}^{n+1,\text{old}}\right\|^2} \tag{14.59}$$

where ω is the under-relaxation factor and $\mathbf{r}_{\chi_{\text{int}}}$ is the interface residual vector. Using the under-relaxation factor, the updated interface variables are obtained as

$$\chi_{\text{int},i}^{n+1,\text{new}} = \chi_{\text{int},i}^{n+1,\text{old}} + \omega_{\chi_{\text{int}}}^{n+1,\text{new}} r_{\chi_{\text{int},i}}^{n+1,\text{new}} \tag{14.60}$$

For a two-way coupled CFSD problem, Fig. 14.7 shows a flow chart for the implicit coupling between the CFD and CSD solvers. For the first iteration, the interface position, velocity, and acceleration in the present time instant are considered equal to that at the previous time instant for the CFD solver. After the solution of CFD solver, the structure equations are solved using the forces obtained at the fluid–solid interface. Using the solution obtained from the CSD solver, the interface variables are updated with an under-relaxation factor [Eq. (14.60)]. Further, the fluid flow is solved using the updated interface variables. The procedure is continued until convergence

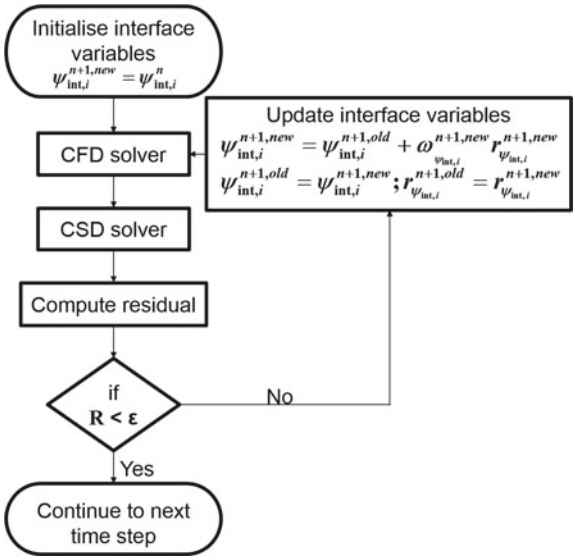


Fig. 14.7 Flow chart for the implicit coupling between the CFD and CSD solvers, for a two-way coupled CFSD

is obtained for the residual [Eq. (14.57)]. An order of accuracy study was presented in our recent study (Thekkethil and Sharma 2019), where the order of accuracy of the present HLE method was demonstrated as second order.

14.4 HLE Method-Based CFSD Application and Analysis

Our HLE method-based CFSD application and analysis are presented here in separate subsections for rigid and flexible structure-based FSD problems.

14.4.1 CFSD Application and Analysis for Fluid–Rigid Structure Dynamics

For the one-way coupled fluid–rigid structure dynamics, CFSD application and analysis are presented here first for 2D flow across a transversely oscillating cylinder and 2D as well as 3D hydrodynamics study on fish-like propulsion of fish-like undulating foil. For the fish-like locomotion, the 2D study is presented for both tethered propulsion and self-propulsion of a fish-like pitching/undulating NACA0012 hydrofoil; the 3D study is presented for tethered propulsion of a batoid fish-like locomotion. The tethered propulsion is simulated by a constant velocity u_∞-based free-stream cross-flow, while a time-wise varying velocity $u_\infty(t)$ is used for the self-propulsion; both the velocities correspond to the propulsion velocity u_p of the foil that is con-

stant $u_p = u_\infty$ for the tethered propulsion and time-varying $u_p(t) = u_\infty(t)$ for the self-propulsion. Here, $u_p(t)$ is obtained from the instantaneous thrust force, using Newton's II law of motion. The non-dimensional computational set-ups for all the one-way coupled CFSD problems are shown in Fig. 14.8.

14.4.1.1 Free-Stream Flow Across a Transverse Oscillating Cylinder

The transverse oscillating circular cylinder in a free-stream flow is a classical benchmark problem to test numerical methods for fluid flow across moving solid. The transverse oscillation is given as $y_e = A_e \sin(2\pi f_e t)$, where A_e is the amplitude and f_e is the frequency of oscillation. The non-dimensional parameters for the problem are the Reynolds number $\mathrm{Re} = \rho_f u_\infty D/\mu_f$, the non-dimensional amplitude A_e/D, and the frequency ratio f_e/f_o. Here, f_o is the natural frequency of vortex shedding.

For $\mathrm{Re} = 185$, $A_e/D = 0.2$, and $f_e/f_o = 1.0$, Fig. 14.9 shows an excellent agreement between our and published (Guilmineau and Queutey 2002) results for vorticity contours and streamlines. Furthermore, our results for mean thrust coefficient $C_{Tm} = 0.432$ and RMS value of lift force coefficient $C_{Lrms} = 1.548$ match very well with respective values of 0.410 and 1.503 reported in the literature (Guilmineau and Queutey 2002).

14.4.1.2 2D Hydrodynamic Study for Tethered Propulsion and Self-propulsion of Anguilliform and Carangiform Fishes-Like Undulating Hydrofoil

LS-IBM-based hydrodynamic analysis of fishes-like tethered propulsion study of a 2D NACA0012 hydrofoil was presented in our recent study (Thekkethil et al. 2018). A fish body is modelled by the foil of chord length c, and a unified kinematic model was proposed. The model is based on the wavelength λ of a travelling wave moving along the foil. The travelling wave-based unified kinematics is represented by a lateral displacement of the centreline of the foil $\triangle y$, given as

$$\triangle y = a(x) \sin\left(\frac{2\pi x}{\lambda} - 2\pi f t\right) \text{ where } a(x) = a_{\max}\frac{x}{c} \tag{14.61}$$

The wave equation consists of amplitude $a(x)$ (varying from head to tail of the foil), wavelength λ, and frequency f of the travelling wave. A linear amplitude variation is considered from head to tail, with maximum amplitude at the tail as $a_{\max}$. The non-dimensional parameters for the problem are the non-dimensional wavelength λ^* $(\equiv \lambda/c)$, non-dimensional frequency St $\left(\equiv 2fa_{\max}/u_p\right)$, non-dimensional maximum amplitude $A_{\max}$ $(\equiv a_{\max}/c)$, and Reynolds number $\mathrm{Re}_{u_p} = \rho_f u_p c/\mu_f$. The unified kinematic model [Eq. (14.61)] represents various types of fishes-like kinematics—anguilliform fishes-like kinematics for the smaller non-dimensional wavelength ($\lambda^* < 1$), caudal fin motion thunniform fishes-like kinematics for the

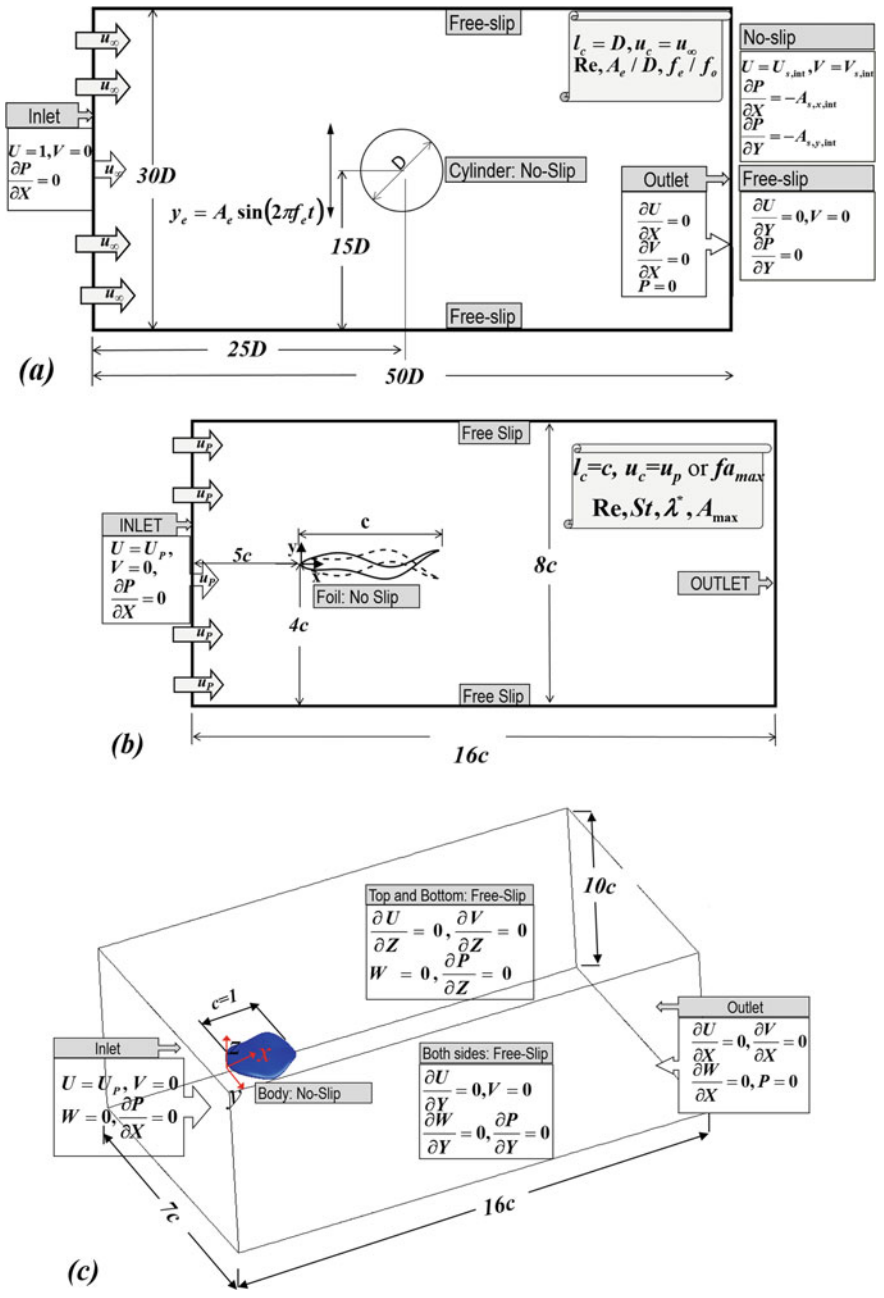


Fig. 14.8 Non-dimensional computational set-up for **a** free-stream flow across a transversely oscillating circular cylinder, **b** tethered/self-propulsion of fish-like undulating 2D NACA0012 hydrofoil, and **c** tethered propulsion of 3D batoid fish-like body

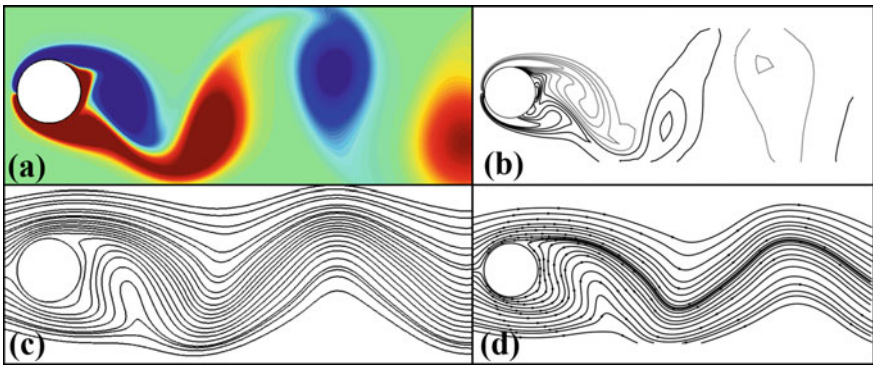


Fig. 14.9 **a**, **b** Streamlines and **c**, **d** pressure contours obtained from the **a**, **c** LS-IBM and **b**, **d** literature (Guilmineau and Queutey 2002), for the transverse oscillating cylinder in a free-stream flow at a time instant corresponding to maximum upward displacement of the cylinder, for constant $\mathrm{Re} = 185$, $A_e/D = 0.2$, and $f_e/f_o = 1.0$

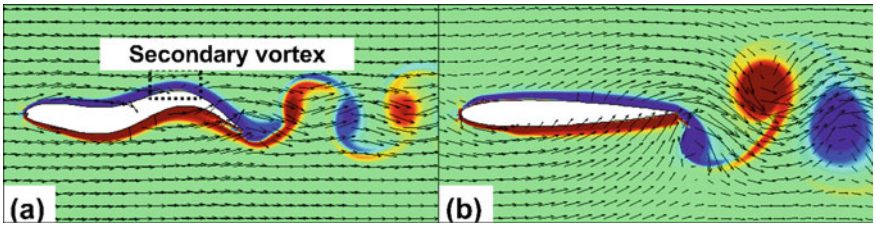


Fig. 14.10 Instantaneous vorticity contour and velocity vector during **a** anguilliform fishes-like undulation at $\lambda^* = 0.8$ and **b** carangiform fishes-like pitching at $\lambda^* = \infty$ for tethered propulsion of NACA0012 hydrofoil at $\mathrm{St} = 0.4$, $A_{\max} = 0.1$, and $\mathrm{Re}_{u_p} = 5000$

larger wavelength ($\lambda^* \to \infty$), and hypothetical fishes-like kinematics that is a combination of the anguilliform and thunniform fishes-like kinematics for the intermediate values of λ^*.

Figure 14.10 shows a reverse von Karman vortex street as a signature of thrust generation. Further, for smaller λ^* as compared to larger λ^*, the vortices are weaker and laterally stretched as compared to larger λ^*. The flow pattern results in a larger thrust force (efficiency) for larger (smaller) λ^*-based carangiform (anguilliform) fishes-like kinematics (Thekkethil et al. 2018).

For self-propelled anguilliform and carangiform fishes-like locomotion in our recent study (Thekkethil 2019) at a constant Reynolds number based on the frequency $\mathrm{Re}_f \left(\equiv \rho_f f a_{\max} c/\mu_f\right)$, Fig. 14.11 shows a temporal variation of vorticity contours and velocity vectors. For the initial time duration, the figure shows that λ^* results in a dipole formation with a strong lateral jet flow that leads to a larger hydrodynamic force and maximum stream-wise acceleration of the foil. Further, the figure shows a

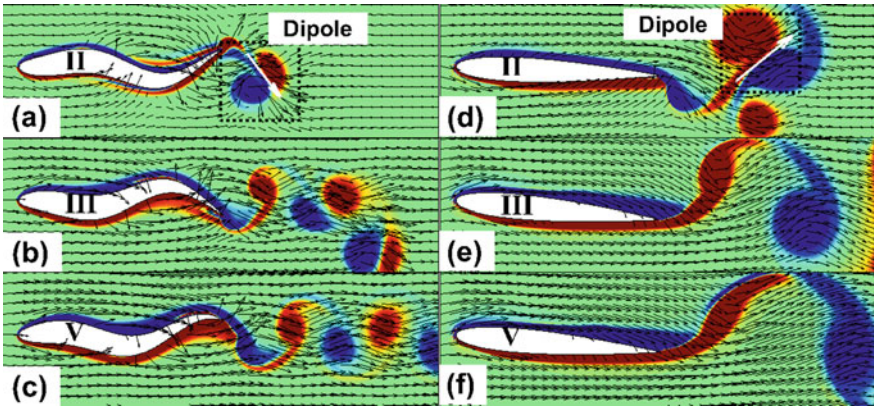


Fig. 14.11 Temporal variation of vorticity contours and velocity vectors during second, third, and fifth cycles of self-propulsion of NACA0012 hydrofoil for **a**–**c** anguilliform fishes-like undulation with $\lambda^* = 0.8$ and **d**–**f** carangiform fishes-like pitching with $\lambda^* = \infty$, at $\mathrm{Re}_f = 1000$ and $A_{\max} = 0.1$

decrease in the jet strength with time (due to the increase in the vortex spacing) that leads to a reduction in the hydrodynamic force. At the dynamic steady state, a zero net thrust force is obtained, resulting in a constant propulsion velocity.

14.4.1.3 3D Hydrodynamics Study for Tethered Propulsion of a Batoid Fishes-Like Body

Hydrodynamic analysis of various types of 3D batoid fishes-like locomotion was presented in our recent work (Thekkethil 2019). The batoid type of fishes uses a bat-like flapping of pectoral fin along with fishes-like undulation of body. The combined motion leads to a 3D kinematics. Figure 14.12 shows the shape of the batoid-like body considered in our recent study (Thekkethil 2019). The body has a hydrofoil cross section in the x-z plane with chord length c in the x-direction and an elliptical cross section in x-y and y-z planes with a span of b in the y-direction. The kinematics is a combination of the wavy motion in the x-z plane and symmetric pitching (a bird-like flapping) motion in the y-z plane. The combination of motions in the x-z and y-z planes can be represented by the transverse displacement of the body with respect to the x-y plane in dimensional form as

$$\triangle z = \frac{a_{\max}}{cb/2} x|y| \sin\left[2\pi\left(\frac{x}{\lambda} - ft\right)\right] \text{ where, } x = [0, c] \text{ and } y = [-b/2, b/2] \tag{14.62}$$

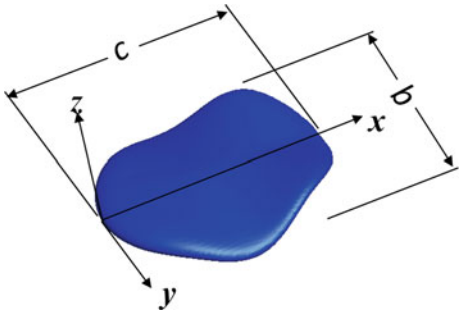


Fig. 14.12 3D view of the batoid-like body

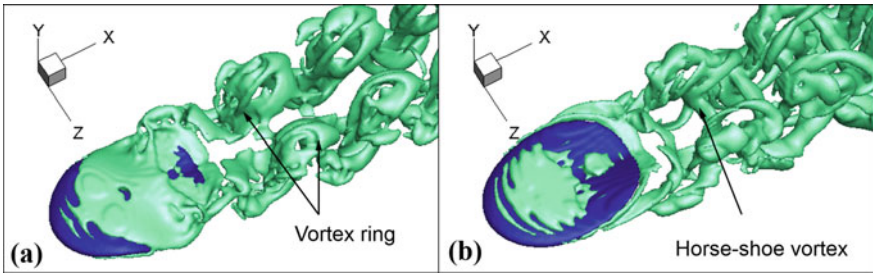


Fig. 14.13 Instantaneous Q-criterion-based vortex structure during a tethered propulsion of the batoid fishes-like body for **a** $\lambda^* = 0.8$ and **b** $\lambda^* = 4.0$, at $A_R = 0.75$, St $= 0.5$, $A_{\max} = 0.15$, and $\mathrm{Re}_{u_p} = 10{,}000$

Here, $a_{\max}$ is the maximum possible amplitude at $x = c$ and $|y| = b/2$. The wavelength of undulation λ and frequency of undulation f are similar to the 2D hydrofoil. For the 3D fishes-like locomotion, aspect ratio A_R ($\equiv b/c$) is an additional non-dimensional parameter. The various λ^* and A_R represent different types of batoid fishes-like locomotion.

For tethered propulsion of or constant propulsion-velocity-based free-stream flow across various types of 3D batoid fishes-like undulating hydrofoils, Fig. 14.13 shows instantaneous Q-criterion-based vortex structure. The figure shows a double pair of vortex rings with each pair on the front and backside connected by the vortex contrail for smaller λ^*. The two vortex rings are formed due to the symmetric pitching motion on both sides of the plane of symmetry. For the larger wavelength ($\lambda^* = 4.0$), a horseshoe vortex structure connecting the two vortex rings is present in addition to the vortex rings. This results in larger hydrodynamic forces for the larger λ^*.

14.4.2 CFSD Application and Analysis for Fluid–Flexible Structure Dynamics

As discussed above, for the fluid–flexible structure dynamics, the interaction between fluid dynamics and structural dynamics is two-way coupled—the fluid flow and structure motion/deformation are dependent on each other. In this section, the application of the present HLE method is presented for three problems: first, a lid-driven cavity-based flow across a flexible plate; second, a Poiseuille flow across a rigid cylinder with a flexible splitter plate; and third, tethered-propulsion-based free-stream flow across a flexible hydrofoil. The first problem is a computationally efficient benchmark problem, recently proposed by us (Thekkethil and Sharma 2019), the second problem is also a commonly used benchmark problem, and the third problem is an extension of our study on hydrodynamics during fishes-like locomotion. The non-dimensional computational set-ups for the two-way coupled CFSD problems are shown in Fig. 14.14.

14.4.2.1 Lid-Driven Cavity Flow-Based Benchmark Problem for Fluid–Flexible Structure Dynamics

We recently proposed (Thekkethil and Sharma 2019) a computationally efficient and easy-to-set up lid-driven cavity flow-based benchmark problem along with benchmark solutions for the two-way coupled FSD. The problem considers a square lid-driven cavity, with cavity length L, and both top and bottom wall act as a lid moving with a constant velocity. A flexible plate, of length $0.5L$ and thickness $0.05L$ hinged at the centre of the cavity, gets deformed due to the lid-driven cavity flow-based hydrodynamic force. In addition to the Reynolds number Re, the two-way coupled FSD problem considers the non-dimensional Young's modulus E^*, density ratio ρ_r, and Poisson's ratio ν_s as non-dimensional governing parameters.

Figure 14.15 shows the steady-state streamlines and pressure as well as vorticity contours. The lid-driven flow creates circular flows near the top and bottom boundaries of the cavity, which results in symmetric bending of the plate, as shown in the figure. The computational time taken for this problem is very small as compared to many benchmark problems reported in the literature.

14.4.2.2 Poiseuille Flow Across a Flexible Splitter Plate Behind a Cylinder

The problem corresponds to a hydrodynamically fully developed flow across a rigid circular cylinder of diameter D with a flexible splitter plate (of a thickness of $0.2D$ and length $3.5D$ attached behind it) in a channel. The problem was first proposed by Turek and Hron (2006) and is widely used as a benchmark problem in the literature.

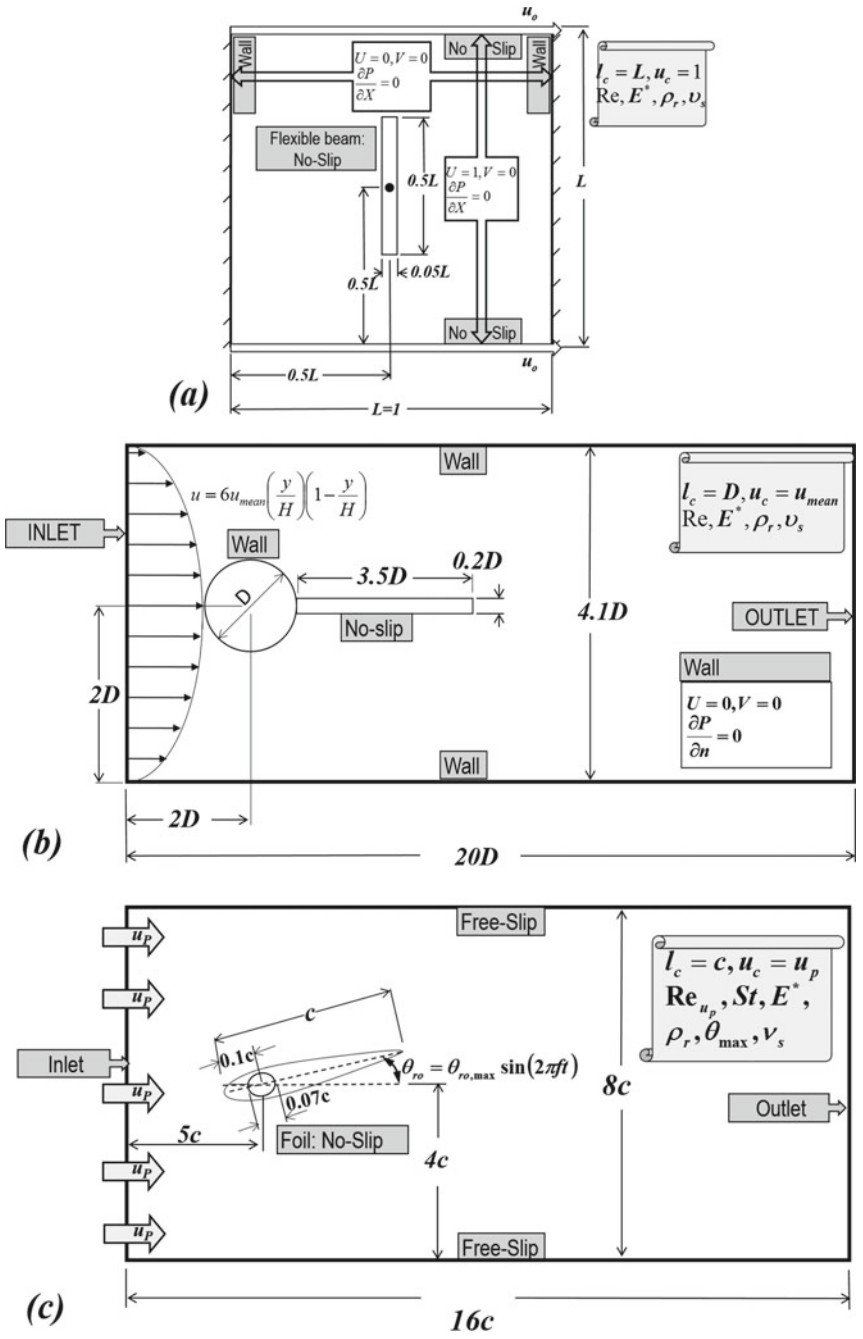


Fig. 14.14 Non-dimensional computational set-up for **a** the benchmark problem on the classical lid-driven cavity flow-induced deformation of a hinged vertical plate, **b** a rigid circular cylinder with a flexible splitter plate in a Poiseuille flow, and **c** tethered propulsion of structurally flexible hydrofoil subjected to pitching motion

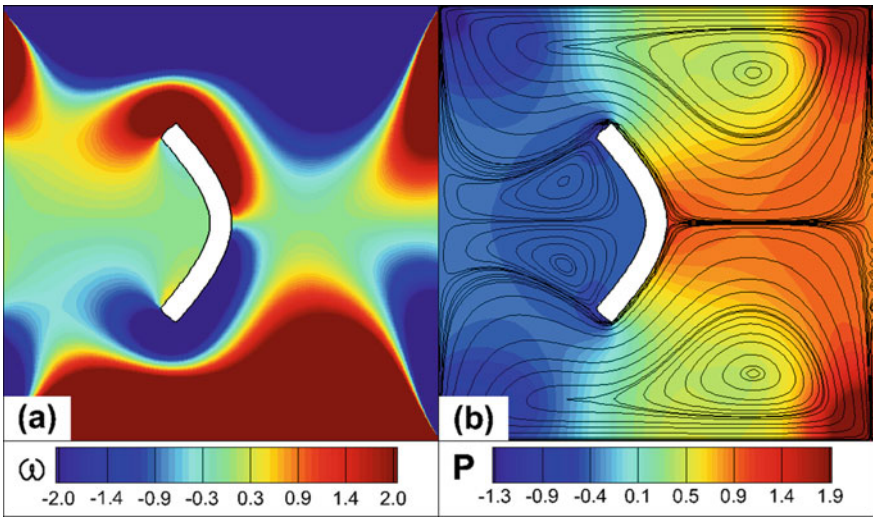


Fig. 14.15 Steady-state **a** vorticity contours and streamlines and **b** pressure contours, for hinged plate in a top and bottom lid-driven cavity at Re = 100, $E^* = 100$, $\rho_r = 10$, and $\nu_s = 0.3$

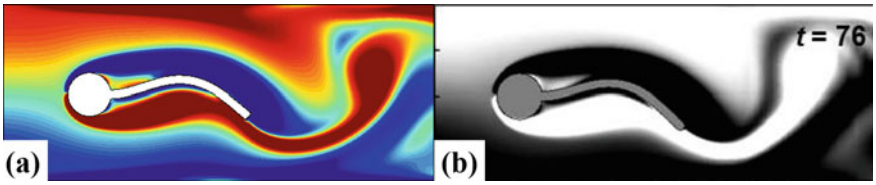


Fig. 14.16 Instantaneous vorticity contour obtained from the **a** LS-IBM and **b** literature (Bhardwaj and Mittal 2012), for the channel flow across flexible splitter plate attached behind rigid circular cylinder at a time instant $t = 76$, for constant Re = 100, $E^* = 1400$, $\rho_r = 10$, and $\nu_s = 0.4$

Figure 14.16 shows an excellent agreement between our (Thekkethil 2019) results and published (Bhardwaj and Mittal 2012) results for a periodic state. Due to the time-wise periodic hydrodynamic forces acting on the body, the plate is subjected to vibration and the periodic state is obtained after a certain number of vortex shedding cycles.

14.4.2.3 2D Hydrodynamics Study for Tethered Propulsion of a Fish-Like Pitching Flexible Hydrofoil

The arrangement of the flexible hydrofoil is shown in Fig. 14.14c. This was proposed in an experimental work (Marais et al. 2012) and was studied numerically in our recent work (Thekkethil 2019).

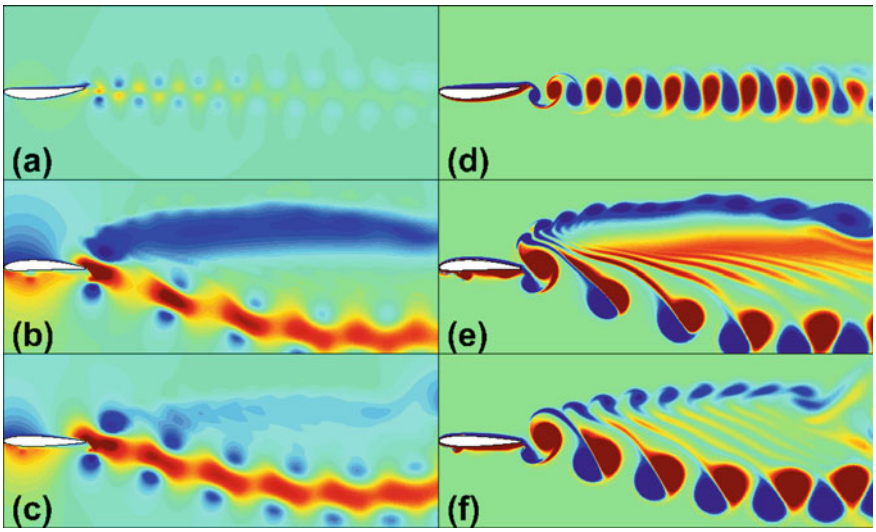


Fig. 14.17 Instantaneous **a**–**c** stream-wise velocity contours and **d**–**f** vorticity contours for the free-stream flow across pitching flexible hydrofoil with **a**, **d** largely flexible ($E^* = 5000$), **b**, **e** moderately flexible ($E^* = 30{,}000$), and rigid ($E^* = \infty$) hydrofoils, at St $= 0.5$, Re $= 5000$, $\theta_{\text{ro,max}} = 8^o$, $\rho_r = 1.0$, and $\nu_s = 0.4$

For the tethered-propulsion-based free-stream flow across the flexible pitching hydrofoil, Fig. 14.17 shows the vorticity contours and stream-wise velocity contours. The figure shows a single straight-jet flow (straight reverse von Karman vortex street) for the largely flexible foil, inclined jet flow (inclined von Karman vortex street) for the rigid foil, and inclined jet flow along with a straight-wake (inclined reverse von Karman vortex street with straight von Karman vortex street) for the moderately flexible foil. The moderate (large) flexibility results in maximum (minimum) thrust generation.

14.5 Closure

This chapter is presented in two parts: first, CFSD development, and second, CFSD application and analysis. For the first part on CFSD development, a detailed numerical methodology in two-dimensional Cartesian coordinate system is presented for the partitioned approach-based hybrid Lagrangian–Eulerian (HLE) method. The methodology is based on physical law-based FVM and level-set-based IBM for CFD development and geometric nonlinear Galerkin FEM-based CSD development, along with an implicit coupling between the CFD and CSD solvers that is numerically sta-

ble for large deformation. The second part demonstrates the HLE method-based CFSD application (with the help of computational set-up) and analysis (of hydrodynamic results) on a variety of rigid and flexible structure-based one-way and two-way coupled CFSD problems, respectively.

References

Belytschko T, Kennedy JM (1975) Finite element study of pressure wave attenuation by reactor fuel subassemblies. J Press Vessel Technol 97(3):172–177

Bhardwaj R, Mittal R (2012) Benchmarking a coupled immersed-boundary-finite-element solver for large-scale flow-induced deformation. AIAA J 50(7):1638–1642

Degroote J, Haelterman R, Annerel S, Bruggeman P, Vierendeels J (2010) Performance of partitioned procedures in fluid-structure interaction. Comput Struct 88(7–8):446–457

Donea J, Fasoli-Stella P, Giuliani S (1976) Finite element solution of transient fluid-structure problems in Lagrangian coordinates. Technical report

Guilmineau E, Queutey P (2002) A numerical simulation of vortex shedding from an oscillating circular cylinder. J Fluids Struct 16(6):773–794

Jacobson A, Kavan L, Sorkine-Hornung O (2013) Robust inside-outside segmentation using generalized winding numbers. ACM Trans Graph (TOG) 32(4):33

Majumdar S, Iaccarino G, Durbin P(2001) Rans solvers with adaptive structured boundary non-conforming grids. In: Annual research briefs, Center for Turbulence Research, Stanford University, pp 353–466

Marais C, Benjamin T, José EW (2012) Godoy-Diana R (2012) Stabilizing effect of flexibility in the wake of a flapping foil. J. Fluid Mech. 710:659–669

Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261

Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, Von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227(10):4825–4852

Namshad T (2019) Computational fluid-structure dynamics development and its application for analysis of various types of 2D/3D fishes-like kinematics, propulsion, & flexibility. Ph.D. thesis, Indian Institute of Technology Bombay

Noh WF (1963) CEL: a time-dependent, two-space-dimensional, coupled Eulerian-Lagrange code. Technical report, Lawrence Radiation Laboratory, University of California, Livermore

Pan D (2006) An immersed boundary method for incompressible flows using volume of body function. Int J Numer Methods Fluids 50(6):733–750

Patankar S (2018) Numerical heat transfer and fluid flow. CRC Press, Boca Raton

Patel JK, Natarajan G (2018) Diffuse interface immersed boundary method for multi-fluid flows with arbitrarily moving rigid bodies. J Comput Phys 360:202–228

Peskin CS (2002) The immersed boundary method. Acta Numer 11:479–517

Sethian JA (1999) Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science, vol 3. Cambridge University Press, Cambridge

Sharma A (2017) Introduction to computational fluid dynamics: development, application and analysis. Wiley & Athena, UK; Ane Books Pvt. Ltd. New Delhi

Shrivastava M, Agrawal A, Sharma A (2013) A novel level set-based immersed-boundary method for CFD simulation of moving-boundary problems. Numer Heat Transf B 63(4):304–326

Thekkethil N, Sharma A (2019) Level set function based immersed interface method and benchmark solutions for fluid flexible-structure interaction. Int J Numer Methods Fluids 91(3):134–157

Thekkethil N, Sharma A, Agrawal A (2018) Unified hydrodynamics study for various types of fishes-like undulating rigid hydrofoil in a free stream flow. Phys Fluids 30(7):077107

Turek S, Hron J (2006) Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In: Fluid-structure interaction. Springer, Berlin, pp 371–385

Udaykumar HS, Mittal R, Rampunggoon P, Khanna A (2001) A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. J Comput Phys 174(1):345–380

Versteeg HK, Malalasekera W (2007) An introduction to computational fluid dynamics: the finite volume method. Pearson Education

Zienkiewicz OC, Taylor RL, Nithiarasu P, Zhu JZ (1977) The finite element method, vol 3. McGraw-Hill, London

Chapter 15
Immersed-Boundary Methods for Simulating Human Motion Events

Jung-Il Choi and Jack R. Edwards

15.1 Introduction

Immersed-boundary methods are a general class of technique that indirectly imposes the effects of a (possibly moving) solid surface on the surrounding flow. While the original immersed-boundary method dates from the work of Peskin (1972), the technique was recast into a form more useful for conventional CFD strategies by Mohd-Yosuf (1997), Verzicco et al. (2000), Fadlun et al. (2000), and others. A review article summarizing these and other techniques is that of Mittal and Iaccarino (2005). A key to these newer immersed-boundary methods is the enforcement of fluid boundary conditions indirectly, through specification of the distribution of the fluid velocity in the vicinity of the immersed boundary. This paper presents a generalization of an immersed-boundary method developed for time-dependent, incompressible flows in Choi et al. (2007). This approach is similar to that of Gilmanov et al. (2003) in that a surface mesh consisting of structured or unstructured elements is embedded within a flow and that flow property variations normal to the surface are reconstructed. The surface meshes may be closed (surrounding a volume of space) or zero-thickness (surfaces alone). The Navier–Stokes equations are solved in cells outside the body (field cells); a constant property condition is enforced for cells inside the body (interior cells); and boundary conditions are enforced through specifying distributions of fluid properties in a collection of band cells just outside the immersed body (band cells). In contrast to many other IB techniques, the methods developed in Choi et al. (2007) can be applied to turbulent flows at high Reynolds numbers by virtue of the use

J.-I. Choi
Department of Computational Science and Engineering, Yonsei University, Seoul 03722, Republic of Korea
e-mail: jic@yonsei.ac.kr

J. R. Edwards (✉)
Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, NC 27695, USA
e-mail: jredward@ncsu.edu

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_15

of power-law interpolation techniques to mimic the near-wall profile of an attached turbulent flow. The methods are also applicable to general curvilinear meshes as well as unstructured meshes. Since the publication of Choi et al. (2007), extensions to particle-laden incompressible flows (Oberoi et al. 2010; Choi et al. 2012), gas-phase contaminant transport (Choi and Edwards 2008, 2012), and compressible, turbulent flows (Ghosh et al. 2010a, b, 2012) have been developed. All of these studies have rendered immersed objects as point clouds, which has advantages if the object is sufficiently detailed but becomes inconvenient if the object is relatively featureless. This report outlines a way of embedding stereo-lithography (STL) files as immersed objects within a computational domain and introduces several techniques for converting scenarios involving complicated and possibly moving objects into detailed large-eddy flow simulations driven by immersed-boundary motion. The presented applications involve realistic human motion activity as well as secondary effects such as buoyancy-driven flow resulting from the human thermal plume.

15.2 Numerical Methods

15.2.1 Governing Equations

For a three-dimensional, time-dependent incompressible flow, the grid-filtered governing equations for a fluid phase can be written as

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0, \tag{15.1}$$

$$\frac{\partial \rho \bar{u}_i}{\partial t} + \frac{\partial}{\partial x_j}(\rho \bar{u}_i \bar{u}_j + \bar{p}\delta_{ij} - \overline{\tau}_{ij} + \tau_{ij}^{\mathrm{SGS}}) = \bar{f}_i, \tag{15.2}$$

where $\bar{u}_i$ is the velocity vector, ρ is the density of the fluid, $\bar{p}$ is the pressure, $\bar{f}_i$ is an external force, μ is the molecular viscosity, $\overline{\tau}_{ij}$ is the viscous stress tensor for a Newtonian fluid, and τ_{ij}^{SGS} is the subgrid-scale (SGS) stress tensor. Note that the overbar represents grid-filtered variables. Based on the Smagorinsky model (Smagorinsky 1963), which assumes that the SGS stress tensor is proportional to the velocity strain rate $\overline{S}_{ij}$, the SGS stress tensor is modeled as $\tau_{ij}^{\mathrm{SGS}} = -2\mu_t \overline{S}_{ij}$. The subgrid-scale eddy viscosity is defined as $\mu_t = \rho(C_s\Delta)^2(2\overline{S}_{ij}\overline{S}_{ij})^{1/2}$, where $C_s(= 0.1)$ is the Smagorinsky constant and Δ is a local grid-filter width, which is set equal to the cube root of the mesh-cell volume.

The mass conservation equations for transport of a set of passive gaseous contaminants in Eulerian framework (Crowe et al. 1996) are as follows:

$$\frac{\partial \overline{\rho}_k}{\partial t} + \frac{\partial}{\partial x_j}(\overline{\rho}_k(\bar{u}_i + \bar{v}_{k,j})) = 0, \tag{15.3}$$

where the subscript k denotes the kth gas species. Here, $\overline{\rho}_k$ is the mass density of species k. The diffusion velocity $\tilde{v}_{k,i}$ is given by Fick's law:

$$\tilde{v}_{k,i} = -\left(\frac{\mu}{Sc} + \frac{\mu_t}{Sc_t}\right)\frac{1}{\bar{\rho}_k}\frac{\partial \overline{Y}_k}{\partial x_i}, \tag{15.4}$$

where the mass fraction $\overline{Y}_k = \overline{\rho}_k/\rho$ and the laminar and turbulent Schmidt numbers are assigned values of 0.72 and 1.0, respectively (Crowe et al. 1996). It is assumed that the mass fractions of the tracer-gas species are small enough that the density of the carrier gas is not affected significantly. We extrapolate contaminant concentration to all physical surfaces; the contaminant concentration is set to zero inside all immersed surfaces.

Under incompressible flow assumptions, the evolution of temperature θ can be written as:

$$\rho C_p\left(\frac{\partial \overline{\theta}}{\partial t} + \bar{u}_j\frac{\partial \overline{\theta}}{\partial x_j}\right) = \frac{\partial}{\partial x_j}\left((\alpha + \alpha_t)\frac{\partial \overline{\theta}}{\partial x_j}\right) + \dot{Q}, \tag{15.5}$$

where $\overline{\theta}$ is temperature, C_p is specific heat capacity at constant pressure, α is the thermal conductivity, α_t is the turbulent thermal conductivity, and $\dot{Q}$ is an external heat source. The thermal conductivities α and α_t are related to the molecular and eddy viscosities through the assumption of constant laminar and turbulent Prandtl numbers (0.72 and 0.9, respectively). Equation (15.5) is solved subject to isothermal, adiabatic, or imposed heat-flux boundary conditions at solid surfaces. Buoyancy effects resulting from temperature gradients are imposed in Eq. (15.2) using the Boussinesq approximation: $\bar{f}_i = \rho_\infty g_i(1 - \overline{\theta}/\theta_\infty)$, where g_i is the gravitational force and the subscript $_\infty$ denotes the undisturbed-flow state.

Basic formulation We solve the three-dimensional incompressible Navier–Stokes equations using a finite volume approach. Time integration of the discrete Navier–Stokes equations is achieved by an artificial compressibility approach (Chorin 1967) which is facilitated by a dual time-stepping procedure at each physical time step. At time level $n + 1$, sub-iteration k, the solution of the discrete representation of Eqs. (15.1) and (15.2) can be written as

$$\mathbf{A}(\mathbf{V}^{n+1,k+1} - \mathbf{V}^{n+1,k}) = -\mathbf{R}^{n+1,k}. \tag{15.6}$$

The flow variables $\mathbf{V} = (\bar{p}, \bar{u}_i)^T$ are advanced from time level n ($\mathbf{V}^{n+1,k=0} = \mathbf{V}^n$) to time level $n + 1$ ($\mathbf{V}^{n+1} = \mathbf{V}^{n+1,k=k_{max}}$) over a number of sub-iterations $k_{\max}$. The system Jacobian matrix is denoted as $\mathbf{A}$, and the corresponding residual vectors $\mathbf{R} = (R_c, R_{M_i})^T$ can be written as

$$R_c^{n+1,k} = \left[\frac{\partial \bar{u}_i}{\partial x_i}\right]^{n+1,k} \tag{15.7}$$

$$R_{M_i}^{n+1,k} = \rho\left(\frac{3\bar{u}_i^{n+1,k} - 4\bar{u}_i^n + \bar{u}_i^{n-1}}{\Delta t}\right) + \left[\frac{\partial}{\partial x_j}(\rho\bar{u}_i\bar{u}_j + \bar{p}\delta_{ij} - \overline{\tau}_{ij} + \tau_{ij}^{\mathrm{SGS}}) - \rho\bar{f}_i\right]^{n+1,k}. \tag{15.8}$$

Equation (15.8) is solved approximately at each sub-iteration using an implicit technique based on incomplete LU decomposition (Wesseling 1995). For the spatial discretization, the inviscid fluxes in the governing equations are discretized using a low-diffusion flux-splitting scheme (LDFSS) (Edwards and Liou 1998; Neaves and Edwards 2006), while second-order central differencing methods are used to discretize the viscous components. For the cases presented later, higher-order spatial accuracy for the interface fluxes is achieved by using the piecewise parabolic method (Colella and Woodward 1984). The effects of smaller subgrid fluctuations are modeled using a Smagorinsky subgrid eddy viscosity (Baurle et al. 2003). The present flow solver uses METIS (Karypis and Kumar 1998) to partition a general multi-block grid over the number of allowable processors. Message-passing interface (MPI) communication routines are used to pass information among the processors. The incompressible flow solver and its components have been validated for a range of model problems (Edwards and Liou 1998).

15.2.2 Cell-Classification Procedure

We develop a classification algorithm for computational nodes based on the signed distance function $\Phi(\mathbf{x}, t)$, which is less than zero for cells within a closed immersed body and greater than zero for cells outside the body. Special procedures discussed later are used to handle zero-thickness immersed surfaces for which the signed distance is always positive.

Classification of computational cells The Heaviside function $G(\Phi(\mathbf{x}, t))$ is defined to be one for points just outside the immersed body and within the immersed body and is zero otherwise. The calculation of the Heaviside function is initiated by first initializing $G(\Phi(\mathbf{x}_k, t)) = 0$ for all points $\mathbf{x}_k$. Then, given a point $\mathbf{x}_k$, if $\Phi(\mathbf{x}_k, t) > 0$ and if any $\Phi(\mathbf{x}_m, t) < 0$, where $\mathbf{x}_m$ is a face, edge, or vertex neighbor of $\mathbf{x}_k$, then $G(\Phi(\mathbf{x}_k, t))$ is set to 1. If $\Phi(\mathbf{x}_k, t) \leq 0$, then $G(\Phi(\mathbf{x}_k, t))$ is also set to 1. The set of nearest neighbors, for a structured grid discretized according to a cell-centered finite volume method, is generally defined as the 26 cells that are immediately adjacent to a particular mesh cell, though smaller subsets can be used. Finally, we can define the Heaviside function as

$$G(\Phi(\mathbf{x}_k, t)) = \begin{cases} 0 \text{ for } \mathbf{x}_k \in \Omega_F \\ 1 \text{ for } \mathbf{x}_k \notin \Omega_F \end{cases}, \tag{15.9}$$

where Ω_F represents the set of the node points shown as the open circles in Fig. 15.1.

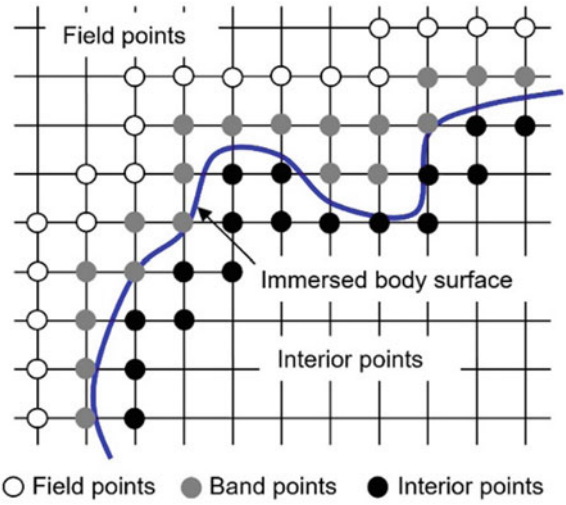


Fig. 15.1 Schematic illustrating classification of cell-centered points for a complex immersed body surface. Open, gray, and close circles represent field (Ω_F), band (Ω_B) and interior points (Ω_I), respectively, and thick line represents an immersed body surface. Adapted from (Choi et al. 2007)

The classification of the node points can be summarized as follows:

- Field points: $\mathbf{x}_k \in \Omega_F$ if $\Phi(\mathbf{x}_k, t) > 0$ and $G(\Phi) = 0$,
- Band points: $\mathbf{x}_k \in \Omega_B$ if $\Phi(\mathbf{x}_k, t) > 0$ and $G(\Phi) = 1$,
- Interior points: $\mathbf{x}_k \in \Omega_I$ if $\Phi(\mathbf{x}_k, t) \leq 0$ and $G(\Phi) = 1$.

where Ω_B and Ω_I represent the set of the node points shown as the gray and closed circles in Fig. 15.1, respectively. The zero iso-surface of the signed distance function defines the immersed body surface.

Surface definition in a computational domain The most popular way to describe 3D objects in computer system is to construct surface meshes composed of triangular elements (henceforth referred to as triangle meshes). This can be done using a computer-aided design (CAD) format or through other means, but the key is that triangle elements with an outward-pointing normal vector are created for each separate component of the object, as different components may move at different rates. The next step is to define 3D surfaces using the *unsigned* distance and classification whether an arbitrary point in a background domain is inside or outside of the objects. Classification can be achieved by counting intersections of a ray going from the given point (outside point from the object) to infinity since the number of intersections must be odd if the point is inside—this is called a *ray tracing method* (Linhart 1990). Another means of classification is to define a signed distance using the inner product between a pseudo-normal vector and a distance vector to an arbitrary point from its closest point on the surface—this is known as a *signed distance computation* (Gouraud 1971; Bærentzen and Aanæs 2005). While the former method needs to visit the parts of the triangle mesh along the ray tracing line, the latter algorithm needs to find the closest point on the mesh. We will apply the signed distance computation which is faster than ray tracing method in order to define 3D surfaces which will be incorporated with the present immersed-boundary method.

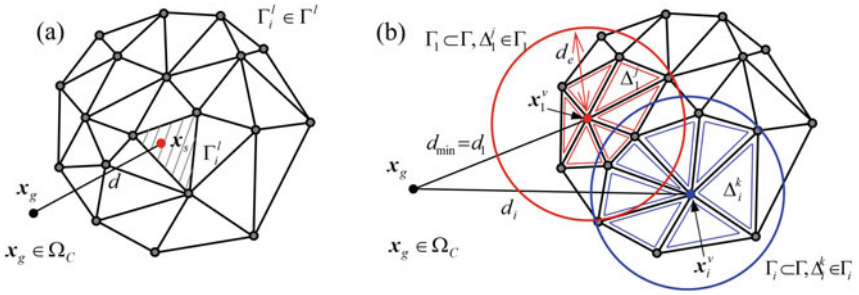


Fig. 15.2 Schematics for **a** a minimal distance from the points in a computation domain to surface points and **b** nearest neighbors for triangle elements. Adapted from (Edwards et al. 2010)

The distance d from grid points $\mathbf{x}_g$ in a computational domain Ω_C to the closest surface point $\mathbf{x}_s$ on triangle meshes Γ^l for lth component is simply defined as $d = \|\mathbf{x}_g - \mathbf{x}_s\|$ in Fig. 15.2a. Computation of the distance to 3D objects can be achieved by using brute force computation, a Voronoi diagram (Hoff et al. 1999), or hierarchical data structures (Payne and Toga 1992; Guéziec 2001). Among these methods, we use a k-d tree hierarchical data structure with a bounding box to accelerate finding the nearest triangle mesh element. For simplicity, we consider the one component's closed surface as shown in Fig. 15.2b. At first, we find a cloud of nearby points $\mathbf{x}_i^v$ from the given point $\mathbf{x}_g$ in a bounding box, in order of the closest distance, using an approximate nearest-neighbor (ANN) searching algorithm (Arya et al. 1998). The next step is to search the closest point in the set of the neighbor triangle meshes $\Delta_i^j \in \Gamma_i$ which are shared with a cloud of nearby vertices $\mathbf{x}_i^v$ since the closest vertex is typically different from the closest point on a triangle mesh. We can define the subset $\Gamma_s = \{\Gamma_i\}$ of the total triangle meshes Γ. Based on the subset Γ_s, the minimum distance can be obtained using point-triangle, point-edge, and point-vertex distance calculations.

In the search process, the subset Γ_s can be reduced using geometric restriction. Modern CAD programs enhance the uniformity of the triangles and control the edge distances. At a given edge distance d_e, we can get a restriction for the searching algorithm. As shown in Fig. 15.2b, the circles show the spheres with radius d_e and origin $\mathbf{x}_i^v$. The entire triangle neighbors Δ_i^j shared with the vertex $\mathbf{x}_i^v$ are included within the spheres. The distances d_i^j in the subset Γ_i are bounded as $|d_i^k - d_i| \le d_e$ with respect to the point-vertex distance d_i. Also, the distance is $|d_1^j - d_1| \le d_e$ for the subset Γ_1 which is the equivalent subset for the minimum point-vertex distance. The difference between two point-vertex distances can be written as $d_i^j - d_1^k - 2d_e < d_i - d_1 < d_i^j - d_1^k + 2d_e$. If $d_i^j < d_1^k$, the difference should be bounded as $d_i - d_1 < 2d_e$. Therefore, the above nearest distance calculation should be repeated for the ith nearest vertex point in the ANN list which satisfies $d_i - d_1 < 2d_e$.

Signed distance computation The signed distance function Φ can be obtained by multiplying the unsigned distance d with the sign of the dot product of the distance vector with the outward normal vector $\mathbf{n}$:

$$\Phi = \text{sgn}((\mathbf{x}_g - \mathbf{x}_s) \cdot \mathbf{n})\, d, \tag{15.10}$$

where $\text{sgn}(\varphi)$ returns a value of 1 for each nonnegative element and -1 for each negative element of φ and $\| \ \|$ denotes the magnitude of the vector.

This simple procedure was found not to work properly for some very complex CAD objects (Choi et al. 2007). Usually, the CAD objects are defined as triangular surface elements that contain each vertex and face-normal vector. If a nearest surface point at a given field point is located on an edge or at a vertex, the simple signed distance function may not be calculated correctly. Therefore, we consider an angle-weighted pseudo-normal vector (Bærentzen and Aanæs 2005), which is defined at surface nodes (vertices) or edges, rather than cell centers of surface triangles. For a given vertex $\mathbf{x}_v$, we identify the triangle elements shared with the vertex and calculate the incident angle α_i for each element with the outward-pointing face-normal vector $\mathbf{n}_i$ (Choi et al. 2007). The angle-weighted pseudo-normal vector $\mathbf{n}_v$ at the vertex can be defined as

$$\mathbf{n}_v = \frac{\sum_i \alpha_i \mathbf{n}_i}{\left\| \sum_i \alpha_i \mathbf{n}_i \right\|}, \tag{15.11}$$

where i denotes the triangle elements that surround the vertex and $\| \ \|$ denotes the magnitude of the vector. Based on the pseudo-normal vector at the vertex and face-normal vector $\mathbf{n}_i$ at the element center $\mathbf{x}_i$, we can determine an inside/outside decision using the same signed distance function in Eq. (15.10) with the data set of the vertices. This procedure essentially averages local fluctuations in the outward normal that could result from small features in the CAD file.

To define a global signed distance function Φ at any given mesh point, a simple priority rule is exercised. First, the global distance function is initialized to a large number. Then, the global signed distance function at a particular point is taken as the minimum of the individual signed distance functions for each component l at that point:

$$\Phi = \min_l(\Phi_l). \tag{15.12}$$

The collections of points that comprise the surfaces are allowed to move according to prescribed rate laws.

Embedding of CAD objects as immersed surfaces One of our major goals is to be able to incorporate general stereo-lithography (STL) files as immersed objects in our program without any additional user intervention. As discussed earlier, the main challenge is in accurately computing the signed distance from any of our mesh points to the nearest point on the STL surface. This challenge is made more difficult for

objects that contain large, flat panels (usually rendered as two triangles) and smaller features that are more refined. In our earlier work, we simply mesh-refined the objects until a clear rendering was achieved, but this required significant pre-processing and led to STL files that could be very large (millions of cells). In this work, we develop a module for directly reading STL files and for computing nearest distances and normal vectors to any panel, edge, or node on the surface. A detailed step-by-step procedure is as follows:

Step 1: Import STL file (ASCII format) and determine element-to-element connectivity. The STL file provides coordinates of vertices of each triangle along with the normal vector associated with the face center of each triangle.
Step 2: Calculate, for every triangle, coordinates of the face center and the midpoint of each edge, pseudo-normal vectors at each vertex, and normal vectors at the midpoint of each edge.
Step 3: Add these additional coordinates/normal vectors to the database.
Step 4: Given a particular field point, use approximate nearest-neighbor (ANN) searching (Arya et al. 1998) to determine a set of nearest vertices to that point.
Step 5: Determine whether the true nearest point to the surface lies on a triangle, at a vertex, or on an edge.
Step 6: Based on this decision, find the nearest point and assign the appropriate normal vector (face-centered, pseudo-normal, or edge-centered) to this point. Calculate signed distance functions at each query cell.

The search for an initial subset of possible vertices is an $N \log(m)$ operation, so this approach is still relatively efficient. Note that, N and m are the number of query points and the listed data points (possible vertices), respectively. The case of very large triangles neighboring small triangles, however, can require expanding the initial subset decision space to include all possible triangles, leading to a complexity of $O(Nm)$.

For moving objects, we initially read ASCII-formatted STL files for the objects at each time step. This reading sequence required non-trivial I/O access times compared to the entire computation. Thus, we developed an improved STL reader to accelerate the reading sequence using binary formatted STL files as well as avoiding redundant procedures. The current status of the reader is that it is able to read directly immersed objects from STL files (ASCII or binary format) and can separate automatically multiple objects in STL files into smaller segments.

Figure 15.3 shows the original avatar rendering in 3DSMax®, the surface triangulation, and the rendered image as an immersed body in the computational domain. The avatar consists of four segments such as body, head, hat, and gun. In order to make a closed immersed surface for the soldier, we merged the body, head, and hat into a single object. Note that, we maintain the skinning in the merging procedure for the original biped motion.

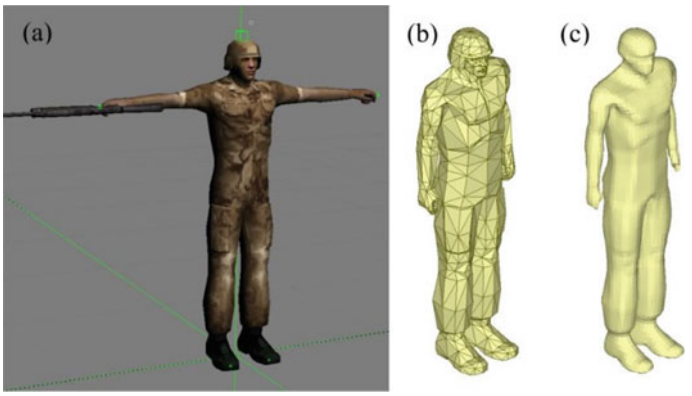


Fig. 15.3 Soldier avatar: **a** 3DSMax® rendering, **b** triangle elements in a STL file, and **c** immersed object rendering in a computational domain

15.2.3 Immersed-Boundary Formulation

Given the classification of the computational domain into field, band, and interior cells as described above, a direct forcing approach is used to enforce the boundary conditions at the interior and band cells. This results in the residual form of the governing equation system shown below which is then solved implicitly, coupled with exterior cells, by use of sub-iteration techniques:

$$\widetilde{R}_i^{n+1,k} = (1 - G(\Phi^{n+1}))R_i^{n+1,k} + G(\Phi^{n+1})\left[\frac{V_i^{n+1,k} - V_{B,i}^{n+1,k}}{\Delta t}\right], \quad i = c, M_x, M_y, M_z. \tag{15.13}$$

This equation represents the blending of the Navier–Stokes residual with a source term that relaxes the primitive variable vector $\mathbf{V} = (\bar{p}, \bar{u}_i)^T$ to its band-cell values. As discussed earlier, other equations representing transport of species concentration and heat may be added to this system.

Determination of information at the interpolation point The developments follow hinge on the determination of flow properties $q(d_I)$ at a certain distance d_I away from the surface (see Fig. 15.4). Given a point within the band $\mathbf{x}_k$ and a list of nearest neighbors to that point $\mathbf{x}_l$, a merit function w_l is defined as

$$w_l = \frac{1}{\sqrt{(|\mathbf{x}_l - \mathbf{x}_k|)^2 - ((\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n})^2 + \varepsilon}} \text{ for } (\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n} > 0, \tag{15.14}$$

otherwise $w_l = 0$.

In this, $(\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n}$ is the projection of the distance from $\mathbf{x}_k$ to $\mathbf{x}_l$ in the direction of the outward normal, and $\|\mathbf{x}_l - \mathbf{x}_k\|$ is the magnitude of the distance vector itself.

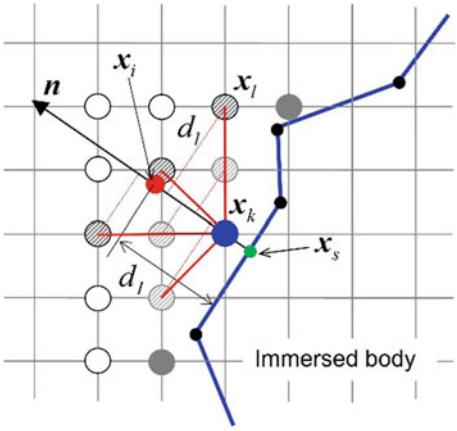


Fig. 15.4 Schematic determination of the distance d_I between the interpolation point $\mathbf{x}_i$ and surface node point for a given band point $\mathbf{x}_k$ using the projected distance d_I from neighbor points $\mathbf{x}_l$ to outward normal line based on surface-normal vector $\mathbf{n}$ at the immersed surface points $\mathbf{x}_s$ marked by green circle. Large closed circle represents the band point to be interpolated with the information at neighbor point. Hatched black and gray circles represent the field points and band points associated with the present determination, respectively

If point $\mathbf{x}_l$ is located directly along the outward normal line corresponding to band point $\mathbf{x}_k$, and if $(\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n}$ is positive, meaning that point $\mathbf{x}_l$ is further away from the surface than point $\mathbf{x}_k$, then the merit function returns a very large value (~$1/\varepsilon$, where ε is 10^{-12}).

The actual calculation of w_l is performed in three stages. First, only field points (those with $\Phi(\mathbf{x}_l, t) > 0$ and $G(\Phi(\mathbf{x}_l, t)) = 0$) are considered as members of the list of nearest neighbors. Then, w_l is calculated according to Eq. (15.14), and the sum of the weights $\sum_m w_m$ is calculated. If this sum is nonzero, then the actual weight function for each nearest neighbor is determined as

$$\omega_l = \frac{w_l}{\sum\limits_m w_m}. \tag{15.15}$$

Otherwise, the process is repeated, now considering both field points and other band points as members of the list of nearest neighbors. If this application also results in no viable interpolation points being found, then the band point $\mathbf{x}_k$ is effectively set to an interior point.

The location at which interpolated properties are defined, d_I, is calculated for a particular field point as

$$d_I = \sum_l \omega_l (\mathbf{x}_l - \mathbf{x}_k) \cdot \mathbf{n}. \tag{15.16}$$

Note that, this distance is in the direction of the normal coordinate. With this, the fluid properties $q(d_I)$ are found by applying the weighting functions,

$$q(d_I) = \sum_m q_m \omega_m. \tag{15.17}$$

Variable reconstruction in band cells The following closures are used for the fluid properties in the band cells, where the subscript 'I' indicates properties obtained at an interpolation point located along the normal line extending outward from the nearest surface location corresponding to the band cell in question, and the subscript 'B' indicates the band cell.

$$\begin{aligned}
&p_B = p(d_I)\\
&u_{B,i} - u_{S,i} = u_{T,i}(d_I)\left(d_B \big/ d_I\right)^k + u_{N,i}(d_I) f_N(d_I, d_B),\\
&u_{N,i}(d_I) = (u_j(d_I) - u_{S,j})\mathbf{n}_j \mathbf{n}_i,\\
&u_{T,i}(d_I) = (u_i(d_I) - u_{S,i}) - u_{N,i}(d_I)
\end{aligned} \tag{15.18}$$

In these expressions, $\mathbf{n}$ is the normal vector at the closest point on the body surface, d is a distance from the nearest surface point, $u_{S,j}$ is the velocity at the nearest surface point, and k is a power-law. The choice of k allows the model to replicate a turbulent velocity profile ($k = 1/7$ or $1/9$) or a laminar profile ($k = 1$). To obtain the temperature distribution near the surface, the following expressions are utilized. These are a low Mach-number simplification of more general relations derived from Walz's formula (Walz 1969):

Isothermal wall:

$$\frac{T_B}{T(d_I)} = \frac{T_w}{T(d_I)} + \left(1 - \frac{T_w}{T(d_I)}\right)\left(\frac{d_B}{d_I}\right)^k \tag{15.19}$$

Adiabatic wall:

$$\frac{T_B}{T(d_I)} = 1 \tag{15.20}$$

The function $f_N(d_I, d_B)$ that scales the normal velocity component in Eq. (15.18) is determined by enforcing a discrete form of the continuity equation at each band cell using a locally parallel flow assumption. A general formulation suitable for compressible flows is given in Ghosh et al. (2010); here, a simpler form suitable for constant-density flows is presented.

$$\begin{aligned}
&f_N(d_I, d_B) = \frac{\left(d_B \big/ d_I\right) d^-}{\left(d_B \big/ d_I\right) d^- + \left(1 - d_B \big/ d_I\right) d^+},\\
&d^- = \left(d_B \big/ 2d_I\right)^k \text{ and } d^+ = \left(1 + d_B \big/ d_I\right)^k \Big/ 2^k.
\end{aligned} \tag{15.21}$$

Note that, this procedure does not rigorously enforce mass conservation within the band cells, as the integral form of the continuity equation is not used. If precise mass conservation is required, the pressure interpolation in Eq. (15.18) can be replaced by the solution of the continuity equation in the band cells. This, however, can lead to oscillations within the band cells, and for some of the moving-body applications presented later, a hybrid approach is utilized. Given that $R_{c,\mathrm{orig}}$ is the initial residual of the continuity equation within a band cell, a modified residual is defined as

$$R_{c,\ \mathrm{mod}} = R_{c,\mathrm{orig}} + C_F \max\left(0, -\sum_k \mathbf{n}_B \cdot \mathbf{n}_k A_k\right) \Delta t^2 \frac{p(d_B) - p(d_I)}{(d_I - d_B)^2} |\mathbf{u}_B \cdot \mathbf{n}_B|. \tag{15.22}$$

This approach (with C_F set to 100) provides additional numerical dissipation within band cells when objects move but reduces to the solution of the continuity equation for non-moving objects.

Interface blocking for zero-thickness immersed surfaces When the continuity equation is solved within band cells, there is a need to identify mesh-cell faces across which mass flow must be restricted ('blocking' interfaces). This is a trivial task for objects that are closed, but for zero-thickness objects, special considerations must be made. To this end, we introduce indices for classifying mesh-cell interfaces as being blocking (no mass transport allowed) versus non-blocking (transport allowed) for zero-thickness immersed objects. The classification of the grid cells in the immersed-boundary (IB) method needs to be robust for any kind of complex immersed surface. Normally, two adjacent triangle elements share one edge; however, the disconnected edges at the boundary of non-closed object only belong to one triangle element. For a given cell's center point $\mathbf{x}_q$, we find the nearest point $\mathbf{x}_s$ on a zero-thickness immersed surface using ANN algorithm (Arya et al. 1998) and then compute the unsigned distance function Φ. Using inner products between the position vector $\mathbf{x}_q$ at the query cell and the position vectors $\mathbf{x}_{nb}$ at adjacent neighboring cells with respect to the nearest point $\mathbf{x}_s$, we can classify the *band* cells for zero-thickness immersed surfaces by detecting a sign change of the inner product; i.e., if $(\mathbf{x}_q - \mathbf{x}_s) \cdot (\mathbf{x}_{nb} - \mathbf{x}_s) < 0$ for $|\Phi| \leq 2\Delta$, then $\mathbf{x}_q$ is *band* cell. Note that Δ is a representative grid resolution.

For an open surface (zero-thickness surface), the blocking index B is only valid for the case that nearest surface points are not on the disconnected edges of the immersed surface from two adjacent cells $\mathbf{x}_i$ and $\mathbf{x}_j$, because the signed distance functions at the cells may not be unique due to the ambiguity of the pseudo-normal vectors at the edges. To avoid the ambiguity, we introduce two incident angles to the parallel direction at the disconnected edges as shown in Fig. 15.5. Let us suppose that the nearest surface points are $\mathbf{x}_s^l$ for the lth immersed object and $\mathbf{x}_s^m$ for the mth immersed object at the cell $\mathbf{x}_i$ and $\mathbf{x}_j$ with the center of the interface $\mathbf{x}_{ij}$, respectively. We can define $B_l(\mathbf{x}_{ij})$ and $B_m(\mathbf{x}_{ij})$ based on the lth and the mth immersed objects, respectively, using the proposed algorithm. For example, if the nearest surface point $\mathbf{x}_s^l$ on the lth immersed object for the cell $\mathbf{x}_i$ is not on a disconnected edge, the blocking

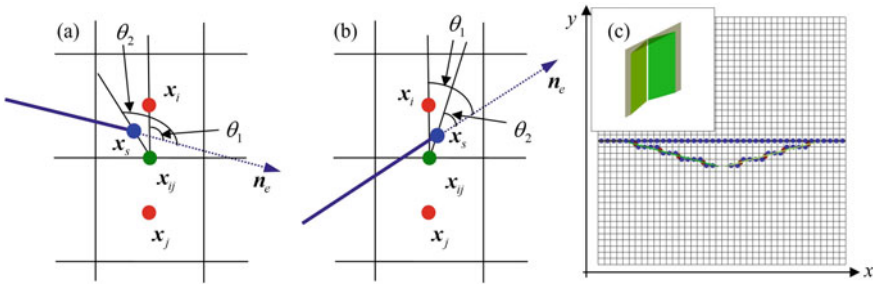


Fig. 15.5 Schematic of classifications for interface blocking, **a** non-blocking, **b** blocking, and **c** blocking index in the vicinity of doors and door frame. Red and blue colored circles represent the blocking index in x and y directions, respectively. Note that, there is no blocking in z direction. The actual doors and door frame are rendered in the inset figure

index $B_l(\mathbf{x}_{ij})$ can be simply determined by an inner product of two position vectors $\mathbf{x}_i - \mathbf{x}_s^l$ and $\mathbf{x}_j - \mathbf{x}_s^l$. However, if the nearest surface point $\mathbf{x}_s^l$ is on a disconnected edge, we need to define two angles θ_1^l and θ_2^l illustrated in Fig. 15.5 for determining the blocking index. The angles are defined as

$$\theta_1^l = \cos^{-1}\left[\left(\left(\mathbf{x}_i - \mathbf{x}_{ij}\right)\cdot \mathbf{n}_e^l\right)/\left(\left|\mathbf{x}_i - \mathbf{x}_{ij}\right|\left|\mathbf{n}_e^l\right|\right)\right] \tag{15.23}$$

$$\theta_2^l = \cos^{-1}\left[\left(\left(\mathbf{x}_s^l - \mathbf{x}_{ij}\right)\cdot \mathbf{n}_e^l\right)/\left(\left|\mathbf{x}_s^l - \mathbf{x}_{ij}\right|\left|\mathbf{n}_e^l\right|\right)\right], \tag{15.24}$$

where $\mathbf{n}_e^l$ is a unit vector that is orthogonal to the line segment of the disconnected edge and the plane involving with the triangle element containing the segment. Thus, the blocking index $B_l(\mathbf{x}_{ij})$ for the lth immersed object can be defined as

$$B_l(\mathbf{x}_{ij}) = \begin{cases} 1, \text{ if } \theta_1^l \geq \theta_2^l & \text{for } \mathbf{x}_s^l \text{ is on a disconnected edge} \\ 1, \text{ if } (\mathbf{x}_i - \mathbf{x}_s^l)\cdot(\mathbf{x}_j - \mathbf{x}_s^l) \leq 0 & \text{for } \mathbf{x}_s^l \text{ is not on a disconnected edge} \\ 0, \text{ otherwise} \end{cases} \tag{15.25}$$

Similarly, we can define $B_m(\mathbf{x}_{ij})$ based on the mth immersed object. Finally, the blocking index $B(\mathbf{x}_{ij})$ for all the immersed objects at the interface $\mathbf{x}_{ij}$ can be defined as

$$B(\mathbf{x}_{ij}) = \begin{cases} 1, \text{ if } B_l(\mathbf{x}_{ij}) + B_m(\mathbf{x}_{ij}) \geq 1 \\ 0, \text{ if } B_l(\mathbf{x}_{ij}) + B_m(\mathbf{x}_{ij}) = 0 \end{cases}, \tag{15.26}$$

Note that $B(\mathbf{x}_{ij})$ indicates that the interface is a *virtual* wall. This means that mass cannot be transferred through the interface and the information at the cell $\mathbf{x}_j$ is excluded in the interpolation stencil for the cell $\mathbf{x}_i$ and vice versa.

15.3 Simulations Involving Human Activity

15.3.1 Problem Definition

The primary use of the developed methodology has been in conducting simulations of realistic human motion, with a specific focus toward capturing induced wake and thermal plume effects on the transport of airborne agents, which can either be gas-phase or particulate in nature. Applications of this capability include entry/exit into shelters designed for collective protection of individuals from harmful agents. Such shelters may use overpressure to inhibit agent transport under static operating conditions and/or airlock systems to remove material that is inevitably transported into the system upon personnel entry. A key to the design of sheltering systems of this type is an understanding of the volume flow of air [normally expressed in cubic feet (CF)] exchanged during an entry event. With this information in place and with knowledge of the agent concentration field, it is possible to predict the mass flow of agent into the shelter.

Such entry events are highly dynamic, involving motion of multiple persons, moving doors, and possibly a transient external flow field. As such, the large-eddy simulation/immersed-boundary methodology described earlier can be used to good effect in capturing the flow physics. The remaining sections describe several applications of this type, along with strategies designed to reduce the output into forms suitable for incorporation into fast-running system performance models.

15.3.2 Agent Transport Due to Thermal Plume and Motion Effects

The first case considered involves simulation of an experiment conducted by Toyon. Incorporated involving tracer-gas transport due to the combined effects of buoyancy (human thermal plume) and wake transport (Juricek 2014). The experimental test chamber (Fig. 15.6) consists of two rooms, a $3 \times 6 \times 8$ ft (L x W x H) antechamber, connected to a second, $12 \times 6 \times 8$ ft main chamber by a swing door (24-in $W \times 70$-in H). Compressed gaseous perfluorocarbon tracer compounds (PDCH and PMCH) mixed in air were released at a flow rate sufficient to ensure detectability. At time $t = 0$, a person initiates the release of the agent and walks from the antechamber into the main chamber, where he stands for 7.5 min. Tracer-gas concentrations (parts per billion) are sampled over one-minute intervals.

Simulation results for the ‘moving’ experiment are presented in Fig. 15.7 for a simulation of 7.5 min in duration and using a 12 M cell mesh. The moving person is rendered as a closed-surface immersed body and is incorporated as a sequence of STL files, generated using 3DSMax® using protocols described earlier. The hinged doors are rendered as zero-thickness immersed objects and are comprised of planar STL files. Rate laws for the door motion are defined in a separate subroutine. The tracer

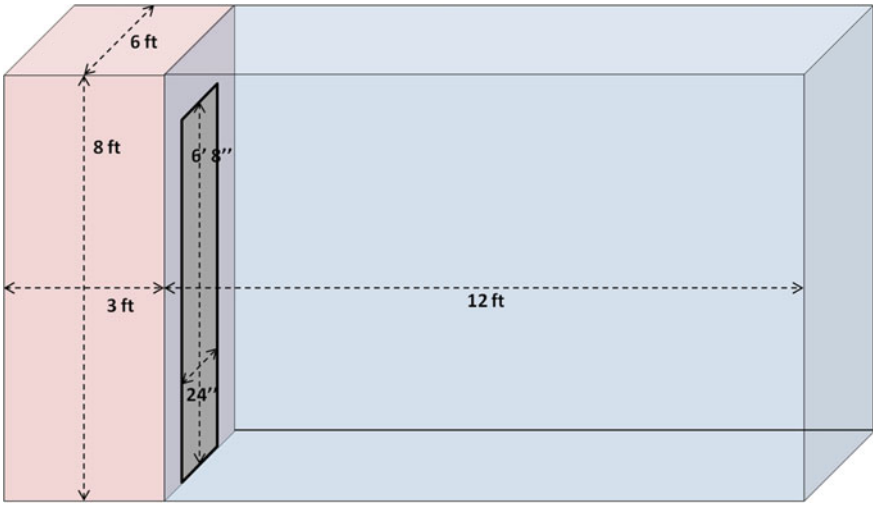


Fig. 15.6 Schematic of room chambers used in simulations

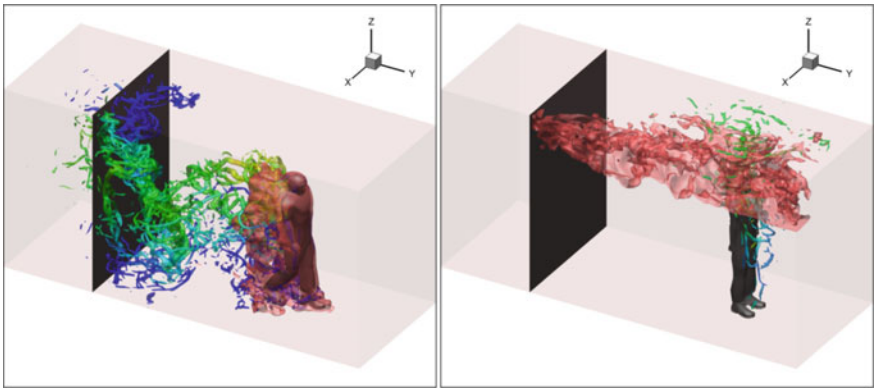


Fig. 15.7 Tracer-gas transport at 3.5 s (left figure) and 300 s (right figure)

gas is 'emitted' from a location under the person's left armpit—this involves the tagging of specific elements of the STL files as mass and momentum sources. In the actual experiment, the person held the tracer-gas emission tube at this same location. A similar approach is used to model human 'breathing' from the nose, though this effect is minor compared to transport due to the thermal plume. At 3.5 s into the event (left component of Fig. 15.7), the person's thermal plume is rendered as a red iso-surface ($T = 304$ K). Iso-surfaces of swirl strength, indicating locations of vortex cores, are colored by tracer-gas concentration. Wakes generated by closing door motion and human walking motion dominate thermal and tracer transport at early

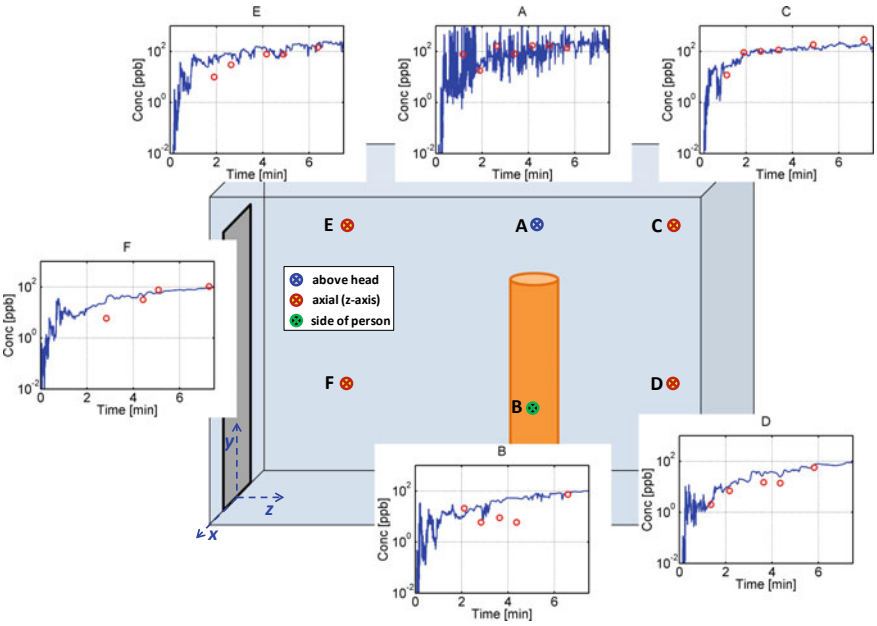


Fig. 15.8 Comparison of predictions with tracer-gas measurements for moving-person entry into the chamber

times. After 5 min (right component of Fig. 15.7), buoyancy-driven flow spreads the tracer-gas plume upward and away from the person.

Quantitative comparisons with experimental gas-sampling measurements are provided in Fig. 15.8. The centermost image shows probe locations within the chamber, while the surrounding images plot agent concentration (ppb) versus time. Probe A is directly above the person, and measurements here are affected both by regular human breathing motion, buoyancy, and (initially) by the decay of velocity fluctuations resulting from the door closing and the person stopping (due to inertia, the wake continues to move forward after the person stops, creating a disturbance field that moves entrained material forward and eventually upward). The predicted concentration levels (sampled at 100 Hz) are very noisy. Filtering the predictions over an interval of 10 s (corresponding to the time required for the gas-sampling syringe pump to operate) reduces the noise significantly. Generally, there is good overall agreement between the simulation and experiment. Probe *C* is further away from the source, and the concentration field in this region is not nearly as intermittent. The predictions are in close agreement with experiment at this location. The general agreement with experiment is reasonable at all probe locations, with some individual samples showing larger differences than others.

15.3.3 Airlock Entry Simulations

The next set of simulations focuses on personnel entry into a multiple-person-entry (MPE) airlock located at the front of a large shelter. These simulations were designed to determine the amount of gas transported into the airlock over the duration of an entry event as a function of the number and arrangement of entering personnel as well as wind speed and wind direction. The computational domain surrounding the shelter and within the interior of the airlock was rendered as a structured, multi-block mesh, with isotropic meshes used in the regions of human activity. Part of the interior of the shelter was also meshed to enable simulations of personnel entering the shelter from the airlock, leading to a total mesh-cell count of 31.3 M. Figure 15.9 shows a wire-frame view of the rendered interior of the complete domain.

Airlock initialization A separate calculation was used to initialize flow within the airlock, which is designed to operate at a target overpressure level. Figure 15.10 (left) shows a side view of the airlock mesh, emphasizing regions of mesh clustering designed to resolve various air jets and exit ports used to facilitate the purging of contaminated gas. Figure 15.10 (right) shows a snapshot of the airlock flow field, highlighting the entering jets of air from the manifold and from the shelter itself, which also operates at an overpressure. In the image, black streamlines emanate from the manifold, while red streamlines emanate from the shelter.

Initialization procedures The external velocity field was initialized using a Pasquill neutrally stable velocity profile. The inputted 'target' velocity for each trial corresponds to the velocity at 2 m above the surface. The inputted flow direction was used to resolve the velocity profile into directions perpendicular to and parallel to the door entrance plane. The orientation is such that 0° corresponds to flow directed into the door, 180° corresponds to flow directed out from the door, and 90° corresponds to flow parallel to the door entrance plane. The simulations were conducted for a fixed

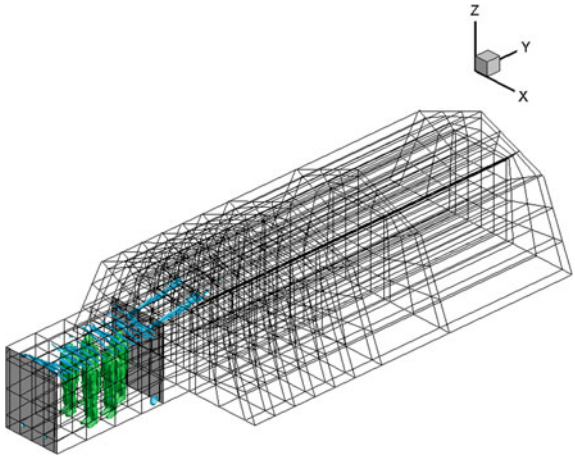


Fig. 15.9 Wire frame rendering of airlock and shelter

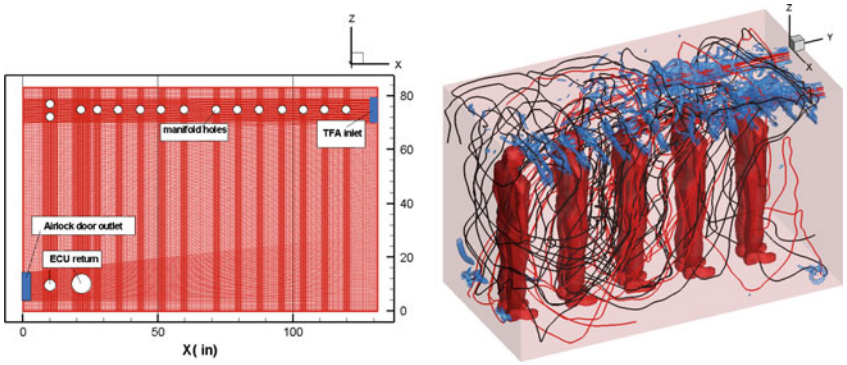


Fig. 15.10 Side view of airlock mesh (left) and snapshot of flow inside airlock (right)

period of time (10 s) prior to the entry event to allow the external flow to stabilize, and discrete wind speeds of 0, 0.8, 1.6, 3.2, 4.8, and 6.0 m/s and discrete wind directions of 0°, 45°, 90°, 135°, and 180° (26 trials, since zero wind speed holds for all directions) were used. Several personnel arrangements were used during the course of the study: single-person entry, five-person single-file entry, four people carrying a patient on a litter, seven-person single-file entry, five-person side-by-side entry, and seven-person side-by-side entry. Animation sequences for each of the entry events were created using 3DSMax®, and the generated sequences of STL files were converted to closed immersed objects using procedures described earlier. The bump-through doors, rendered as planar STL files and containing embedded vents for overpressure control, were 'opened' and 'closed' through the use of specially defined rate laws and were rendered as zero-thickness immersed surfaces.

Five-person, side-by-side airlock entry Figure 15.11 shows snapshots corresponding to the entry of five people side by side into the multi-person airlock. The average walking speed of the group is 1.1 m/s, and the wind speed is zero for this case. Isosurfaces of swirl strength, colored by agent concentration, illustrate the flow patterns generated upon entry. Red contours correspond to a normalized agent concentration of unity, while blue contours correspond to a normalized agent concentration of zero. Frame A corresponds to conditions just prior to entry. Highlighted flow features include air jets entering the airlock from the multi-port manifold and the exiting of air through the door values to maintain the target overpressure. The doors open (Frame B) just prior to entry, leading to an initial expulsion of air in the direction of the entry. A suction pressure is created behind the exiting vortex, allowing flow outside the airlock to migrate into the system. This, combined with the effects of wakes induced by moving personnel, induces net agent transport into the airlock (Frame C). As the doors close, the airlock begins to recover the target overpressure, and flow again emerges from the door vents (Frame D). Figure 15.12 shows a close-up view of wake structures generated as the group makes their way through the released air stream. The time is just after Frame B above; the doors are rendered as transparent to provide a better view of the interior of the airlock. Figure 15.13 (left) plots cubic

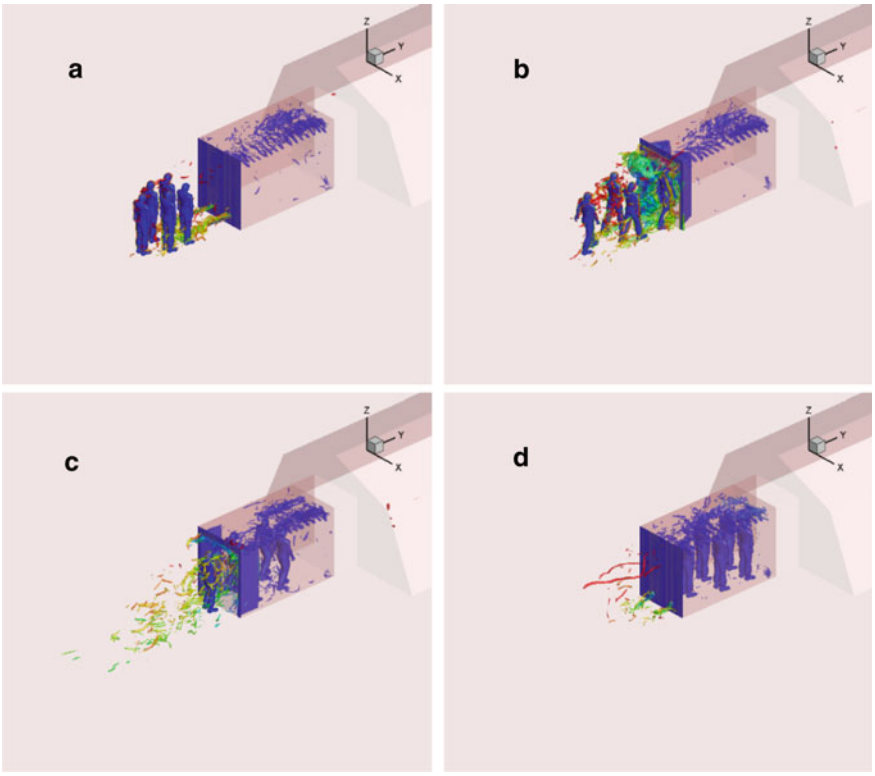


Fig. 15.11 Five-person side-by-side entry into MPE

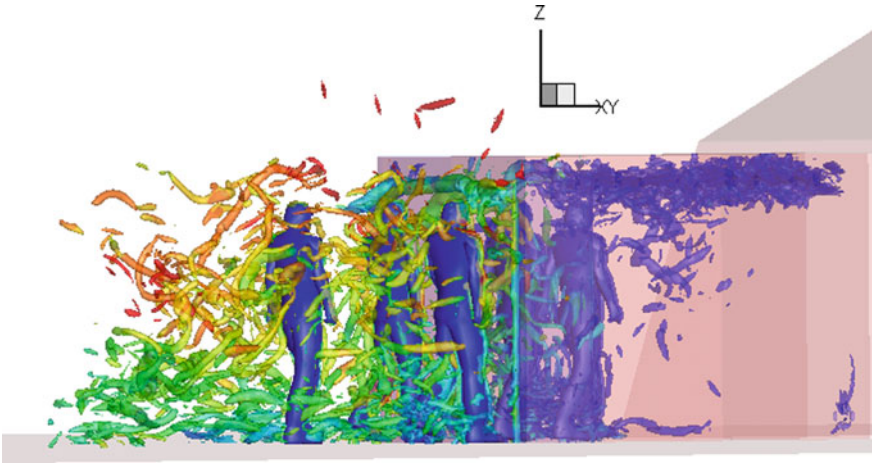


Fig. 15.12 Close-up view of group entering airlock

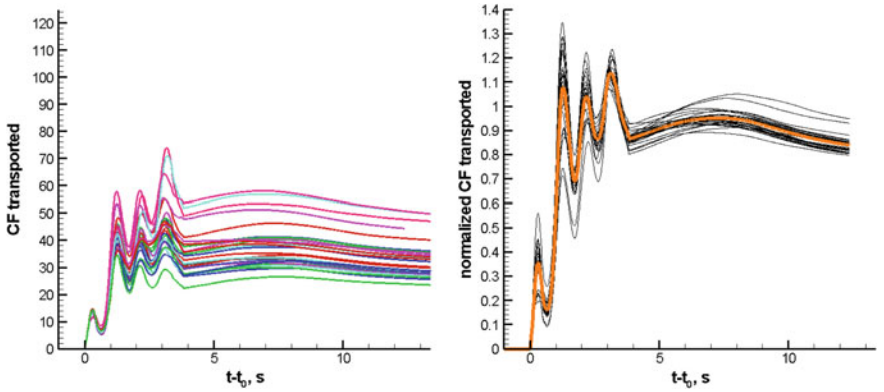


Fig. 15.13 Raw (left) and normalized (right) CF transported versus time

feet of gas transported into the airlock versus time for different wind speed/wind direction combinations. The transport histories are similar to one another and can be effectively collapsed by normalizing by the target CF value at the door closing point (the average of the upper and lower peaks), as shown in Fig. 15.13 (right).

For eventual inclusion into a fast-running system performance model, it is necessary to correlate the target CF transported at the door closing point as a function of wind speed and wind direction. One might expect that wind vectors more aligned with the entry event would enhance transport into the airlock, as would higher wind speeds, but the entry event can also be in the wake of the shelter for wind directions greater than 90°, leading to interactions with vortices shed by the airlock and shelter edges. The dependence is thus not trivial, and our best approach has been to fit the target CF as a function of wind speed and direction angle using a single hidden-layer, ten-node neural network with a sigmoidal activation function:

$$\begin{aligned} \mathrm{CF}_{\mathrm{target}}(V,\theta) &= (c_1 + \sum_{k=1}^{10} b_k h_k)c_2 + c_3 \\ h_k &= \tfrac{1}{1+\exp(-x_k)} \\ x_k &= a_{1,k} + a_{2,k}\tfrac{V-a_{3,k}}{a_{4,k}} + a_{5,k}\tfrac{\theta-a_{6,k}}{a_{7,k}} \end{aligned} \quad , \tag{15.27}$$

Figure 15.14 shows scatter plots of $\mathrm{CF}_{\mathrm{target}}$ predicted by Eq. 15.27 for 50,000 randomly distributed (V, θ) ordered pairs, with V varied from 0 to 6 m/s and θ varied from 0 to 180°. A good coverage of the factor space is indicated, and most of the trial data points lie within the predicted factor space. The average error is 4.20%, and the largest error is around 8%. It is also to be noted that this case shows the expected trends of increased transport into the airlock for higher wind speeds and directions more aligned with the movement of the group. The zero wind speed values represent the effects of wake transport in the absence of wind motion. CF transported generally increases with the number of personnel, but the arrangement also affects transport. A

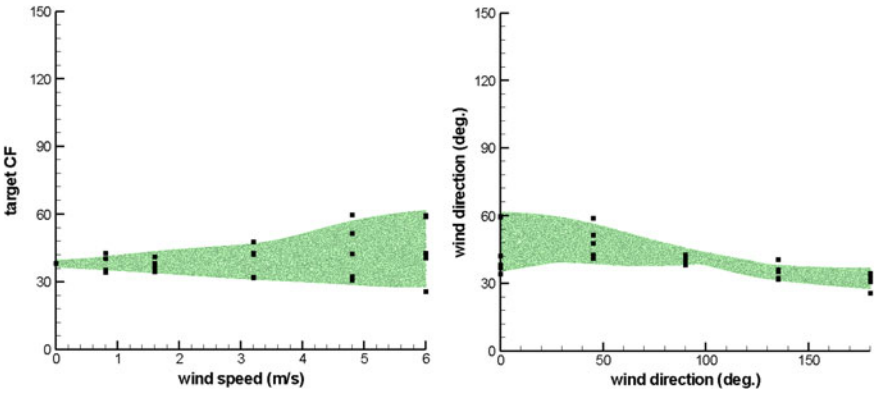


Fig. 15.14 Predicted target CF versus wind speed and wind direction

similar case conducted with five people entering in single file results in nearly twice as much transport at zero wind speed (60 CF vs. 36 CF). This is partially due to the duration of the event, which is ~3.5 s for the side-by-side entry versus 5 s for the single-file entry.

15.3.4 Flow Over a Ruined Building

The last example, while not involving moving entities, illustrates the process of constructing a scenario, creating geometries as sets of STL files, and rendering the objects as closed or zero-thickness immersed bodies. The scenario involves a person buried in rubble releasing a gas-phase taggant to aid in his rescue. A CAD description of a ruined building was obtained from Turbosquid.com (an online retailer for 3D CAD models used in gaming). The building geometry is that of a small house with four small rooms that has collapsed upon itself. The geometry was imported into 3DSMax® and then exported as a binary STL file. This file was then read into Autodesk's NetFabb®, a tool for assembling, repairing, and modifying STL files for use in 3D printing. The STL file for the soldier used in the earlier simulations was added to the scenario, rescaled, and repositioned, so that he was 'trapped' under a portion of the building. The STL files were then exported and pre-processed using the steps described earlier for inclusion as immersed objects in the simulation. The geometry is open to the air above, as shown in Fig. 15.15. The placement of the person and the wind direction is also shown in the figure. The same Pasquill boundary layer used in the shelter simulation was used in this case, which contains about 64 M cells with an isotropic region surrounding the region occupied by the object. The cell size in the isotropic region is 1 in. The person holds the taggant canister and also breathes but otherwise is stationary (his legs are pinned underneath a part of the building).

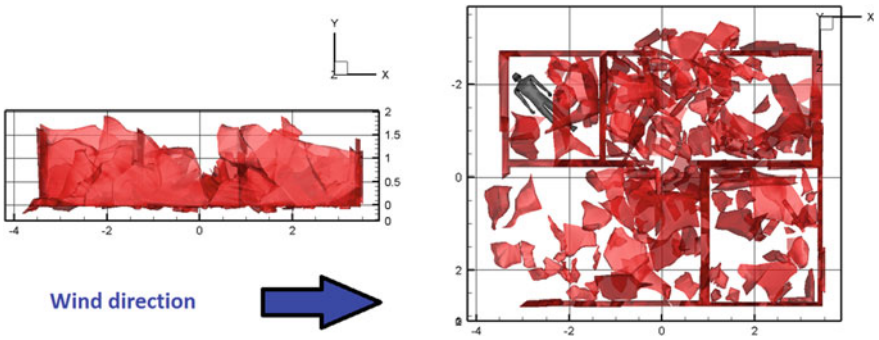


Fig. 15.15 Collapsed building containing injured person

Mass and momentum sources were applied at locations on the person's STL object to mimic taggant release and transient breathing.

Figure 15.16 shows the flow structures that emerge after several transit times. In the left image, an iso-surface of taggant mass fraction (0.0001) is shown colored by velocity magnitude. The image on the right shows iso-surfaces of swirl strength colored by the logarithm of taggant mass fraction. The irregular geometry

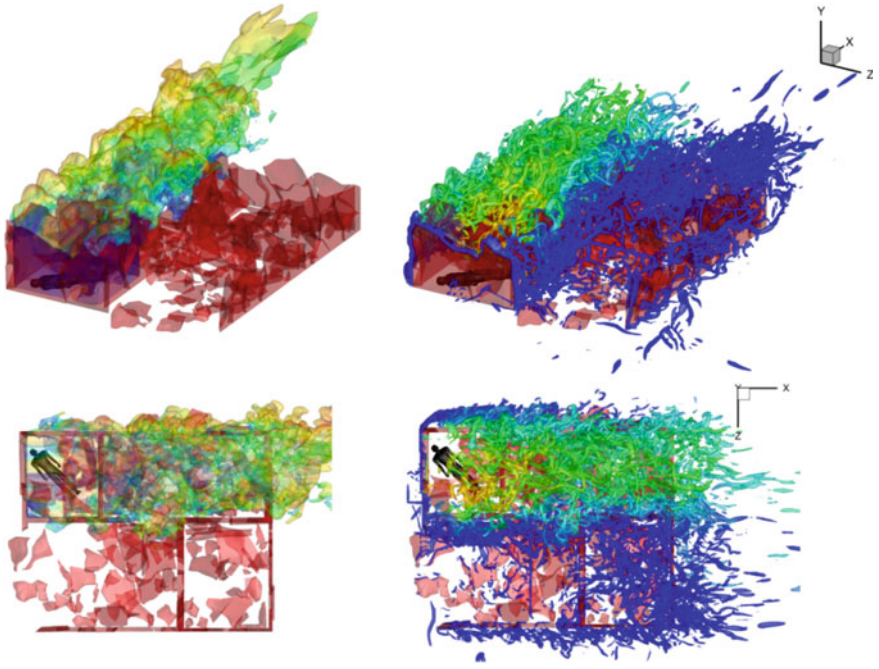


Fig. 15.16 Taggant concentration (left) and swirl strength (right) iso-surfaces: flow over a ruined building—bottom images show top-down views

of the building provides sources for turbulence generation as well as low-momentum regions that may trap fluid. The small enclosure in which the person is placed is one such region—the taggant fills the entire enclosure before being entrained into the external wind field. The breath gas remains within the enclosure, but breathing is a periodic source of effluent—later times would show the expulsion of the breath gas from the enclosure. The fact that the chosen taggant (SF6) is non-buoyant keeps the plume close to the surface.

15.4 Conclusion

An immersed-boundary method suitable for general flow simulations has been presented. The model is grid-topology independent and is based on the decomposition of a computational domain into cells inside an immersed body (field cells), cells outside but adjacent to an immersed body (band cells), and cells far away from an immersed body (field cells). Immersed objects are generated initially as sets of closed-surface or zero-thickness stereo-lithography (STL) files. Procedures for rendering these files as immersed objects within the domain hinge first on splitting such objects into simpler units and secondly on the calculation of the signed distance from each field cell to the embedded surfaces. Interpolation methods based on turbulent boundary layer theory are used to connect the flow solution in band cells to specified surface boundary conditions and to the solution of the Navier–Stokes equations in the field cells. The approach differs from others in the literature in its use of power-law forms for the near-surface velocity, thus enabling the method to mimic the energizing effect of a turbulent boundary layer without excessive near-surface resolution. Applications have been presented for cases involving gas-phase agent transport as induced by human activity (including realistic human motion, breathing, and buoyancy effects due to the human thermal plume) and by other factors, such as an external flow field and moving doors. The combination of large-eddy simulation techniques for capturing wake-induced turbulence and the developed immersed-boundary techniques for representing the effects of stationary and moving objects on the flow evolution provides a powerful framework for conducting realistic simulations of complicated time-dependent flows.

Acknowledgements This work has been supported by the Naval Surface Warfare Center, Dahlgren Division (N001178-08-C-3030) and by Toyon Corporation under a subcontract from the Air Force Research Laboratory (FA8650-13-M-6449).

References

Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1998) An optimal algorithm for approximate nearest-neighbor searching. J ACM 45:891–923

Bærentzen JA, Aanæs H (2005) Signed distance computation using the angle weighted pseudonormal. IEEE T Vis Comp Graph 11(3):243–253

Baurle RA, Tam CJ, Edwards JR, Hassan HA (2003) Hybrid simulation approach for cavity flows: blending, algorithm, and boundary treatment issues. AIAA J 41:1463–1480

Choi J-I, Edwards JR (2008) Large eddy simulation and zonal modeling of human-induced contaminant transport. Indoor Air 18:233–249

Choi J-I, Edwards JR (2012) Large-eddy simulation of human-induced contaminant transport in room compartments. Indoor Air 22:77–87

Choi J-I, Oberoi RC, Edwards JR, Rosati JA (2007) An immersed boundary method for complex incompressible flows. J Comput Phys 224:757–784

Choi J-I, Edwards JR, Rosati JA, Eisner AD (2012) Large eddy simulation of particle re-suspension during a footstep. Aerosol Sci Technol 46(7):767–780

Chorin AJ (1967) A numerical method for solving incompressible Navier-Stokes equations. J Comput Phys 2:12–26

Colella P, Woodward PR (1984) The piecewise parabolic method (PPM) for gas-dynamical simulations. J Comput Phys 54:174–201

Crowe CT, Troutt TR, Chung JN (1996) Numerical models for two-phase turbulent flows. Annu Rev Fluid Mech 28:11–43

Edwards JR, Liou M-S (1998) Low-diffusion flux-splitting methods for flows at all speeds. AIAA J 36:1610–1617

Edwards JR, Choi J-I, Ghosh S, Gieseking DA, Eischen JD (2010) An immersed boundary method for general flow applications. In: FEDSM-ICNMM2010-31097, ASME 2010 3rd joint US-European fluids engineering summer meeting

Fadlun EA, Verzicco R, Orlandi P, Mohd-Yusof J (2000) Combined immersed boundary/finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161:35–60

Ghosh S, Choi J-I, Edwards JR (2010a) Numerical simulation of effects of micro vortex generators using immersed boundary methods. AIAA J 48(1):92–103

Ghosh S, Choi J-I, Edwards JR (2010b) Simulation of shock/boundary layer interactions with bleed using immersed boundary method. J Propul Power 26(2):203–214

Ghosh S, Choi J-I, Edwards JR (2012) Numerical simulation of the effects of mesoflaps in controlling shock/boundary layer interactions. J Propul Power 28(5):955–970

Gilmanov A, Sotiropoulus F, Balaras E (2003) A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids. J Computat Phys 191:660–669

Gouraud H (1971) Continuous shading of curved surfaces. IEEE T Comp 20(6):623–629

Guéziec A (2001) Meshsweeper: dynamic point-to-polygonal-mesh distance and applications. IEEE T Vis Comp Graph 7(1):47–61

Hoff KE, Culver T, Keyser J, Lin M, Manocha D (1999) Fast computation of generalized voronoi diagrams using a graphics hardware. In: Proceedings of the SIGGRAPH'99, pp 277–285

Juricek B et al (2014) Volatile organic compound odor signature modeling. Phase I SBIR Final Report, Air Force Contract FA8650-13-M-6449

Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20:359–392

Linhart J (1990) A quick point-in-polyhedron test. Comput Graph 14(3):445–448

Mittal R, Iaccarino G (2005) Immersed boundary methods. Ann Rev Fluid Mech 37:239–261

Mohd-Yosuf J (1997) Combined immersed boundary/B-spline methods for the simulation of flow in complex geometries, Ann Res Briefs CTR 317–328

Neaves MD, Edwards JR (2006) All-speed time-accurate underwater projectile calculations using a preconditioning algorithm. ASME J Fluids Eng 128:284–296

Oberoi RC, Choi J-I, Edwards JR, Rosati JA, Thornburg J, Rodes CE (2010) Human-induced particle re-suspension in a room. Aerosol Sci Technol 44(3):216–229
Payne BA, Toga AW (1992) Distance field manipulation of surface models. Comp Graph Appl 12(1):65–71
Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10:220–252
Smagorinsky J (1963) General circulation experiments with primitive equations, the basic experiment. Mon Weather Rev 91:99–164
Verzicco R, Mohd-Yusof J, Orlandi P, Haworth D (2000) LES in complex geometries using boundary body forces. AIAA J 38:427–433
Walz A (1969) Boundary layers of flow and temperature (English translation), MIT Press, Cambridge
Wesseling P (1995) Introduction to multigrid methods, NASA CR-195045

Chapter 16
Immersed Boundary Method for High Reynolds Number Compressible Flows Around an Aircraft Configuration

Taro Imamura and Yoshiharu Tamaki

16.1 Introduction

The boundary layer on the surface of a transport aircraft at the cruise condition is almost fully turbulent. The Reynolds number (Re) of the flow based on the main wing chord length is on the order of 10^7 (Wahls 2001; Green and Quest 2011). In addition, modern aircraft have high-aspect-ratio wings and long fuselages that increase their surface area. Thus, the computational costs of a direct numerical simulation or a large eddy simulation (LES) for an external flow around an aircraft are still too high for engineering purposes. Choi and Moin (2012) reported that more than 10^8 cells are required to spatially resolve the flow around a wing whose aspect ratio is 4, even when a wall-modeled LES is used. The simulation also requires many time steps because the time scale of the unsteady turbulent vortices is several orders of magnitude smaller than that of the mean flow. Therefore, the Reynolds-averaged Navier–Stokes (RANS) simulation is widely used for external flows around an aircraft, especially for industrial application. In the derivation of the RANS equation, the temporal fluctuation component and the mean component are decomposed. The computation is carried out only through the mean component, and a steady-state solution is obtained unless strong instabilities (e.g., separated flows behind a bluff body or artificial oscillating motion) exist in the flow field. Under the cruise condition, the flow is mostly attached to the surface; thus, RANS simulations are fairly accurate. For example, in the Drag Prediction Workshops (DPWs) (2017), the RANS simulation capability for an aircraft aerodynamic prediction was widely investigated. These studies (Sclafani et al. 2010, 2013; Lee-Rausch et al. 2014; and Hashimoto

T. Imamura (✉)
The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan
e-mail: imamura@g.ecc.u-tokyo.ac.jp

Y. Tamaki
Tohoku University, 6-6-01 Aramaki-Aza-Aoba, Aoba-ku, Sendai, Japan
e-mail: tamaki@cfd.mech.tohoku.ac.jp

S. Roy et al. (eds.), *Immersed Boundary Method*, Computational Methods in Engineering & the Sciences, https://doi.org/10.1007/978-981-15-3940-4_16

et al. 2014) validated the results on body-fitted grids. The computational grids for RANS simulations are designed to resolve the viscous sublayer of the turbulent boundary layer, using high-aspect-ratio cells that conform to the wall surface. Using the immersed boundary method (IBM) on Cartesian grids for the wall boundary condition, the grids are not aligned to the wall surface (non-body-fitted grids). The cells' aspect ratio near the wall is fixed to unity when Cartesian grid is used which is not suitable for high Reynolds number flow simulations. To resolve the viscous sublayer, many cells are required as compared with that of the typical body-fitted grid. In simple 2D problems, research (Takahashi and Imamura 2014; de Tullio et al. 2007) has proved that turbulent boundary layers can be reproduced when the viscous sublayer is sufficiently resolved. However, simulating 3D turbulent flows using such a fine grid is not realistic. Simulations of flows around high-aspect-ratio wings are quite difficult to perform owing to the uniform cell size requirement in the span-wise direction.

This chapter presents a methodology for simulating a high Reynolds number flow using RANS equation on hierarchical Cartesian grids in combination with IBM. We propose a new approach which applies the modified wall function to IBM. Additionally, a flux-based method is developed based on the balance of the numerical fluxes in order to evaluate the aerodynamic forces.

The remainder of this chapter is organized as follows. Section 2 describes the baseline flow solver using Cartesian grids and the IBM for turbulent flow simulation. A method to calculate the aerodynamic force acting on the immersed bodies is also explained. Section 3 provides numerical results, e.g., turbulent flow over a 2D bump and around an aircraft configuration. Finally, Sect. 4 summarizes the chapter.

16.2 Numerical Methods

16.2.1 The Baseline Grid Generator and Flow Solver (UTCart)

The specification of the baseline flow solver the University of Tokyo Cartesian-grid-based automatic flow solver (UTCart) is described. UTCart consists of two parts: the grid generation and the flow solver. First, the hierarchical Cartesian grid is automatically generated using tree data structures, i.e., the quadtree (2D) or oct-tree (3D). The shapes of input objects are defined by sets of line segments in 2D or by Standard Triangulated Language files (i.e., sets of triangular facet segments) in 3D. Then, binary tree structures and bounding boxes are constructed for each object to search the nearest segments. The cells intersecting the input object are treated as *wall cells*. In addition, the cells inside the object are classified as *body cells*, whereas those outside the object are classified as *fluid cells*. The grid distribution around the object is controlled by the following two options. The first option is to control the numbers of cells in the layers of the same cell size. Figure 16.1 illustrates the

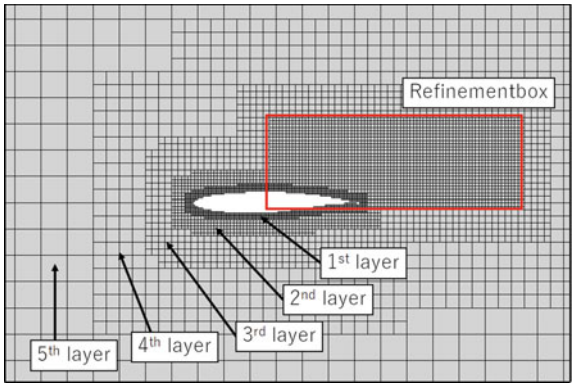


Fig. 16.1 Layers and the *refinement box* of the generated grid

case where the minimum number of cells in each layer is set to 4. The minimum cell size near the wall (the first layer) is doubled after at least four cells of the same size, and this continues to the far-field boundary. The number of cells in each layer is controlled as necessary. The second option is the *refinement box* which is used to refine uniformly a certain area of the computational domain. The *refinement box* is specified by the minimum/maximum coordinates of the rectangular (2D) or cuboid (3D) and the uniform cell size inside. After the generation of the hierarchical Cartesian grid, the grid is partitioned using the METIS library (2019) for a parallel computation based on the message passing interface. In each divided grid domain, sleeve cells are specified for the communication between the domains.

In the second step, a flow calculation is performed. The numerical methods in the solver are summarized in Table 16.1. The flow simulation by UTCart is based on the compressible Euler/Navier–Stokes equations in a conservation form. For high Reynolds number flows, RANS simulations are carried out using a turbulence model. The governing equations are as follows:

$$\frac{\partial \boldsymbol{Q}}{\partial t} + \frac{\partial \left(\boldsymbol{F}_j - \boldsymbol{F}_{V,j}\right)}{\partial x_j} = 0, \tag{16.1}$$

where $\boldsymbol{Q} = [\rho, \rho u_i, \rho E]^{\mathrm{T}}$ is the vector of conservative variables. $\boldsymbol{F}_j = \left[\rho u_j, \rho u_i u_j + p\delta_{ij}, (\rho E + p)u_j\right]^{\mathrm{T}}$ is inviscid flux, and $\boldsymbol{F}_{V,j} = \left[0, \tau_{ij}, \tau_{jk}u_k - q_j\right]$ denotes viscous flux ($i, j, k = 1, 2$ for 2D, and $i, j, k = 1, 2, 3$ for 3D). Here, ρ is the density, u_i is the velocity, E is the total energy per unit mass, τ_{ij} is the viscous stress tensor, and q_j is the heat flux. The ideal gas law for relating the thermal quantities is

$$p = \rho RT, E = \frac{p}{\rho(\gamma - 1)} + \frac{1}{2}u_k u_k, \tag{16.2}$$

where T is the temperature, R is the gas constant, and $\gamma = 1.4$ is the ratio of the specific heat. The viscous stress tensor and the heat flux are approximated as

Table 16.1 Numerical methods for UTCart

Governing equations	Compressible Euler equations
	Compressible Navier–Stokes equations
	Compressible Reynolds-averaged Navier–Stokes equations
Turbulence model	Spalart–Allmaras (SA-noft2)
Discretization method	Cell-centered finite volume method
Type of grids	Unstructured hierarchical Cartesian grids
Inviscid flux	SLAU with third-order MUSCL
Limiter	Minmod or van Albada
Viscous flux	Second order
Convective and diffusive flux of SA	Second order
Gradient evaluation	WLSQ (G)
Time integration method	MFGS or LU-SGS (Yoon and Jameson 1988)
Time-stepping method	Local time-stepping method

$$\tau_{ij} = 2(\mu + \mu_t)\left[S_{ij} - \frac{1}{3}S_{kk}\delta_{ij}\right],\ q_j = -c_p\left(\frac{\mu}{Pr} + \frac{\mu_t}{Pr_t}\right)\frac{\partial T}{\partial x_j},$$

where μ is the molecular viscosity, μ_t is the eddy viscosity, $S_{ij} = 1/2\left(\partial u_j/\partial x_i + \partial u_i/\partial x_j\right)$, and $c_p = \gamma/(\gamma - 1)R$ is the specific heat at constant pressure. Prandtl number Pr is set to 0.72, and turbulent Prandtl number Pr_t is set to 0.9. When the eddy viscosity μ_t is set to 0, Eq. (16.1) becomes the Navier–Stokes equations. In Euler calculations, the molecular viscosity μ is additionally set to 0. Spalart–Allmaras one-equation turbulence model (SA) (Spalart and Allmaras 1992) calculates the eddy viscosity. The version of SA used in this research is an SA-noft2 model (Turbulence Modeling Resource 2019), which neglects the f_{t2} term. The equations of SA-noft2 are as follows:

$$\frac{\partial}{\partial t}(\widetilde{\nu}) + u_i\frac{\partial}{\partial x_i}(\widetilde{\nu}) = \frac{1}{\sigma}\left[\frac{\partial}{\partial x_i}\left((\nu + \widetilde{\nu})\frac{\partial \widetilde{\nu}}{\partial x_i}\right) + c_{b2}\frac{\partial \widetilde{\nu}}{\partial x_i}\frac{\partial \widetilde{\nu}}{\partial x_i}\right] + c_{b1}\widetilde{S}\widetilde{\nu} - \mathrm{c_{w1}}f_w\left(\frac{\widetilde{\nu}}{d}\right)^2, \tag{16.3}$$

$$\mu_t = \rho\widetilde{\nu}f_{v1},\ f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3},\ \chi = \frac{\widetilde{\nu}}{\nu},\ \widetilde{S} = \Omega + \frac{\widetilde{\nu}}{\kappa^2 d^2},\ \Omega = \sqrt{2W_{ij}W_{ij}},$$

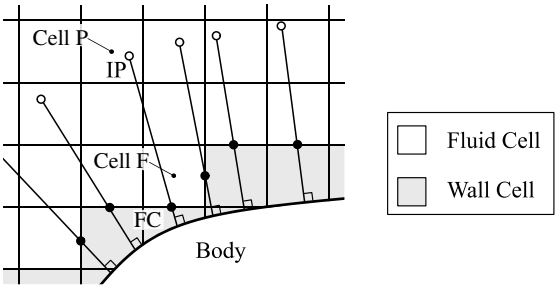


Fig. 16.2 Schematic of the wall boundary condition of the immersed surface

$$W_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right), f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, f_w = g\left[\frac{1+c_{w3}^6}{g^6+c_{w3}^6}\right]^{\frac{1}{6}},$$

$$g = r + c_{w2}\left(r^6 - r\right), r = \min\left[\frac{\widetilde{\nu}}{\widetilde{S}\kappa^2 d^2}, 10\right],$$

where d denotes the distance from the local point x_i to the nearest point on the wall surface. The closure constants are

$$c_{b1} = 0.1355, \sigma = 2/3, c_{b2} = 0.622, \kappa = 0.41,$$

$$c_{w1} = c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma, c_{w2} = 0.3, c_{w3} = 2, \text{ and } c_{v1} = 7.1.$$

The governing equations are discretized by the cell-centered finite volume method. The hierarchical Cartesian grids are treated as unstructured data structure. The inviscid flux is evaluated using the simple low-dissipation advection upstream splitting method (AUSM) scheme (Shima and Kitamura 2011). The third-order monotonic upwind scheme for conservation laws (MUSCL) is used to increase the spatial accuracy. The viscous flux is calculated using a modified second-order central difference (Wang et al. 2010). The accuracy of the convective and diffusive flux of the SA model is second order. Gradients of the primitive variables are calculated using the weighted least square method (WLSQ) (Shima et al. 2013). The matrix-free Gauss–Seidel, which is an implicit time integration method, is used for the time integration (Shima 1997). All the numerical computations are steady; thus, a local time-stepping method is introduced to accelerate convergence. The subsonic far-field boundary conditions are determined based on the method proposed by Chakravarthy and Osher (1983).

16.2.2 Immersed Boundary Method for UTCart

Figure 16.2 is the schematic of the grid near the wall boundary. The cells intersecting the body surface are the *wall cell*, and the cell completely inside the fluid domain is the *fluid cell*. UTCart imposes the wall boundary condition at the center between *fluid cell* and the *wall cell* [point face center (FC)]. A discrete-forcing IBM is used to determine the boundary conditions. Here, the IBM for inviscid and low Reynolds number viscous flows is explained. The IBM for high Reynolds number flow is described in the next subsection.

To determine the physical quantities of FC, an image point (IP) is set on the wall-normal line through FC, assuming one-dimensional variable profiles between the IP and the wall. The distance between the IP and the wall is d_{IP} related to the size of the ambient cells Δx by

$$d_{\mathrm{IP}} = r_{\mathrm{IP}} \Delta x, \tag{16.4}$$

where r_{IP} is the ratio of the IP distance to the cell size on the wall, which is a constant value. The minimum value for r_{IP} is $\sqrt{2}$ in 2D and $\sqrt{3}$ in 3D for the IPs to be located in the *fluid cells*. Typically, the r_{IP} value is set to 2–3. An exception may occur where two walls are located close to each other. If IP is located in the wall, the wall boundary is considered to be a step-wise face, and the value at FC is determined using the value at the *fluid cell* including the FC to avoid a numerical problem.

In the explanation below, the quantities at the IP and FC are represented by subscripts IP and FC, respectively. The primitive variables $\boldsymbol{q}$ at the IP is linearly interpolated locally inside the cell as

$$\boldsymbol{q}_{\mathrm{IP}} = \boldsymbol{q}_P + \left.\frac{\partial \boldsymbol{q}}{\partial x_j}\right|_P \left(x_{j,\mathrm{IP}} - x_{j,P}\right), \tag{16.5}$$

where the subscript P denotes the value at the center of the cell including the IP. Then, the primitive variables at FC are calculated using the quantities at the IP. For example, the pressure is assumed to satisfy the zero-gradient condition on the wall. The wall-normal velocity must satisfy the non-penetration condition, where the normal velocity is zero on the wall. Thus, a linear profile between the IP and the wall is assumed. The boundary condition for the tangential velocity u_t depends on whether the wall is slip or non-slip. The numerical flux at FC is calculated using the primitive variables at FC. An upwind scheme calculates the inviscid flux. The viscous flux is calculated using only the quantities at FC assuming the adiabatic wall boundary condition for the heat flux.

During the grid partitioning for the parallel flow computation, a modification is applied to the list of sleeve cells when IBM is used (Imamura et al. 2017). As illustrated in Fig. 16.2, physical quantities at the IP are used to define the wall boundary condition at the FC which is an interface between the *fluid cell* and *wall cell*. Extra communication is required if IP and FC are located in different domains.

16.2.3 Wall Function for RANS

A wall boundary condition for UTCart to simulate turbulent flows is presented. The SA wall model developed by Allmaras et al. (2012) is used to evaluate the effect of the neglected molecular viscosity and construct a universal law of the wall. This wall velocity model is derived under the assumption for the law of the wall analysis: incompressible, zero pressure gradient, constant outer edge velocity, ignore advection terms, and gradient terms parallel to the wall. The shape of this function is presented in Fig. 16.3.

$$u^+ = f_{\mathrm{SA}}\left(y^+\right), \tag{16.6}$$

where u^+ and y^+ are the normalized tangential velocity using wall friction velocity u_τ and distance in the wall unit, respectively. By substituting the tangential velocity at IP in Eq. (16.6), Newton's iteration is performed to obtain u_τ. Then, the tangential velocity at FC is calculated as

$$u_{t,\mathrm{FC}} = u_\tau f_{\mathrm{SA}}\left(y^+_{\mathrm{FC}}\right). \tag{16.7}$$

Furthermore, the temperature at FC is calculated by the Crocco–Busemann relationship (White 2006):

$$T_{FC} = T_{\mathrm{IP}} + \frac{\mathrm{Pr}^{1/3}}{2c_p}\left(u^2_{t,\mathrm{IP}} - u^2_{t,\mathrm{FC}}\right). \tag{16.8}$$

Then, the density at FC is calculated as:

$$\rho_{\mathrm{FC}} = \frac{p_{\mathrm{FC}}}{RT_{\mathrm{FC}}}. \tag{16.9}$$

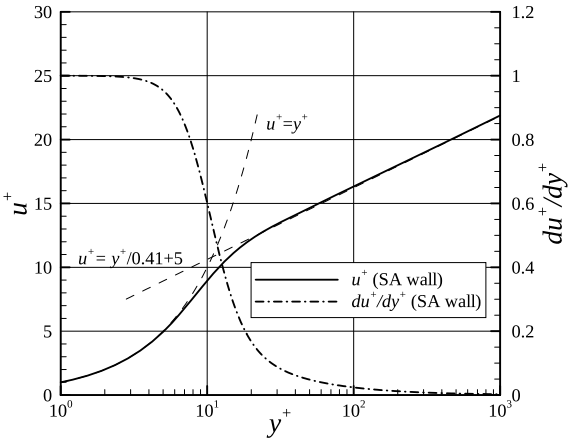


Fig. 16.3 SA wall model developed by Allmaras et al. (2012)

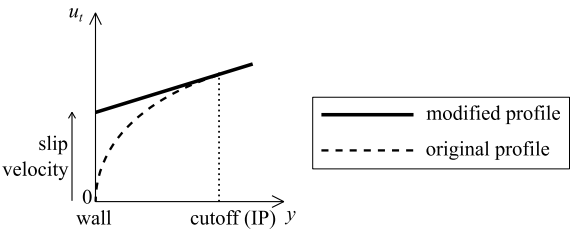


Fig. 16.4 Modification of the velocity profile

The velocity profile defined by the wall function in Eq. (16.6) is a nonlinear function. However, a spatial scheme with second-order accuracy reconstructs a linear (first-order polynomial) profile within a cell. As a result, the velocity profile assumed by the wall function cannot be reproduced in the cell. To overcome this problem, the velocity profile and related flow variables should be modified. This idea originates from Capizzano (2011). The tangential velocity profile is modified using the first derivative of the SA wall model:

$$f_{\mathrm{SA,mod}}\left(y^{+}\right) = u^{+}_{\mathrm{IP}} + \left.\frac{\mathrm{d}f_{\mathrm{SA}}}{\mathrm{d}y^{+}}\right|_{\mathrm{IP}} \left(y^{+}_{\mathrm{IP}} - y^{+}\right). \tag{16.10}$$

The inviscid flux on the face is calculated using the tangential velocity $u^{+}\left(y^{+}_{\mathrm{FC}}\right)$ obtained by Eq. (16.10). In this velocity profile (Fig. 16.4), the tangential velocity at $y^{+} = 0$ is nonzero; thus, a virtual slip velocity is imposed on the wall. Note that the viscous flux on the face is directly calculated as $\tau_{\mathrm{FC}} = \rho_{\mathrm{FC}} u_{\tau}^{2}$.

Along with the velocity profile modification, it is important to maintain the balance of the shear stress,

$$(\nu + \nu_t)\frac{\mathrm{d}u}{\mathrm{d}y} = \frac{\tau_w}{\rho}, \tag{16.11}$$

where τ_w is the wall shear stress. Note that Eq. (16.11) is an approximate relationship in the inner layer of the boundary layer, where the convection and pressure gradients are negligible. Thus, a modification is required on the eddy viscosity profile corresponding to the modification of the velocity profile. In the modified velocity profile of Eq. (16.10), the velocity gradient $(\mathrm{d}u/\mathrm{d}y)$ is constant. Accordingly, ν_t must be constant in the region between the IP and the wall to maintain the constant shear stress implied by Eq. (16.11). Here, the near-wall solution of $\widetilde{\nu}$ is retained, and only the wall-damping function f_{v1} in Eq. (16.3) is modified to avoid additional complexity. To realize the constant profile of the eddy viscosity, the wall-damping function must be

$$f_{v1} \sim \frac{1}{d}, \tag{16.12}$$

because of the near-wall solution of $\widetilde{\nu}$ is proportional to the wall distance d. For the implementation, the profile of f_{v1} must be continuous. Thus, the wall-damping

function is redefined as

$$f_{v1} = \begin{cases} f_{v1,\text{original}} (d \geq d_{\text{cutoff}}) \\ f_{v1,\text{near-wall}} (d < d_{\text{cutoff}}) \end{cases}. \tag{16.13}$$

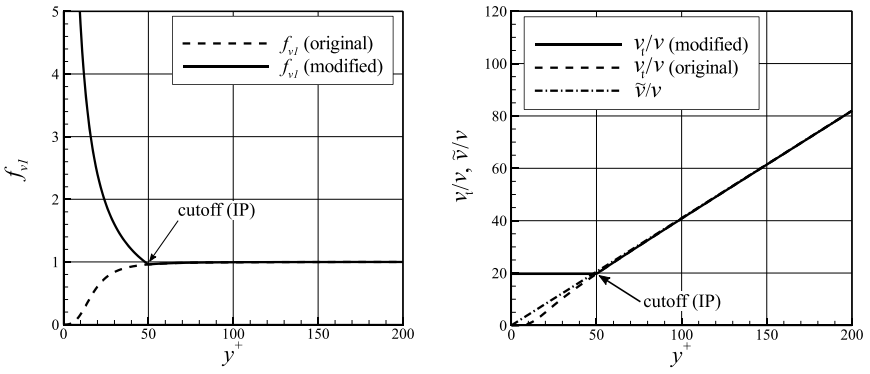
Cutoff distance d_{cutoff} is equal to the distance between the IP and the wall, d_{IP}. The original damping function f_{v1} is

$$f_{v1,\text{original}} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \tag{16.14}$$

and $f_{v1,\text{near-wall}}$ is a modified damping function near the wall defined as

$$f_{v1,\text{near-wall}} = r_d \frac{(\chi r_d)^3}{(\chi r_d)^3 + c_{v1}^3}, \tag{16.16}$$

where $r_d = d_{\text{cutoff}}/d$. Note that $f_{v1,\text{near-wall}}$ is a product of r_d and the original f_{v1} at $d = d_{\text{cutoff}}$. When the IP is located in the log layer of the turbulent boundary layer, $f_{v1,\text{near-wall}}$ is approximately equal to r_d. This function depends on the relative position of the IP in the boundary layer. For example, the shape of the function with $y_{\text{IP}}^+ = 50$ is illustrated in Fig. 16.5. The modified eddy viscosity profile has a kink at the cutoff point. The following technique is used to calculate the viscous fluxes on the faces. Here, face lr is considered, which is the face between cells l and r. The eddy viscosity on the face is required to calculate the viscous flux on faces l, r, and $\nu_{t,lr}$. However, the simple average of $\nu_{t,l}$ and $\nu_{t,r}$ is different from the true value of the profile if the kink exists between cells l and r. This may cause numerical errors. Thus, the following procedure is adopted to eliminate the effect of the kink. The averages of left and right cells for $\widetilde{\nu}$, ν and d are calculated as follows:



(a) Wall-damping function, f_{v1} (b) Eddy viscosity

Fig. 16.5 Modification of the eddy viscosity profile

$$\begin{aligned}\widetilde{\nu}_{lr} &= r_{lr}\widetilde{\nu}_l + (1 - r_{lr})\widetilde{\nu}_r,\\ \nu_{lr} &= r_{lr}\nu_l + (1 - r_{lr})\nu_r,\\ d_{lr} &= r_{lr}d_l + (1 - r_{lr})d_r,\end{aligned} \tag{16.17}$$

where r_{lr} is the ratio of the cell sizes:

$$r_{lr} = \frac{\Delta x_r}{\Delta x_l + \Delta x_r}. \tag{16.18}$$

The eddy viscosity is calculated by those quantities:

$$\nu_t|_{lr} = \widetilde{\nu}_{lr} f_{v1}(\chi_{lr}, d_{lr}), \tag{16.19}$$

where $\chi_{lr} = \widetilde{\nu}_{lr}/\nu_{lr}$. The ν profile is nearly linear near the wall, and the numerical error is smaller than the simple average of ν_t.

Corresponding to the modification of the velocity and eddy viscosity profiles, the thermal boundary condition has now been reconsidered. The Crocco–Busemann relationship in Eq. (16.8) is differentiated in terms of wall-normal coordinate yields:

$$\frac{\mathrm{d}T}{\mathrm{d}y} = \frac{\mathrm{Pr}^{1/3}}{c_p} u_t \frac{\mathrm{d}u_t}{\mathrm{d}y}. \tag{16.20}$$

In the modified velocity profile in Eq. (16.10), the normal gradient of the tangent velocity is constant below the IP. Here, u_τ is assumed to be nearly constant because the velocity gradient in the log layer is small. Thus, the temperature gradient is nearly constant below the IP, and the temperature profile becomes a linear profile:

$$T_{\mathrm{FC}} = T_{\mathrm{IP}} - \left.\frac{\mathrm{d}T}{\mathrm{d}y}\right|_{\mathrm{IP}} (y_{\mathrm{IP}} - y_{\mathrm{FC}}), \tag{16.21}$$

where the temperature gradient at IP is calculated in Eq. (16.20).

The proposed method is thoroughly tested through the simulations of the flat-plate turbulent boundary layer. Further details are discussed by Tamaki et al. (2017) and Tamaki (2018).

16.2.4 Force Calculation Method

To compute the aerodynamic force, the *polygon-based method* (Nonomura and Onishi 2017) which integrates over the input CAD surface is often used. In this method, the physical quantities (e.g., pressure) on the Cartesian grid are interpolated and/or extrapolated onto the CAD surface before the integration. This method is the same as the force integration method of conventional body-fitted grids, except for

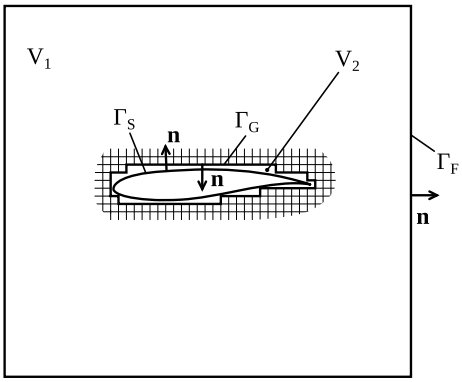


Fig. 16.6 Description of the computational domain and the boundaries

the interpolation and/or extrapolation. However, arbitrariness exists in the interpolation and/or extrapolation formula; thus, the computed forces may contain additional numerical errors. In addition, the integration accuracy depends on the resolution of the CAD surface. Therefore, the calculation of force acting on the immersed body based on the flow solution needs to be explored.

To remove the uncertainties related to the previous discussion, new force integration is developed based on the balance of the numerical flux. This idea is similar to the far-field methods (van Dam 1999; Kusunose and Crowder 2002); however, the integration surface is the step-wise cell boundary between the *fluid cell* and the *wall cell*. Unlike the far-field method, the pressure and viscous component of the force are calculated using this new method because the integration surface is near the object surface. The force can also be decomposed when multiple objects exist in the computational domain.

As illustrated in Fig. 16.6, an immersed body Γ_S in Cartesian grids is considered. The step-wise cell boundary near the wall and the far-field boundary are named Γ_G and Γ_F, respectively. Note that the normal vectors of Γ_G and Γ_F are pointing outside the computational domain. Furthermore, the domain between Γ_G and Γ_F and that between Γ_G and Γ_S are named V_1 and V_2, respectively. To perform component-wise integration of the aerodynamic force, the integral over Γ_F is replaced by that over Γ_G. The momentum equation is integrated over domain V_1 assuming neither mass source nor body force exists in the domain. The near-field integration formula for the aerodynamic force is described as follows:

$$F_i = \int_{\Gamma_G} \left\{ \rho\left(u_i - U_{\infty,i}\right)u_j + (p - p_\infty)\delta_{ij} - \tau_{ij} \right\} n_j \mathrm{d}S. \tag{16.22}$$

Equation (16.22) is discretized on the faces that compose Γ_G:

$$F_i = \sum_{\text{face}\in\Gamma_G} \left[\left\langle \left(\rho u_i u_j + p\delta_{ij}\right)\hat{n}_j \right\rangle - \left\langle \rho u_j \hat{n}_j \right\rangle U_{\infty,i} - p_\infty \delta_{ij}\hat{n}_j - \left\langle \tau_{ij}\hat{n}_j \right\rangle \right]_{\text{face}}, \tag{16.23}$$

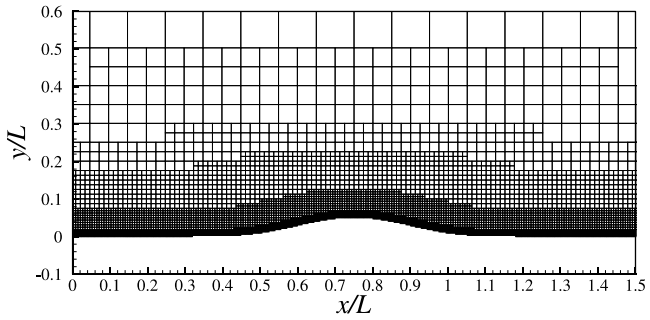
where $\langle(\rho u_i u_j + p\delta_{ij})\hat{n}_j\rangle$ are the momentum components of the inviscid flux, $\langle\rho u_j \hat{n}_j\rangle$ is the mass component of inviscid flux, and $\langle\tau_{ij}\hat{n}_j\rangle$ is the momentum components of the viscous flux. Note that $\hat{n}_j$ is the normal vector component of the faces on Γ_G. Here, the integral of the viscous flux is considered to be the viscous component of the aerodynamic force, and the remainder is considered to be pressure component. The aerodynamic forces acting on each part of the immersed body (or each object) can be decomposed when the faces are classified with respect to the nearest part or object. Thus, it is suggested that one uses the same inviscid and viscous numerical fluxes as those in the flow calculation of the flux components in Eq. (16.23). The evaluated force directly reflects the accuracy of the flux used in the flow calculation, and no additional numerical error is produced.

16.3 Numerical Results

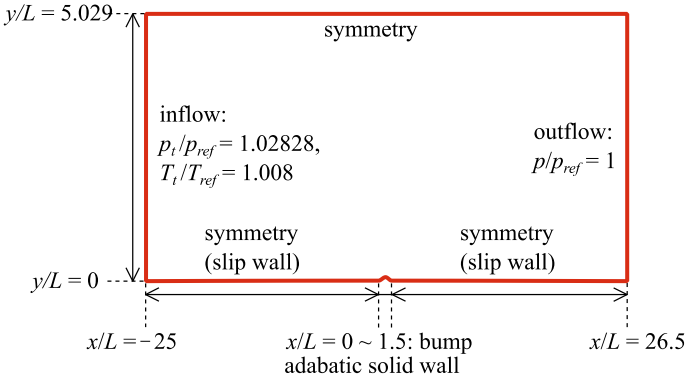
16.3.1 Subsonic Flow Over a 2D Bump

The first test case is the subsonic flow over a 2D bump defined in the NASA Turbulence Model Resource (TMR) (2019). The effect of the stream-wise pressure gradient is small compared to that of viscous force, except for the location close to a separation point (Tennekes and Lumley 1972). In this problem, the validity of the proposed IBM is investigated in a flow with a mild pressure gradient. This is because the effect of the stream-wise pressure gradient is neglected in the baseline, which is an approximated governing equation for the proposed IBM. The Reynolds number based on reference length L, and the free-stream Mach number of 0.2 is 3×10^6, and the free-stream temperature is 300 K. The overview of the grid and the boundary conditions are illustrated in Fig. 16.7. Five grids with different grid resolutions are prepared to check the trend of grid convergence as tabulated in Table 16.2. In addition, r_{IP} is fixed to 3 for this problem. CFL3D (2019) computes the reference result on the 1409 $\times$ 641 grid. These reference computational results are also provided in the TMR. The y^+_{IP} in Table 16.2 is estimated by c_f of this reference result. The results of the original IBM and modified IBM are compared to clarify the importance of the modification proposed in Sect. 2.3. The specification of these methods is summarized in Table 16.3.

The distributions of the pressure and skin friction coefficients on the bump are illustrated in Figs. 16.8 and 16.9, respectively. The reference result by CFL3D is also illustrated in the same figures. On one hand, a large oscillation is observed on the pressure coefficient C_p in the original IBM results, and the skin friction deviates from the reference result. This trend is obvious in the fine grids; the result in grid 5 predicts the peak of c_f at a different location, and the magnitude of c_f is approximately 30% smaller than the reference result. As a result, no trend of grid convergence is observed in the original IBM results. However, the modified IBM reproduces the distribution of c_f more accurately. The oscillation of C_p is smaller than the original IBM result,



(a) Computational grid



(b) Boundary conditions

Fig. 16.7 Computational grid over the bump

Table 16.2 Setting of computational grids over the 2D bump

Grid	Min. cell size	Number of cells	y^+_{IP} at $x/L = 0.75$ (estimation)
1	1.57×10^{-3}	21,762	784
2	7.86×10^{-4}	43,246	392
3	3.93×10^{-4}	82,978	196
4	1.96×10^{-4}	164,638	98.0
5	9.82×10^{-5}	325,698	49.0

Table 16.3 Specification of the original and modified IBMs

	Original IBM	Modified IBM
Velocity profile	SA wall model, Eq. (16.6)	Linear, Eq. (16.10)
f_{v1}	Original definition in SA, Eq. (16.3)	Modified, Eq. (16.13)
Temperature profile	Crocco–Busemann relationship, Eq. (16.8)	Linear, Eq. (16.21)

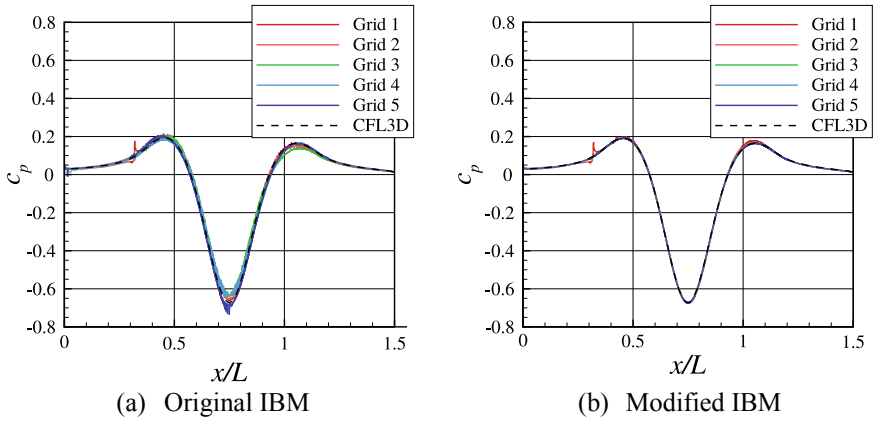


Fig. 16.8 Distribution of the pressure coefficient on the bump

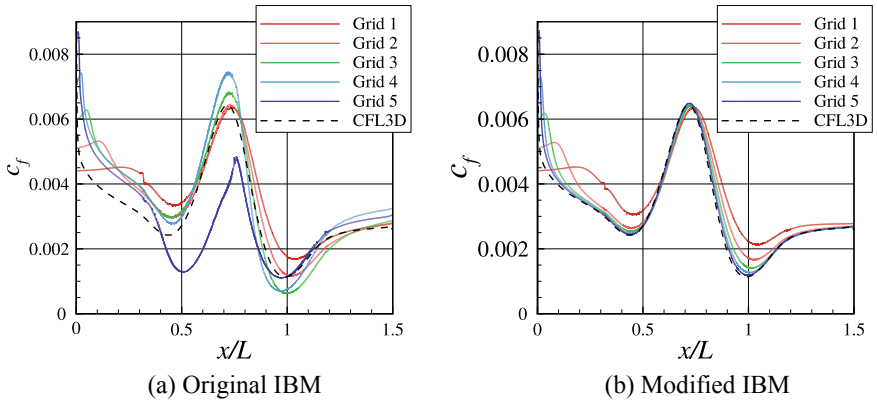


Fig. 16.9 Distribution of the skin friction coefficient on the bump

and the magnitude of C_p is also more accurate. In addition, the skin friction on the finer grids has better agreement with the reference result; thus, a correct grid convergence trend toward the reference result is confirmed. Therefore, the modified IBM can reproduce this flow with a certain degree of accuracy.

16.3.2 Flow Analysis Around the NASA Common Research Model

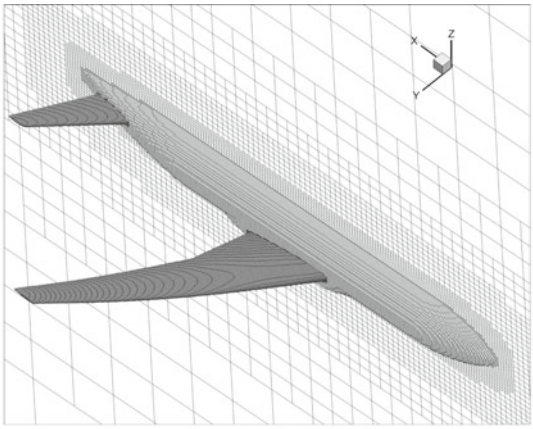
To investigate the capability of the proposed framework for aerodynamic prediction on a civil transport aircraft, transonic flows around the NASA common research model (CRM) (Vassberg et al. 2008) are simulated (Tamaki 2018; Tamaki

and Imamura 2018). The NASA CRM was developed as a benchmark in the DPWs (2017). This geometry is widely tested in wind tunnel experiments (Rivers and Dittberner 2014; Ueno et al. 2014) and in numerical simulations (Sclafani et al. 2010, 2013; Lee-Rausch et al. 2014; Hashimoto et al. 2014; Yamamoto et al. 2012; Vassberg et al. 2014; and Tinoco et al. 2017). Using CFD simulations, a domestic workshop in Japan, the Aerodynamic Prediction Challenge (APC) (2019) workshop, was held recently to investigate the accuracy of the aerodynamic prediction of the NASA CRM. The geometry tested in this workshop consists of a fuselage, main wings, and horizontal tails with the incident angle of attack of $i_H = 0°$. The calculation setting in this section is adjusted to the condition of the experiment (Ueno et al. 2014) in Japan Aerospace Exploration Agency (JAXA) transonic wind tunnel, using a 2.16% scale model (the mean aerodynamic chord $c_{\text{ref}} = 151.31$ mm). The free-stream Mach number is 0.847; the free-stream temperature is 284 K; and the Reynolds number based on the mean aerodynamic chord is 2.26×10^6. The angles of attack are from $-1.79°$ to $5.72°$. In the wind tunnel experiment, the wing is deformed by the aerodynamic force acting on it (Tinoco et al. 2017). The geometry used in this simulation is also deformed based on the experimental data. The deformation (twist and bend) of the wing was measured (Ueno et al. 2014), and the data were provided in the workshop (2019).

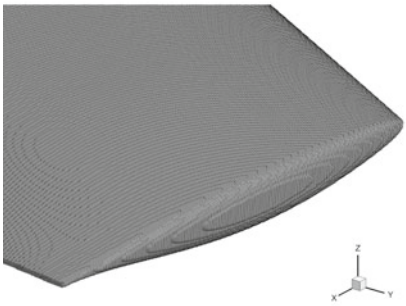
The grid is shown in Fig. 16.10. Here, a symmetric boundary condition is assigned on the $y = 0$ plane, and a half-span model is simulated. To reduce the computational cost, two different cell sizes are specified on the wall. The wing upper surface and the tail are covered by the finest level of the cell because the flow features in those regions are important in terms of accurate aerodynamic force simulation. The other parts (the fuselage and wing lower surface) are covered with the second next level of the cell to reduce the computational cost. The ratio of the IP distance to the cell size, r_{IP}, is set to 3. Coarse, medium, and fine grids are prepared to check the grid sensitivity. In addition, a "medium-b" grid is created by changing the number of cells in the second layer (refer to Fig. 16.1). Table 16.4 describes the specification of these grids. The lengths in the table are based on the actual scale of the NASA CRM ($c_{\text{ref}} = 275.8$ inch). The cell number slightly changes when the wing is deformed, and the numbers presented in the table are $\alpha = 2.94°$.

The UTCart computational cases are as follows. First, the grid sensitivity is examined at $\alpha = 2.94°$ on the coarse, medium, medium-b, and fine grids. Then, the flows at $\alpha = -1.79$, 0.62, 2.47, 2.94, 3.55, 4.65, and $5.72°$ are simulated on the medium grid. Furthermore, reference calculations are conducted by a flow solver FaSTAR, developed by JAXA (Hashimoto et al. 2012), on body-fitted grids. The computational grids are provided in the APC workshop (Third Aerodynamic Prediction Challenge (APC-III) 2019).

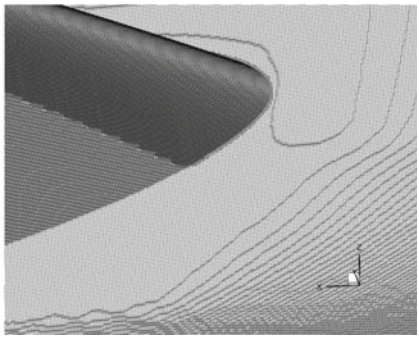
Figure 16.11 compares the surface pressure coefficient distributions of the two flow solvers. The qualitative features (e.g., the position of the shock on the wing upper surface) have good agreement with each other. Figure 16.12 presents the surface pressure coefficient distributions on the section of the wing. As illustrated in Fig. 16.13, the definition of the sections follows that of the APC workshop. These sections are identical to the positions of the pressure taps of the experiment. At the



(a) Overview of the computational grid (very coarse grid,only for visualization)



(b) Main wing tip

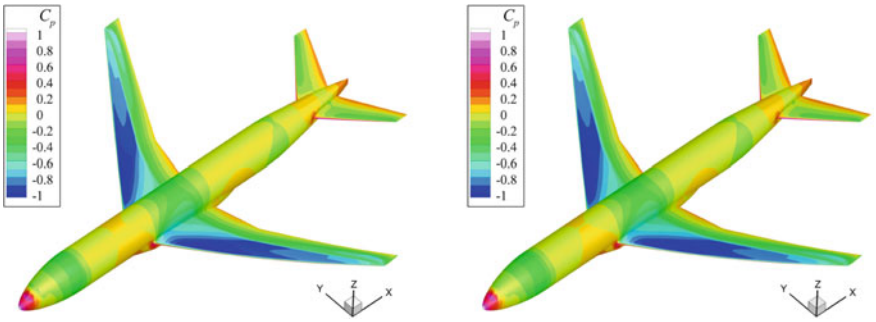


(c) Main wing root

Fig. 16.10 Computational grid for UTCart (medium grid, except for the overview)

Table 16.4 Settings of the computational grid around the NASA CRM for UTCart

	Coarse	Medium	Medium-b	Fine
Total cell number	31,055,490	61,988,288	54,335,363	117,882,932
Domain size (inch)	4.80×10^4	3.60×10^4	3.60×10^4	5.40×10^4
$\Delta x_{\min}$ (inch)	0.732	0.549	0.549	0.412
Number of cells in the first layer	3	3	3	3
Number of cells in the second layer	3	6	3	8
Number of cells in the rest of the layers	3	3	3	3
$\frac{c_{\text{ref}}}{\Delta x_{\min}}$	753	1004	1004	1339



(a) UTCart (medium grid) (b) FaSTAR (fine grid)

Fig. 16.11 Surface pressure coefficient calculated by UTCart, medium grid ($\alpha = 2.94°$)

inboard sections, the surface pressure coefficient in the UTCart result has good agreement with the FaSTAR result and the experimental data. The pressure distributions at the outboard sections are slightly different from the FaSTAR result. The UTCart grid size on the upper surface of the wing is uniform. Accordingly, the number of cells in the local chord is smaller than that of the outboard sections, indicating that the grid resolution relative to the local chord length is low in the outboard sections and is assumed to be one of the causes of the inaccuracy. Furthermore, the shock thickness of the UTCart result is thinner than that in the FaSTAR result at Section I. This indicates that the UTCart computational grid has a higher grid resolution in the chord-wise direction than the grid for FaSTAR.

Figure 16.14 presents the component-wise aerodynamic coefficients. The pressure drag computed by UTCart is overestimated, especially on the coarse grid. The pressure drag in the medium-b grid result is 3 drag counts (1 drag count is 10^{-4}) larger than the value of the medium grid result. This indicates that the pressure drag is dependent on the grid resolution in the region away from the wall, revealing that a proper grid refinement is required. Furthermore, the viscous drag is overestimated by 7 drag counts even on the fine grid. This difference is caused by the wing and the body. Simultaneously, the lift coefficient in the UTCart result is overestimated as compared to the FaSTAR result, whereas the pitching moment coefficient is underestimated. For these two coefficients, the trend of grid convergence is observed toward the FaSTAR result. The main cause of these discrepancies is the main wing. It may also be due to the grid resolutions that capture the curvature of the leading edge and the thickness of the trailing edges.

Figure 16.15 shows the computed and measured drag polar (drag coefficient vs. lift coefficient) of this aircraft configuration. The basic trend of each coefficient indicates fair agreement between the UTCart and FaSTAR results and between the UTCart results and the experimental data.

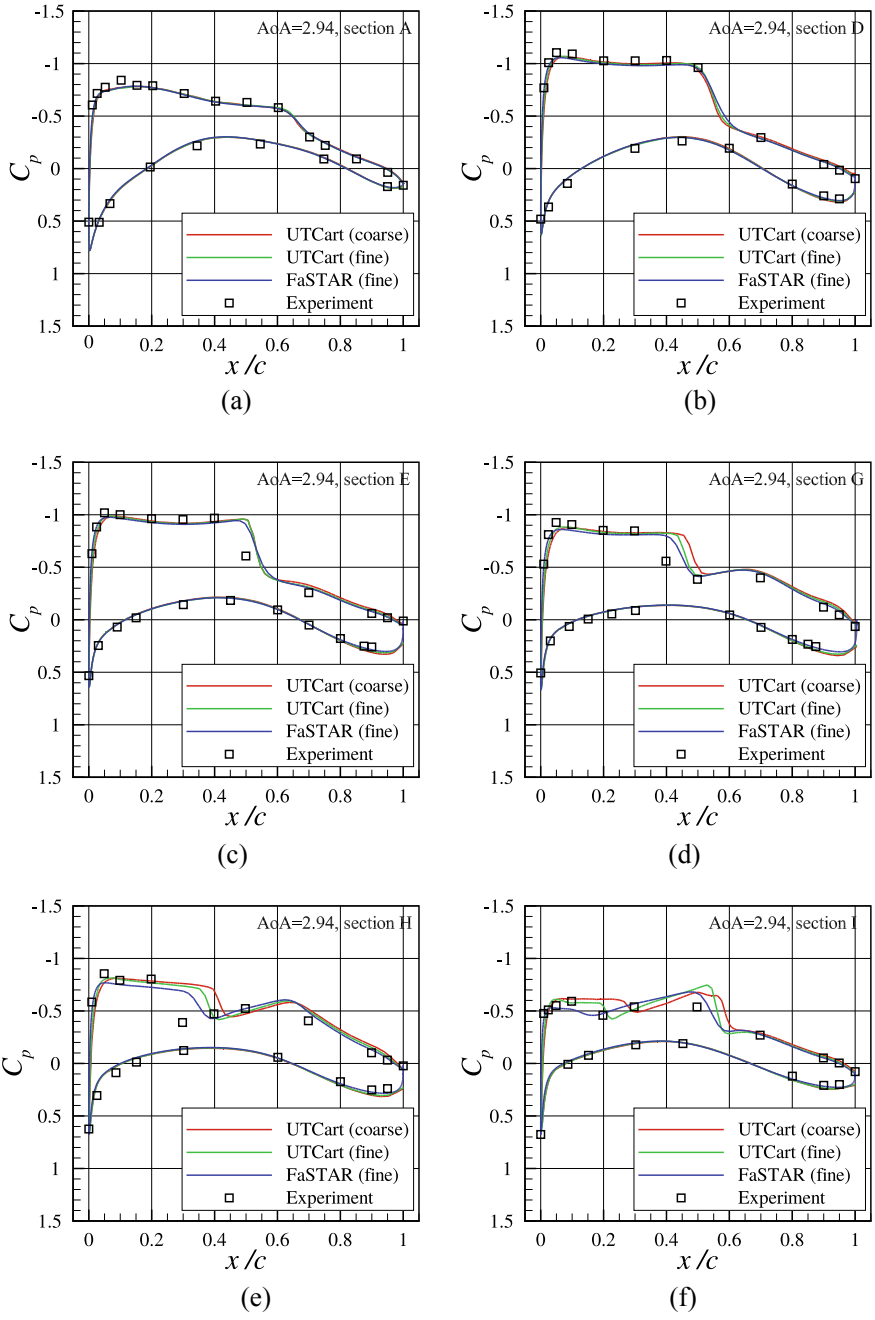


Fig. 16.12 Surface pressure coefficient on the wing sections ($\alpha = 2.94°$)

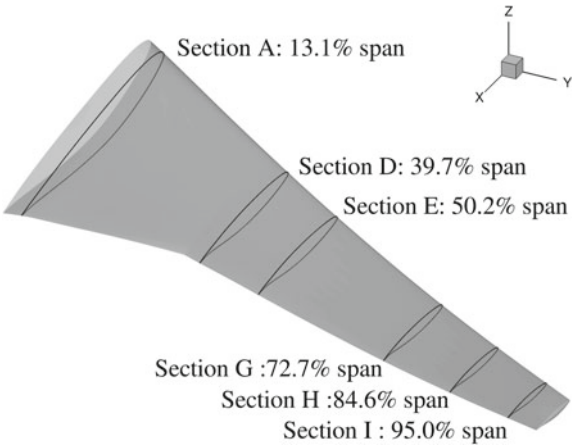
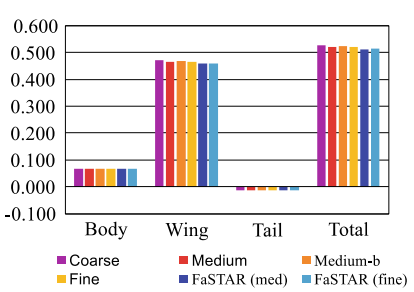
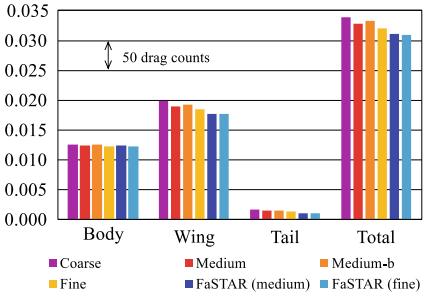


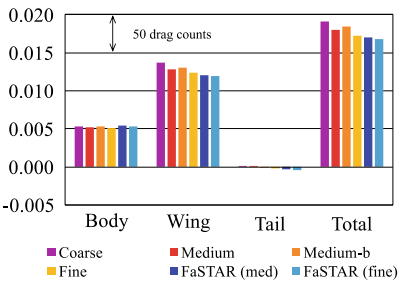
Fig. 16.13 Definition of the wing sections of the NASA CRM



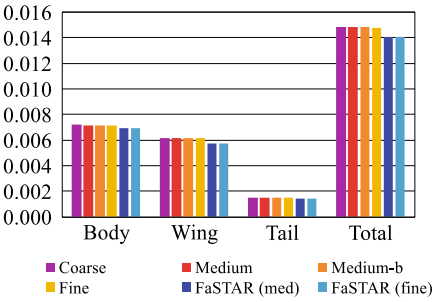
(a) Lift coefficient C_L



(b) Drag coefficient C_D



(c) Pressure drag coefficient C_{Dp}



(d) Viscous drag coefficient C_{Dv}

Fig. 16.14 Comparison of the aerodynamic coefficients of the NASA CRM ($\alpha = 2.94°$)

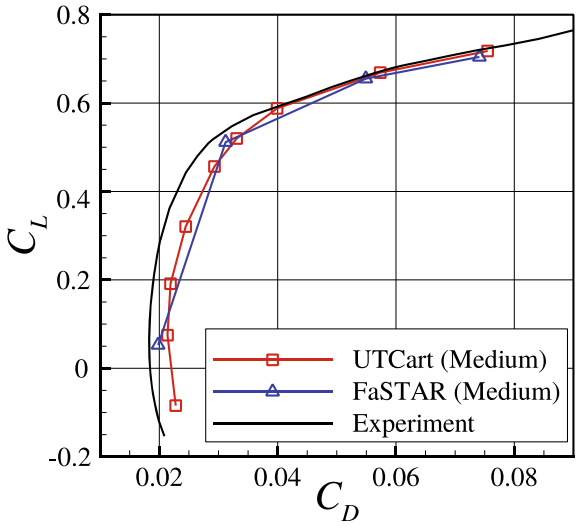


Fig. 16.15 Drag polar of NASA CRM

16.4 Summary

We explored the methodology for high Reynolds number flow simulations using hierarchical Cartesian grids in combination with IBM. To reduce the computational cost, the wall function, i.e., the model of the near-wall part of the turbulent boundary layer, was combined with IBM. The velocity of the wall model was modified to linear profile to avoid numerical problems. We also demonstrated that the modification of the eddy viscosity is essential to retain the balance of the shear stress near the wall. The temperature profile is also modified accordingly. The object surface was immersed in the Cartesian grid, and uncertainty was thus remarked in the evaluation of the aerodynamic force. We clarified the relation between the aerodynamic force and the numerical flux in the flow calculation. In the 2D bump problem, modified IBM, the new approach introduced in this study, achieved higher accuracy than that of the original IBM in predicting the skin friction and pressure coefficients. Consistent grid convergence toward the converged solution was observed. In the flow simulations in the NASA CRM configuration under the cruise condition, the flow patterns showed fair agreement with those of FaSTAR and experimental data. Also, the basic trend of aerodynamic coefficients was predicted correctly using the UTCart.

The proposed framework can be used to estimate the basic flow feature around a complex geometry within a short time. Although the accuracy of the conventional CFD simulation may be higher once a well-tailored body-fitted grid is prepared, the proposed framework can also predict the flow with a certain degree of accuracy. The grid generation is fully automatic; thus, the total workload for the flow simulations is reduced compared to that of the conventional simulation on body-fitted grids. In addition, shape optimization problems are conducted without a manual procedure in the sequence of calculations. Thus, the proposed framework will be beneficial as a tool for aerodynamic design.

Acknowledgements This work was supported by JSPS KAKENHI grant numbers 23760767 [Grant-in-Aid for Young Scientists (B)] and 15H05559 [Grant-in-Aid for Young Scientists (A)]. Additionally, the code development would not possible without the effort of many alimonies of Rinoie-Imamura Laboratory of the Department of Aeronautics and Astronautics, the University of Tokyo. We would like to thank their contributions.

References

Allmaras SR, Johnson FT, Spalart PR (2012) Modification and clarification for the implementation of the Spalart-Allmaras turbulence model. In: ICCFD7-1902

CFL3D Version 6 Home Page. https://cfl3d.larc.nasa.gov/. Retrieved on 29 May 2019

Capizzano F (2011) Turbulent wall model for immersed boundary methods. AIAA J 49(11):2367–2381. https://doi.org/10.2514/1.j050466

Chakravarthy SR, Osher S (1983) Numerical experiments with the Osher upwind scheme for the Euler equations. AIAA J 21(9):1241–1248

Choi H, Moin P (2012) Grid-point requirements for large eddy simulation: Chapman's estimates revisited. Phys Fluids 24(1):011702

de Tullio MD, de Palma P, Iaccarino G, Pascazio G, Napolitano M (2007) An immersed boundary method for compressible flows using local grid refinement. J Comput Phys 225(2):2098–2117

Drag Prediction Workshop. https://aiaa-dpw.larc.nasa.gov. Retrieved on 12 Oct 2017

Green J, Quest J (2011) A short history of the European transonic wind tunnel ETW. Progr Aerosp Sci 47(5):319–368

Hashimoto A, Murakami K, Aoyama T, Yamamoto K, Murayama M, Lahur PR (2014) Drag prediction on NASA common research model using automatic hexahedra grid-generation method. J Aircr 51(4):1172–1182

Hashimoto A, Murakami K, Aoyama T, Ishiko K, Hishida M, Sakashita M, Lahur P (2012) Toward the fastest unstructured CFD code 'FaSTAR'. In: Proceedings of the 50th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, AIAA Paper No. 2012-1075. https://doi.org/10.2514/6.2012-1075

Imamura T, Tamaki Y, Harada (2017) Parallelization of a compressible flow solver (UTCart) on cell-based refinement Cartesian grid with immersed boundary method. In: Proceedings of the 29th international conference on parallel computational fluid dynamics

Kusunose K, Crowder JP (2002) Extension of wake-survey analysis method to cover compressible flows. J Aircr 39(6):954–963. https://doi.org/10.2514/2.3048

Lee-Rausch EM, Hammond DP, Nielsen EJ, Pirzadeh S, Rumsey CL (2014) Application of the FUN3D solver to the 4th AIAA drag prediction workshop. J Aircr 51(4):1149–1160

METIS—Serial graph partitioning and fill-reducing matrix ordering. http://glaros.dtc.umn.edu/gkhome/metis/metis/overview. Retrieved on 21 May 2019

NASA, Turbulence Modeling Resource. http://turbmodels.larc.nasa.gov/. Retrieved on 21 May 2019

Nonomura T, Onishi J (2017) A comparative study on evaluation methods of fluid forces on Cartesian grids. Math Probl Eng. https://doi.org/10.1155/2017/8314615

Rivers MB, Dittberner A (2014) Experimental investigations of the NASA common research model. J Aircr 51(4):1183–1193. https://doi.org/10.2514/1.c032626

Sclafani AJ, Vassberg J, Rumsey C, DeHaan M, Pulliam TH (2010) Drag prediction for the NASA CRM wing/body/tail using CFL3D and OVERFLOW on an overset mesh. In: Proceedings of the 28th AIAA applied aerodynamics conference, 2010, AIAA Paper No. 2010-4219

Sclafani AJ, Vassberg JC, Winkler C, Dorgan AJ, Mani M, Olsen ME, Coder J (2013) DPW-5 analysis of the CRM in a wing-body configuration using structured and unstructured meshes. In: Proceedings of the 51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, AIAA Paper No. 2013-0048

Shima E, Kitamura K (2011) Parameter-free simple low-dissipation AUSM-family scheme for all speeds. AIAA J 49(8):1693–1709

Shima E, Kitamura K, Haga K (2013) Green-Gauss/weighted-least-squares hybrid gradient reconstruction for arbitrary polyhedra unstructured grids. AIAA J 51(11):2740–2747

Shima E (1997) A simple implicit scheme for structured/unstructured CFD. In: Proceedings of the 29th fluid dynamics symposium (in Japanese)

Spalart PR, Allmaras SR (1992) A one-equation turbulence model for aerodynamic flows. In: Proceedings of the 52nd aerospace science meeting, AIAA Paper No. 92-0439. https://doi.org/10.2514/6.1992-439

Takahashi Y, Imamura T (2014) High Reynolds number steady state flow simulation using immersed boundary method. In: Proceedings of the 52nd aerospace science meeting, AIAA Paper No. 2014-0228

Tamaki Y, Harada M, Imamura T (2017) Near-wall modification of Spalart-Allmaras turbulence model for immersed boundary method. AIAA J 55:3027–3039. https://doi.org/10.2514/1.J055824

Tamaki Y, Imamura T (2018) Turbulent flow simulations of the common research model using immersed boundary method. AIAA J 56(6):2271–2282. https://doi.org/10.2514/1.J056654

Tamaki Y (2018) Turbulent flow simulations around aircraft using hierarchical Cartesian grids and the immersed boundary method. Ph.D. Dissertation Thesis of the University of Tokyo

Tennekes H, Lumley JL (1972) A first course in turbulence. MIT press, Cambridge

Third Aerodynamic Prediction Challenge (APC-III). https://cfdws.chofu.jaxa.jp/apc/apc3/. Retrieved on 21 May 2019

Tinoco EN, Brodersen OP, Keye S, Lain KR, Feltrop E, Vassberg JC, Mani M, Rider B, Wahls RA, Morrison JH (2017) Summary of data from the sixth AIAA CFD drag prediction workshop: CRM cases 2 to 5. In: Proceedings of the 55th AIAA aerospace sciences meeting, AIAA Paper No. 2017-1208. https://doi.org/10.2514/6.2017-1208

Ueno M, Kozai M, Koga S (2014) Transonic wind tunnel test of the NASA CRM: Volume 1, Tech. rep., Japan Aerospace Exploration Agency, JAXA-RM-13-017E

Vassberg JC, Tinoco EN, Mani M, Rider B, Zickuhr T, Levy D, Brodersen O, Eisfeld B, Crippa S, Wahls R, Morrison JH, Mavriplis DJ, Murayama M (2014) Summary of the fourth AIAA computational fluid dynamics drag prediction workshop. J Aircr 51(4):1070–1089. https://doi.org/10.2514/1.c032418

van Dam CP (1999) Recent experience with different methods of drag prediction. Progr Aerosp Sci 35(8):751–798. https://doi.org/10.1016/s0376-0421(99)00009-3

Vassberg JC, DeHaan MA, Rivers SM, Wahls RA (2008) Development of a common research model for applied CFD validation studies. In: Proceedings of the 26th applied aerodynamics conference, AIAA Paper No. 2008-6919. https://doi.org/10.2514/6.2008-6919

Wahls R (2001) The national transonic facility: A research retrospective, Proceeding of the 39th Aerospace Meeting, 2001, AIAA Paper No. 2001-0754.

Wang G, Schöppe A, Heinrich R (2010) Comparison and evaluation of cell-centered and cell-vertex discretization in the unstructured TAU-code for turbulent viscous flows. In: ECCOMAS CFD 2010

White FM (2006) Viscous fluid flow, 3rd edn. McGraw-Hill, New York

Yamamoto K, Tanaka K, Murayama M (2012) Effect of a nonlinear constitutive relation for turbulence modeling on predicting flow separation at wing-body juncture of transonic commercial aircraft. In: Proceedings of the 20th AIAA applied aerodynamics conference, AIAA Paper No. 2012-2895. https://doi.org/10.2514/6.2012-2895

Yoon S, Jameson A (1988) Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations. AIAA J 26(9):1025–1026