

Immersed boundary method for high-order flux reconstruction based on volume penalization

Jiaqing Kou^{a,b,*}, Saumitra Joshi^{a,b}, Aurelio Hurtado-de-Mendoza^{a,b}, Kunal Puri^a, Charles Hirsch^a,
Esteban Ferrer^{b,c}

^aNUMECA International S.A., Chaussee de la Hulpe 187, Brussels, B-1170, Belgium

^bETSIAE-UPM-School of Aeronautics, Universidad Politécnica de Madrid, Plaza Cardenal Cisneros 3, E-28040 Madrid, Spain

^cCenter for Computational Simulation, Universidad Politécnica de Madrid, Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain

Abstract

In the last decade, there has been a lot of interest in developing high-order methods as viable option for unsteady scale-resolving-simulations which are increasingly important in the industrial design process. High-order methods offer the advantage of low numerical dissipation, high efficiency on modern architectures and quasi mesh-independence. Despite significant advances in high-order solution methods, the general CFD workflow (geometry, CAD preparation, meshing, solution, post-processing) has largely remained unchanged, with mesh generation being a significant bottleneck and often determining the overall quality of the solution. In this work, we aim to combine the numerical advantages of the high-order Flux-Reconstruction (FR) method and the simplicity of the mesh generation (or lack thereof) of the Immersed Boundary Method (IBM) for steady and unsteady problems over moving geometries. The volume-penalization (penalty-IBM) method is selected for its ease of implementation and robustness. Detailed discussions about numerical implementation, including the boundary representation, mask function, data reconstruction, and selection of the penalization parameter are given. Advantages of combining volume penalization in the high-order framework are shown by various numerical test cases. The approach is firstly demonstrated for the linear advection-diffusion equation by investigating the numerical convergence for the coupled FR-IBM approach. Thereafter, the accuracy of the approach is demonstrated for canonical (static) test cases in 2D and 3D when compared to a standard body-fitted unstructured simulation. Finally, the efficiency of the method to handle moving geometries is demonstrated for the flow around an airfoil with pitching and plunging motions.

Keywords: volume penalization, flux reconstruction, immersed boundary method, high-order method, moving boundary

Contents

2	1 Introduction	2
	2 The Governing Equations	4

*Corresponding author.

Email address: jiaqing.kou@numeca.be, jiaqing.kou@alumnos.upm.es (Jiaqing Kou)

4	3 The Flux Reconstruction Method	5
	4 Immersed Boundary Method	8
6	4.1 The volume Penalization method	8
	4.2 Boundary representation and mask function	11
8	4.3 Treatment for moving boundaries	12
	4.4 Surface data reconstruction	13
10	4.5 Overview of the algorithm	15
	4.6 Discussions on error estimate and selection of penalization parameter	16
12	5 Test Cases	17
	5.1 One-dimensional advection-diffusion equation	18
14	5.2 Flow past a cylinder	20
	5.3 Flow past a NACA0012 airfoil	23
16	5.4 Flow past a sphere	25
	5.5 Flow past a pitching and plunging airfoil	26
18	6 Conclusions	29
	Appendix A Analytical mask function for different geometries	29
20	Appendix B Smoothing the mask function	30
	Appendix C Comparison of data reconstruction methods	31

22 1. Introduction

Despite significant advantages, the general CFD workflow used in the industrial design process has largely remained unchanged. The typical workflow consists of geometry definition, CAD preparation, meshing, numerical solution, post-processing and subsequent design optimization. Of these, meshing is often the most time consuming and can have a significant impact on the overall quality of solution. The bottleneck associated with mesh generation could be eased with the development of mesh independent schemes or alternatively, by developing methods designed for simple Cartesian grids via the Immersed Boundary Method (IBM). The desire to achieve the former has been a motivating factor for the development of high-order schemes on unstructured grids over the last decade. High-order methods are known to be more efficient for a given level of accuracy, highly scalable on modern architectures and display a level of quasi mesh independence for industrially relevant problems [1, 2]. Examples of high-order methods include Discontinuous Galerkin (DG) [3], Flux Reconstruction (FR) [4] and Spectral Difference (SD) [5] [6].

By contrast, the development of high-order methods on Cartesian grids for complex moving geometries using IBM or related approaches has been relatively unexplored. One of the earliest proponents of exploring this idea was Adrian Lew and his coworkers [7, 8] where the advantages and optimal order of convergence of using the DG method over standard finite-differences for a 2D Poisson problem was reported. The method was subsequently applied to problems in elasticity [9]. The solution of the Poisson problem with IBM and DG was also considered in [10, 11] and in [12] for the Hybridized Discontinuous Galerkin (HDG) method.

The HDG method on irregular domains was also developed for the Navier-Stokes equations in [13]. Fidkowski and Darmofal [14] were the first to report the use of cut-cell method to solve steady compressible flows over two-dimensional geometries based on the DG and Finite Element Method (FEM). The high-order cut-cell approach was also studied in [15, 16, 17] and more recently in [18] where stable discretizations on degenerate meshes are presented for high-order finite-difference methods. While the cut-cell approaches are undoubtedly superior on static grids, the extension to moving grids is far from straightforward.

The challenge to use high-order DG type methods for industrially relevant problems lies in the efficient handling of complex geometries which on Cartesian grids can be combined with efficient Adaptive Mesh Refinement (AMR) and highly optimized for GPU architectures [19, 20, 21]. Here, the IBM is a natural choice and in this work we present an approach that combines the flexibility and ease of implementation of the high-order FR method [4, 22, 23] on Cartesian grids with the IBM for moving geometries. The approach utilizes the compact stencil of the high-order operators and offers the possibility of locally varying the polynomial order for a more accurate representation of the boundary conditions than traditional finite difference or finite volume methods. The flexibility of handling complex moving geometries stems from the use of the IBM approach. Indeed, since its introduction [24], the IBM approach has been shown to be versatile and applicable to a variety of problems ranging from flow over complex geometries [25], multiphase flows [26] to Fluid Structure Interaction (FSI) [27, 28]. In an IBM based approach the governing equations (compressible or incompressible flow) are solved on a simple background Cartesian grid. The methods can be differentiated by the way in which the influence of the immersed boundary on the fluid is taken into account. This can be either through a *cut-cell* approach [29, 30, 31], ghost-fluid method [32, 33], direct forcing [34, 35, 36] or by the introduction of source terms [37] to take into account the presence of the geometry. Alternatively, IBM approaches can be classified as *sharp interface* or *diffuse interface* methods. It is worth noting that there are additional methods that are specific to a certain class of problems. These are the immersed interface method [38] and family of embedded boundary method [39] for viscous flow and FSI. The interested reader is referred to [40, 41, 42, 43] and the references cited therein for comprehensive reviews of the IBM method and their applications. Recently, a comprehensive convergence analysis with regard to the spatial and temporal resolution is presented by Zhou and Balachandar [44]. A systematic study on convergence of IBM and an overset grid method is performed by Vreman [45].

In recent years, the volume penalization method [37, 46] has attracted a lot of attention due to its robustness, simplicity and proofs of convergence [37, 42]. It follows a basic physical intuition that the solid wall can be modelled as a porous medium with vanishing diffusivity [47]. A characteristic or mask function χ is introduced that is 1 in the solid domain and 0 elsewhere. A source or penalty function is introduced and is active in the solid domain. The source term is designed to impose the desired boundary condition. When compared with the other IBM approaches, the reconstruction procedure and the distribution of the source term are not needed, thus largely reducing the computational cost. The extension to moving boundary problems is also straightforward. The penalization method can be traced back to the works of Courant [48] who introduced such an approach to transform constrained optimization problems into problems free of constraints. The volume penalization method for the Navier-Stokes was first proposed by Arquis and Caltagirone [49] to simulate the natural convection flow inside a fluid-porous cavity where a Brinkman type penalization was introduced to the momentum equation. Rigorous proofs of the convergence is given by Angot et al. [37] and Carbou and Fabrie [50] where it was proven that, as the penalization parameter η approaches 0, the solution of the penalized Navier-Stokes equations will converge to the solution of the Navier-Stokes equations with

no-slip boundary conditions. The extension of volume penalization method to general Robin type boundary conditions method was investigated by Ramière et al. [51] and Kadoch et al. [47] and Sakurai et al. [52] for the finite-volume method a pseudo-spectral method respectively. The volume penalization method has also been applied to compressible flows by Liu and Vasilyev [53], Brown-Dymkoski et al. [54] and Abgrall et al. [46]. The method has been successfully used for complex problems such as flapping wings [55], two-phase flow [56], FSI [57], thermal flows [58], and turbulent rotating flows [59]. The comparison between direct forcing method and the penalization method is studied in by Piquet et al. [60], where it was found that the volume-penalization method is a suitable and a possibly competitive IBM method for viscous flows in terms of predictive performance, accuracy and computational cost. A review of volume penalization method for numerical simulation of complex flows is given by Schneider [42].

Despite the amount of publication devoted to volume penalization, this technique has not been studied with the high-order methods. It does not need to treat complicated cell cuts, and is easy to be extended to moving boundaries. Therefore, it is worth investigating the performance of high-order methods with volume penalization method, which is the aim and novelty of the present work. The high-order method adopted in the current study is based on the FR approach [4, 22, 23]. It provides a differential framework for discontinuous finite element schemes, which is a unifying framework for high-order methods and can recover existing high-order schemes. Volume penalization is used to impose the no-slip boundary condition within the solid body. The present approach allows locally refining the solution near the wall to improve both the accuracy and the smoothness of the solution, using local p-refinement of the FR scheme. Increasing the polynomial order also leads to improved solution for moving boundary simulation. To the authors' knowledge, these advantages have not been reported in previous works about IBM.

This paper is organized as follows. Section 2 gives an introduction of the Navier-Stokes equations. Section 3 presents the high-order FR method for general conservation law. Section 4 details the penalty IBM method used in the present study, along with the method for surface data reconstruction and the handling of moving boundary. The proposed method is tested in Section 5, where cases with increasing complexity are shown. Finally, conclusions are drawn in Section 6.

2. The Governing Equations

The governing equations for a compressible viscous fluid are written as

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_x}{\partial x} + \frac{\partial \mathbf{F}_y}{\partial y} + \frac{\partial \mathbf{F}_z}{\partial z} = 0, \quad (1)$$

where \mathbf{U} denotes the vector of conserved variables $\mathbf{U} = (\rho, \rho u, \rho v, \rho w, E)^T$. ρ is the density, u , v and w are the velocity components and E is the total energy. The equations are closed by the ideal gas equation-of-state:

$$E = \frac{P}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2 + w^2), \quad (2)$$

where P is the static pressure and γ is the ratio of specific heats. The flux vectors \mathbf{F}_x , \mathbf{F}_y , \mathbf{F}_z contain the inviscid and viscous fluxes and are written as

$$\mathbf{F}_x = \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ u(E + P) \end{pmatrix} - \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + q_x \end{pmatrix} = \mathbf{F}_{x,inv} + \mathbf{F}_{x,usc} \quad (3)$$

$$\mathbf{F}_y = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + P \\ \rho vw \\ v(E + P) \end{pmatrix} - \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + q_y \end{pmatrix} = \mathbf{F}_{y,inv} + \mathbf{F}_{y,usc} \quad (4)$$

$$\mathbf{F}_z = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + P \\ w(E + P) \end{pmatrix} - \begin{pmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + q_z \end{pmatrix} = \mathbf{F}_{z,inv} + \mathbf{F}_{z,usc}. \quad (5)$$

In these equations, $\tau_{ij} = \mu(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial v_k}{\partial x_k})$ is the viscous stress tensor with μ denoting the dynamic viscosity. The heat flux vector $\nabla \mathbf{q}$ is given by

$$\frac{\partial \mathbf{q}}{\partial x_i} = \lambda \frac{\partial T}{\partial x_i}, \quad (6)$$

where λ is the thermal conductivity and T is the static temperature. The equations are solved in non-dimensional form with the introduction of the Prandtl number $\text{Pr} = \mu \frac{C_p}{\lambda}$, the Reynolds number $Re = \rho_{\text{ref}} V_{\text{ref}} L_{\text{ref}} / \mu_{\text{ref}}$ and the Mach number $M = V_{\text{ref}} / \sqrt{\gamma R_{\text{gas}} T_{\text{ref}}}$, with C_p being the specific heat capacity at constant pressure and R_{gas} being the gas constant. Finally, V_{ref} , L_{ref} , T_{ref} are reference velocity, length and temperature, respectively. The discretization of these equations with the Flux-Reconstruction method is described next.

3. The Flux Reconstruction Method

Flux reconstruction is a high-order framework which unifies a number of other high-order methods like the SD method and the nodal DG method. FR was first introduced by Hyunh for advection [4] and diffusion [22] equations. This method is detailed below. Consider the following hyperbolic system of conservation law:

$$\begin{aligned} \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{F}_{\text{inv}} + \mathbf{F}_{\text{usc}}) &= \mathbf{S} \\ \mathbf{Q} - \nabla \mathbf{U} &= 0 \end{aligned} \quad (7)$$

where \mathbf{F}_{inv} , \mathbf{F}_{usc} , \mathbf{S} refer to the inviscid flux, viscous flux and the source term. These vectors are functions of solution \mathbf{U} and its gradient $\mathbf{Q} = \nabla \mathbf{U}$. The space dimension is defined as \mathcal{D} . After space discretization,

the computational domain Ω is divided into N_c distinct cells. In each cell, the discrete solution \mathbf{U}_i^δ is locally approximated by a polynomial of degree P , defined at N_p solution points. In addition, the flux at each interface of an element is approximate by a polynomial of degree $P + 1$, defined at N_f flux points on the element interface. An isoparametric spatial mapping $\mathcal{M} : \mathbf{x} \rightarrow \boldsymbol{\xi}$ is defined to transform physical coordinates of the solution and flux points to reference coordinates [61, 62]. In the present study, we utilize a standard tensor-product formulation with a Legendre polynomial basis to define the polynomial interpolation. The solution and flux points are located at the Gaussian quadrature points. The standard flux reconstruction process for the general conservation law can include seven stages [63] as follows:

1. Getting the interpolated solution at flux points. The interpolated solution at the flux point $\boldsymbol{\xi}^{\delta F}$ is given by the following polynomial interpolation:

$$\mathbf{U}^{\delta D}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathbf{U}_i^{\delta D} \mathcal{I}_i^P(\boldsymbol{\xi}) \quad (8)$$

where \mathcal{I}_i^P refers to the nodal basis function defined at each solution point with polynomials of degree P , and $\mathbf{U}_i^{\delta D}$ is the solution at the i th solution point.

2. Obtaining common solution at the flux point, computed from the left and right interpolated solutions. The Local Discontinuous Galerkin (LDG) is chosen for the common solution:

$$\mathbf{U}_{f,j}^{\delta I} = \{\!\!\{ \mathbf{U}_{f,j}^{\delta F} \}\!\!\} - \beta \cdot \llbracket \mathbf{U}_{f,j}^{\delta F} \rrbracket \quad (9)$$

where β refers to an upwinding parameter, δF and δI refers to the interpolated flux at flux points and the common flux, respectively. $\{\!\!\{ \cdot \}\!\!\}$ and $\llbracket \cdot \rrbracket$ compute the mean and jump values of the interpolated solution. The correction solution is subsequently given as:

$$\mathbf{U}_{f,j}^{\delta C} = \mathbf{U}_{f,j}^{\delta I} - \mathbf{U}_{f,j}^{\delta F}. \quad (10)$$

3. Computing the gradient of solution \mathbf{Q} from the correction solution $\mathbf{U}^{\delta C}$ and the discrete solution $\mathbf{U}^{\delta D}$:

$$\tilde{\mathbf{Q}}^{\delta D} = \tilde{\nabla} \mathbf{U} = \tilde{\nabla} \mathbf{U}^{\delta D} + \tilde{\nabla} \mathbf{U}^{\delta C} \quad (11)$$

where the discrete gradient $\tilde{\nabla} \mathbf{U}^{\delta D}$ in the reference space is computed from the gradient of the discrete solution $\mathbf{U}^{\delta D}$:

$$\tilde{\nabla} \mathbf{U}^{\delta D}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathbf{U}_i^{\delta D} \tilde{\nabla} \mathcal{I}_i^P(\boldsymbol{\xi}) \quad (12)$$

The corrected gradient $\tilde{\nabla} \mathbf{U}^{\delta C}$ is computed by transforming the correction solution from the flux points to the solution points. This is achieved by the correction function:

$$\tilde{\nabla} \mathbf{U}^{\delta C}(\boldsymbol{\xi}) = \sum_{f=1}^{N_{face}} \sum_{j=1}^{N_f} \tilde{\nabla} \mathcal{C}_{f,j}^{P+1}(\boldsymbol{\xi}) \cdot \mathbf{U}_{f,j}^{\delta C} \quad (13)$$

where the function $\mathcal{C}_{f,j}^{P+1}$ is the correction function, which is of polynomial order $P + 1$. Then gradient is transformed to physical space.

4. Computing the discrete flux at each solution point from the solution and gradient values. The inviscid and viscous flux at each solution point is defined by the same polynomial interpolation with degree P :

$$\mathbf{F}_{\text{ivc}}^{\delta\text{D}}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathbf{F}_{\text{ivc},i}^{\delta\text{D}} \mathcal{I}_i^P(\boldsymbol{\xi}), \quad \mathbf{F}_{\text{vsc}}^{\delta\text{D}}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathbf{F}_{\text{vsc},i}^{\delta\text{D}} \mathcal{I}_i^P(\boldsymbol{\xi}). \quad (14)$$

The interpolated fluxes $\mathbf{F}_{\text{ivc}}^{\delta\text{F}}$ and $\mathbf{F}_{\text{vsc}}^{\delta\text{F}}$ at flux points are obtained from the above formulation.

5. Obtaining the interaction flux $\mathbf{F}^{\delta\text{I}}$ at the flux point. This flux is approximated by a Riemann solver $\mathcal{R}(\mathbf{U}_{f,j,-}^{\delta\text{F}}, \mathbf{Q}_{f,j,-}^{\delta\text{F}}, \mathbf{U}_{f,j,+}^{\delta\text{F}}, \mathbf{Q}_{f,j,+}^{\delta\text{F}})$. For the inviscid flux, the Rusanov flux is used. The viscous flux is obtained by the LDG approach, where the interaction flux is computed as:

$$\mathbf{F}_{\text{vsc},f,j}^{\delta\text{I}} = \mathbf{F}_{\text{vsc},f,j}^{\delta\text{F}} + \tau \cdot \llbracket \mathbf{U}_{f,j}^{\delta\text{F}} \rrbracket + \beta \cdot \llbracket \mathbf{F}_{\text{vsc},f,j}^{\delta\text{F}} \rrbracket \quad (15)$$

where the τ is the parameter controlling the jump of the solution, and β is the upwinding parameter defined previously. For LDG approach, a combination of $\beta = 0.5$ and $\tau = 0.1$ is used here, in order to promotes compactness of the FR scheme in multiple dimensions [63].

6. Computing the flux correction term. This term is constructed from the interaction flux, the interpolated flux and the correction function. The correction flux is the difference between the interaction and the interpolated fluxes:

$$\mathbf{F}_{f,j}^{\delta\text{C}} = \mathbf{F}_{f,j}^{\delta\text{I}} - \mathbf{F}_{f,j}^{\delta\text{F}} \quad (16)$$

The corrected divergence of flux correction term is transformed from the flux difference at the boundary to the solution points through the correction function:

$$\tilde{\nabla} \cdot \mathbf{F}^{\delta\text{C}}(\boldsymbol{\xi}) = \sum_{k=1}^{\mathcal{D}} \sum_{f=1}^{N_{\text{face}}} \sum_{j=1}^{N_f} \tilde{\nabla}_k \mathcal{C}_{f,j}^{P+1}(\boldsymbol{\xi}) \cdot \mathbf{F}_{f,j,k}^{\delta\text{C}} \quad (17)$$

where k refers to the spatial direction index.

7. Calculating divergence of the continuous flux from the local discrete flux divergence and the corrected divergence:

$$\tilde{\mathbf{U}}_t^{\delta\text{D}} = -\tilde{\nabla} \cdot \mathbf{F} = -\tilde{\nabla} \cdot \mathbf{F}^{\delta\text{D}} - \tilde{\nabla} \cdot \mathbf{F}^{\delta\text{C}} = -\sum_{k=1}^{\mathcal{D}} \sum_{i=1}^{N_p} \tilde{\nabla}_k \mathcal{I}_i^P(\boldsymbol{\xi}) \cdot \mathbf{F}_{i,k}^{\delta\text{D}} - \sum_{k=1}^{\mathcal{D}} \sum_{f=1}^{N_{\text{face}}} \sum_{j=1}^{N_f} \tilde{\nabla}_k \mathcal{C}_{f,j}^{P+1}(\boldsymbol{\xi}) \cdot \mathbf{F}_{f,j,k}^{\delta\text{C}} \quad (18)$$

Finally, the flux divergence needs to be transformed into the physical space. Once the divergence of continuous flux is obtained, the equation can be advanced in time by any explicit or implicit time-marching method. The governing equation can be discretized as

$$\frac{d\mathbf{U}}{dt} = \mathbf{R}(\mathbf{U}) \quad (19)$$

where $\mathbf{R}(\mathbf{U})$ refers to the residual of the equation, which is a function of \mathbf{U} . For the time marching method,

we use the classic TVD Runge-Kutta method. Equations of time integration are

$$\mathbf{U}^* = \mathbf{U}^n + \Delta t \mathbf{R}(\mathbf{U}^n) \quad (20)$$

$$\mathbf{U}^{**} = \frac{1}{4}[3\mathbf{U}^n + \mathbf{U}^* + \Delta t \mathbf{R}(\mathbf{U}^*)] \quad (21)$$

$$\mathbf{U}^{n+1} = \frac{1}{3}[\mathbf{U}^n + 2\mathbf{U}^{**} + 2\Delta t \mathbf{R}(\mathbf{U}^{**})] \quad (22)$$

where Δt denotes the time step, and n is the present time index. Note that when IBM source term is present, a splitting approach can be used to handle the stiffness of the source term. It should also be pointed out that the performance of flux reconstruction depends on six factors [63], including the location of solution and flux point, the Riemann solvers used for computing the common solution values and the interaction fluxes, and the form of the correction functions for solution and flux values. The boundary conditions, like the far-field characteristic boundary condition, are imposed in a weak-Riemann formulation following Mengaldo et al. [64]. The ghost state from the boundary side of the face is given from the boundary condition, while the flux is calculated by a Riemann solver.

4. Immersed Boundary Method

The basic idea of IBM is to impose boundary conditions to the non body-fitted mesh with proper numerical treatment. Volume penalization is a particular method, which imposes boundary condition through penalizing the velocity of solution points in the solid body. This method is easy to understand with rigorous theoretical foundation, therefore it is selected for the present study.

4.1. The volume Penalization method

The volume penalization method imposes boundary conditions by introducing penalization source terms to the governing equations. In this approach, a mask function which distinguishes between the fluid region Ω_f and solid region Ω_s is firstly defined:

$$\chi(\mathbf{x}, t) = \begin{cases} 1, & \text{if } \mathbf{x} \in \Omega_s \\ 0, & \text{otherwise} \end{cases} \quad (23)$$

This mask function is used to determine whether the IBM force should be imposed to the current solution point [46] [54]. For moving boundaries, $\chi(\mathbf{x}, t)$ is time-dependent. It should be noted that this usual definition of mask function will lead to a sharp jump of source term between the solid and fluid points near the boundary, which may lead to spurious oscillations (or Gibbs phenomena) of the hydrodynamic forces on moving obstacles [55]. In addition, for static obstacle, this sharp mask function may also lead to oscillations of flow variables near the wall. These oscillations can be reduced by smoothing the mask function, which will smooth the transition between the solid and fluid points, thus allowing smaller penalty force for solution points near the wall. This strategy is also tested and will be discussed in Appendix B. The volume penalization method for a high-order, Cartesian mesh is illustrated in Fig. 1. All solution points, covered by the solid region, need to be penalized to impose the boundary condition.

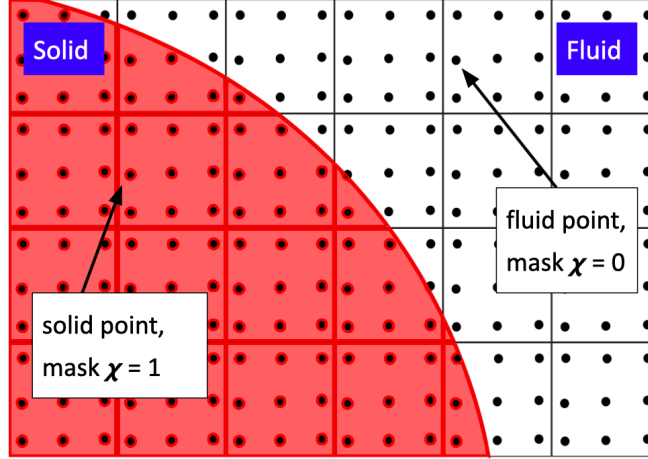


Figure 1: Schematic illustration of volume penalization for high-order method. The computational domain is discretized by the Cartesian grid. Solution points defining the high-order polynomial are represented by black circles. The polynomial order for the Cartesian grid is $P = 2$. The solid body Ω_s is highlighted in the red region.

The Navier-Stokes equations with IBM is written as:

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{RHS} + \chi \mathbf{S}(\mathbf{U}) \quad (24)$$

where \mathbf{S} refers to the IBM forcing term. \mathbf{RHS} refers to the right hand side term of the Navier-Stokes equation

$$\mathbf{RHS} = -\left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z}\right). \quad (25)$$

For the Dirichlet boundary condition for velocity $\mathbf{u}_s = (u_s, v_s, w_s)^T$ of the solid body, the source term is considered as:

$$\mathbf{S}(\mathbf{U}) = \frac{1}{\eta} \times \begin{pmatrix} 0 \\ \rho u_s - \rho u \\ \rho v_s - \rho v \\ \rho w_s - \rho w \\ \frac{\rho}{2}(u_s^2 + v_s^2 + w_s^2) - \frac{\rho}{2}(u^2 + v^2 + w^2) \end{pmatrix} \quad (26)$$

where η denotes the penalization parameter for IBM. The penalization terms proposed were used in [46] for compressible Navier-Stokes equations. Generally, the penalization parameter η should be sufficiently small to ensure accuracy. The stiffness of the equations depends on η , where small η value leads to very stiff source terms. In practice, the explicit time step Δt is suggested to be the penalization parameter [57]. Discussion on this argument is given in Sec. 4.6. The above equation can be used for moving bodies, where the solid velocity is updated based on the equations of motion. When no-slip boundary condition is considered, the condition $\mathbf{u}_s = (0, 0, 0)^T$ will be imposed.

The governing equation Eq.24 is marched in time by efficient time integration methods. Due to the stiffness of the source term, we use the second-order Strang splitting [65] approach to add the source term. As discussed by Piquet et al. [60], with Strang splitting, penalization terms are computed exactly for the momentum and energy equations. At time step n , the following sequence of operations is performed:

$$\text{step 1 : } \frac{\mathbf{U}_1 - \mathbf{U}^n}{\Delta t_1} = \mathbf{S}(\mathbf{U}_1), \Delta t_1 = \Delta t/2, \mathbf{U}_{1,0} = \mathbf{U}^n \quad (27)$$

$$\text{step 2 : } \frac{\mathbf{U}_2 - \mathbf{U}_1}{\Delta t_2} = \mathbf{RHS}(\mathbf{U}_2), \Delta t_2 = \Delta t, \mathbf{U}_{2,0} = \mathbf{U}_1 \quad (28)$$

$$\text{step 3 : } \frac{\mathbf{U}^{n+1} - \mathbf{U}_2}{\Delta t_3} = \mathbf{S}(\mathbf{U}^{n+1}), \Delta t_3 = \Delta t/2, \mathbf{U}_0^{n+1} = \mathbf{U}_2 \quad (29)$$

202 Currently, in step 2, we use the third-order TVD Runge-Kutta method to perform explicit time marching.
 In step 1 and step 3, when adding the source term, both implicit or explicit forcing methods can be considered.
 204 The implicit forcing method leads to better numerical stability, which is especially beneficial for penalty
 method due to a very stiff source term. The approaches are as follows:

Explicit forcing : The explicit formulation is simply given as:

$$\frac{\mathbf{U}_1 - \mathbf{U}^n}{\Delta t_1} = \mathbf{S}(\mathbf{U}^n), \quad (30)$$

$$\mathbf{U}_1 = \mathbf{U}^n + \Delta t_1 \cdot \mathbf{S}(\mathbf{U}^n). \quad (31)$$

Implicit forcing : For implicit implementation of the penalty method, the backward Euler method with first-order Taylor expansion leads to the following formulation

$$\frac{\mathbf{U}_1 - \mathbf{U}^n}{\Delta t_1} = \mathbf{S}(\mathbf{U}^n) + \frac{\partial \mathbf{S}(\mathbf{U}^n)}{\partial \mathbf{U}} (\mathbf{U}_1 - \mathbf{U}^n) \quad (32)$$

After some manipulation, the following equation is obtained:

$$\left(\mathbf{I} - \Delta t_1 \frac{\partial \mathbf{S}}{\partial \mathbf{U}}(\mathbf{U}^n) \right) \mathbf{U}_1 = \mathbf{U}^n + \Delta t_1 \left(\mathbf{S}(\mathbf{U}^n) - \frac{\partial \mathbf{S}}{\partial \mathbf{U}}(\mathbf{U}^n) \mathbf{U}^n \right) \quad (33)$$

where \mathbf{I} is the identity matrix. From Eq. 26, the Jacobian matrix of the IBM force term, is then derived as

$$\frac{\partial \mathbf{S}}{\partial \mathbf{U}} = -\frac{1}{\eta} \times \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{2}(u^2 + v^2 + w^2) & u & v & w & 0 \end{pmatrix} \quad (34)$$

Therefore, the inversion of matrix $\mathbf{I} - \Delta t_1 \frac{\partial \mathbf{S}}{\partial \mathbf{U}}(\mathbf{U}^n)$ can be derived analytically

$$\left(\mathbf{I} - \Delta t_1 \frac{\partial \mathbf{S}}{\partial \mathbf{U}}(\mathbf{U}^n) \right)^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{\eta}{\Delta t_1 + \eta} & 0 & 0 & 0 \\ 0 & 0 & \frac{\eta}{\Delta t_1 + \eta} & 0 & 0 \\ 0 & 0 & 0 & \frac{\eta}{\Delta t_1 + \eta} & 0 \\ \frac{\Delta t_1}{2\eta}(u^2 + v^2 + w^2) & -\frac{\Delta t_1 u}{\Delta t_1 + \eta} & -\frac{\Delta t_1 v}{\Delta t_1 + \eta} & -\frac{\Delta t_1 w}{\Delta t_1 + \eta} & 1 \end{pmatrix} \quad (35)$$

By Substituting Eq.34 and Eq.35 into Eq.33, the first and third step of time integration can be efficiently solved.

4.2. Boundary representation and mask function

Representation of solid boundaries is a crucial aspect of IBM approaches. It serves for two main purposes: 1) the definition of mask function. For general geometries where a simple shape function cannot be found, the mask function χ should be determined by effective methods to identify whether the present solution point is inside or outside the solid body. 2) the computation of aerodynamic coefficients. This depends on how the surface of obstacle is discretized, since the flow quantities on the surface need to be interpolated from data of its surrounding solution points, and we also need to get the surface normal and area for force computation.

The discretization of immersed boundary should be flexible enough to handle complex geometries. In the present study, we choose to use a set of Lagrangian marker points to represent the solid boundary, defined as immersed boundary (IB) points. The marker points are connected by linear elements, i.e., line segments in two dimensions and triangular elements in three dimensions. Calculations of the geometrical quantities, including the surface normal, the interpolation stencil for data reconstruction, and the surface distance, can be performed efficiently with this representation [66, 36]. The development of algorithm to compute the mask function also depends on such discretization.

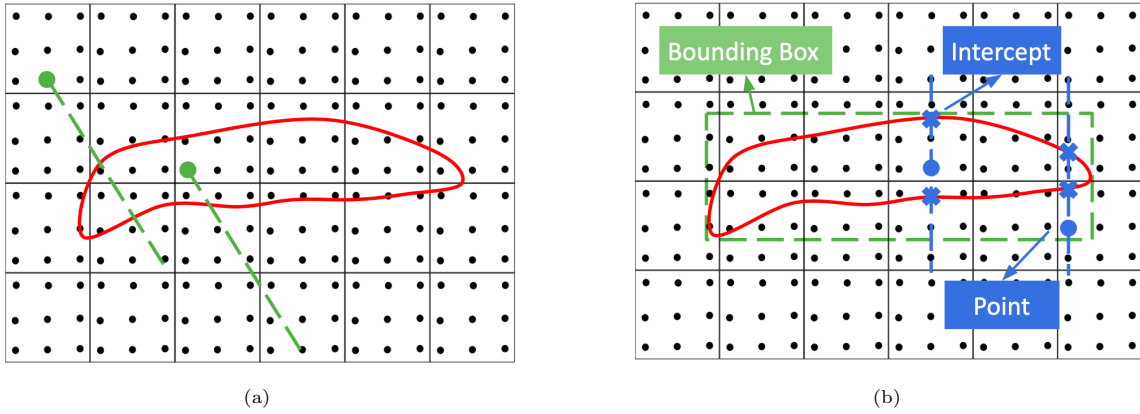


Figure 2: Determination of mask function for general geometries, based on (a) ray casting and (b) the simplified approach used in the present study. The polynomial order for the Cartesian grid is $P = 2$.

For simple geometries like a cylinder, a sphere or an airfoil, analytical shape functions exist and can be

used to define the mask function, as listed in Appendix A. However, in order to handle other geometries whose shape function is difficult to obtain, we still need an algorithm to compute the mask function. Therefore, in the present work, a method to get mask function for general shapes, rather than using the analytical function, has been developed. This method takes an algorithm to identify whether the present solution point is inside or outside the solid body. This forms a typical 'point in polygon' (PIP)¹ problem in computational geometry [67]. PIP problem refers to a set of problems, which determine whether a given point in the plane lies inside, outside, or on the boundary of a polygon. One common approach is the ray casting method. Ray casting approach generates a ray starting from the point and going in any fixed direction, and tests how many times a ray intersects the edges of the polygon. If the point is outside the polygon, the ray will intersect the edges an even number of times. On the contrary, if the point is inside the polygon, the ray will intersect the edges an odd number of times. Schematic illustration of the ray casting method is shown in fig. 2a.

In practice, the ray casting method can be implemented in multiple ways. We take a simplified method to achieve this. For any solution point in the computation domain, we will first define those points that lie outside the bounding box of the surface (i.e., the box formed by coordinates of the rectangular border that fully encloses the solid body) to be the fluid points. For other solution points lie inside the bounding box, we will draw a line along y-axis direction (in 2D) or z-axis (in 3D), as shown in Fig. 2b. After that, the maximum and minimum intercepts in y coordinate (2D) or in z coordinate (3D) are identified. If the corresponding coordinate of the solution point lies in between the maximum and minimum values, then this point lies in the solid and its mask is set to 1. From an implementation point of view, this can be achieved by firstly looking for the nearest IB points along x direction (2D) or x-y plane (3D). After that the minimum and maximum values of these points in y coordinate or z coordinate are easily obtained, and the comparison is subsequently performed. Therefore, the boundary should be sufficiently resolved by the IB points, in order to make sure the the neighboring search involved in getting the mask function is accurate. A limitation of this method is that it is only applicable to convex geometry. For non-convex geometry, more than two intercepts exist. The ray tracing method can be applied to such cases to check how many times the ray intersect the edges along the y or z direction. However, the present approach still works well for the test cases considered in this work.

4.3. Treatment for moving boundaries

This subsection introduces the treatment of moving boundaries (with rigid motion) for penalty IBM method. When the boundary moves, the mask function and the velocity at each solution point should be updated accordingly. The basic idea of updating the mask for each solution point is to first recover the position of this point relative to the static solid body (in the non-inertial reference frame), then determine the mask value based on the recovered position and the mask function defined for the static solid body $\chi(\mathbf{x}, 0)$. The velocity is updated based on the rigid motion equation of the obstacle.

For the cases considered in this study, i.e., a two-dimensional solid body with rigid motion, the translation in x-axis and y-axis, and the rotation motions are included [55]. These displacements are defined respectively as a , b , and θ , following positive x and y direction, and clockwise rotation direction. The initial rotation axis at $t = 0$ is given as (x_r, y_r) . Although the present treatment of moving boundaries is not limited to any

¹https://en.wikipedia.org/wiki/Point_in_polygon

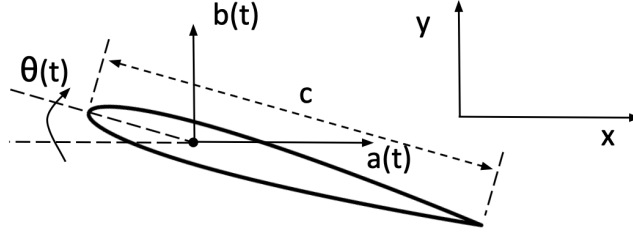


Figure 3: Schematic illustration of an airfoil free to move in translation and rotation directions, with chord length c . a , b and θ are displacements in x and y directions, and the pitching angle, respectively. This definition is applicable to any rigid body.

specific geometry, the schematic illustration of a moving airfoil is shown in Fig. 3 as an example. When the boundary is moving, as the first step, the coordinates are translated into the reference coordinates in the non-inertial reference frame:

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x - a(t) - x_r \\ y - b(t) - y_r \end{pmatrix} + \begin{pmatrix} x_r \\ y_r \end{pmatrix} \quad (36)$$

The mask for any solution point placed in (x, y) is determined based on the transformed coordinates in the non-inertial reference frame, through substituting \tilde{x} and \tilde{y} into the original mask functions at $t = 0$. The solid velocity at each solution point is subsequently updated as follows:

$$\dot{x} = \dot{a}(t) + \dot{\theta}(t)(y - b(t) - y_r) \quad (37)$$

$$\dot{y} = \dot{b}(t) - \dot{\theta}(t)(x - a(t) - x_r) \quad (38)$$

It should be pointed out that this method is not applicable to flexible structures, e.g., when the boundary shape changes with time. In such cases, recovering the position of solution point in the non-inertial reference frame becomes difficult. A solution is to redefine the mask function at each time step based on the updated position of surface points. The velocity of solution points immersed in the solid should also be approximated numerically [57]. In addition, if the movement is too big in one time step, there can be a loss of mass due to the fast transition between fluid and solid state. Some strategies can be considered to handle such problems [36, 68], which will be explored in the future.

4.4. Surface data reconstruction

The reconstruction of data on the solid surface is an important aspect to get the distribution of quantities of interest (e.g., pressure and friction force). The integrated aerodynamic coefficients, like lift and drag, are computed from these quantities. The basic idea to get the surface quantity for each surface point (i.e., IB point) is to perform interpolation based on data of surrounding solution points.

In the high-order framework, it is straightforward to consider using the high-order polynomial defined in each element to perform data reconstruction. Procedures in such cases is very simple: 1) find the element where the current IB point lies; 2) compute the reference coordinate of this IB point in the present element; 3) apply the polynomial interpolation scheme at this reference coordinate. This method, however, has some potential drawbacks that lead to inaccuracy. This is mainly due to the fact that in most of cases, such

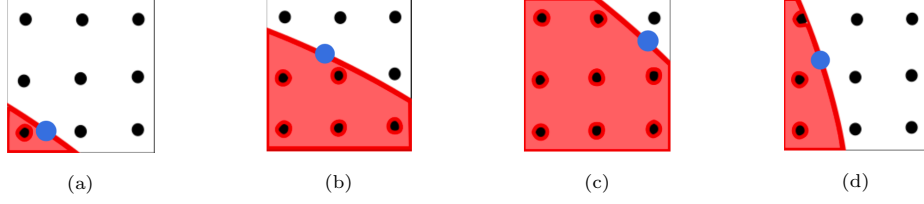


Figure 4: Different locations of solid surface point within a numerical element with polynomial order $P = 2$. Blue circle is the surface marker point. Red region is the solid region. Black points are solution points inside the element.

interpolation formula will involve solution points immersed in the solid, which usually have nonphysical values. The schematic illustration for such problem is given in Fig. 4, where in all cases, nonphysical values at the solid solution point will be involved in interpolation. A comparison between polynomial interpolation and the data reconstruction method used in the present study is shown in Appendix C, illustrating the failure of directly using high-order polynomial for data reconstruction.

Other than polynomial interpolation, an alternative method is to interpolate the data from several nearest solution points. Compared with interpolation based on high-order polynomial, an advantage is that it has more flexibility to select the interpolation points, without involving the solid solution points with nonphysical values. Here, the interpolation methods described in [69] are followed, and are adapted to the high-order framework. In general, the interpolation framework in [69] is based on the inverse distance between the IB point and the interpolation point. Candidate interpolation points are chosen from the nearest fluid points around the IB point. The value and gradient of conservative variables are interpolated. To compute aerodynamic coefficients, the pressure and shear stress are subsequently obtained from the interpolated variables. In particular, the Inverse Distance Weight at Interpolation Point (IDW-IP) method is used for interpolation in the present study.

For standard IDW method, the inverse distance between each surface point and the solution point is used as a weight to compute the value of any variable, as shown below

$$\mathbf{U}_{IB} = \frac{\sum_i \mathbf{U}_i / d_i}{\sum_i 1 / d_i} \quad (39)$$

where \mathbf{U}_i and d_i refer to the solution vector and distance to surface point of the i th. For IDW-IP method, following [70], we first define the interpolation point (IP) as a virtual point close to a specific IB point that lies along the normal of that point. Here the normal can be computed efficiently based on the surface representation method described in Sec. 4.2. The distance of IP to the IB point is defined as

$$d_{IP} = \frac{\sum_i d_{2,i} / d_{1,i}}{\sum_i 1 / d_{1,i}} \quad (40)$$

where d_1 is the perpendicular distance from any solution point to the surface normal of the IB point, while d_2 is the projection of the distance from the solution point to the IB point along the surface normal. These points and distances are illustrated in Fig. 5.

Like IDW method, the data is interpolated from the surrounding solution points but with d_1 as the distance. From the above equation, the interpolation formulation is given as [69]:

$$\mathbf{U}_{IP} = \frac{\sum_i \mathbf{U}_i / d_{1,i}}{\sum_i 1 / d_{1,i}} \quad (41)$$

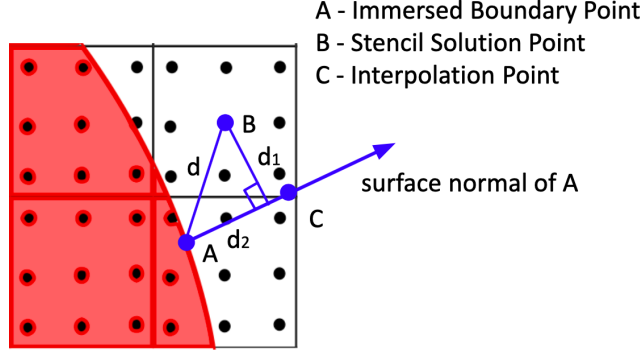


Figure 5: Schematic illustration of data reconstruction method on a Cartesian grid with polynomial order $P = 2$.

Interested readers can refer to [69] for more details. After extensive testing, we found a good balance between accuracy and efficiency is to choose the nearest $2N_p \sim 3N_p$ flow points, where N_p refers to the number of solution points for near-wall elements. The integrated aerodynamic loads, including the lift and drag coefficients, is obtained from the reconstructed data at IB points. In particular, the reconstructed surface pressure P and shear stress τ_{ij} are used. The boundary representation discussed in Sec. 4.2 allows getting the surface normal and area efficiently. The distribution of surface points for force calculation is sufficiently dense to guarantee the accuracy of integration. The lift and drag coefficients are given as:

$$C_l = \frac{1}{(1/2)\rho_\infty U_\infty l} \int_{\partial S} (\tau_y \mathbf{n} - P_y) dS \quad (42)$$

$$C_d = \frac{1}{(1/2)\rho_\infty U_\infty l} \int_{\partial S} (\tau_x \mathbf{n} - P_x) dS \quad (43)$$

where l is the characteristic length.

The implementation of data reconstruction (IDW-IP method) in the high-order framework is as follows: 1) locate the element where the current IB point lies; 2) find neighboring elements of this element (and neighbors of the neighboring element) and get the coordinates and masks for all the solution points in these elements; 3) compute the distance of all candidate solution points to the IB point, and rank the points based on the distance in increasing order; 4) select the points according to the rank, and discard the points immersed in the solid, until the preset number of stencil points is reached; 5) compute the distance d_1 for all candidate stencil points, and compute the weighting coefficients for each stencil point based on Eq. 41.

4.5. Overview of the algorithm

The proposed IBM approach based on volume penalization and high-order FR can be summarized in Fig. 6. Overall, the following procedures are needed:

(1) Import the background mesh and IBM geometry. The geometrical information is extracted from the IBM marker points, including the surface normal and surface area. The distance to the surface is also needed when using smooth strategy for the mask Appendix B. The reconstruction stencil is computed for each marker point, respectively. For static boundary, these operations are only performed once in the simulation.

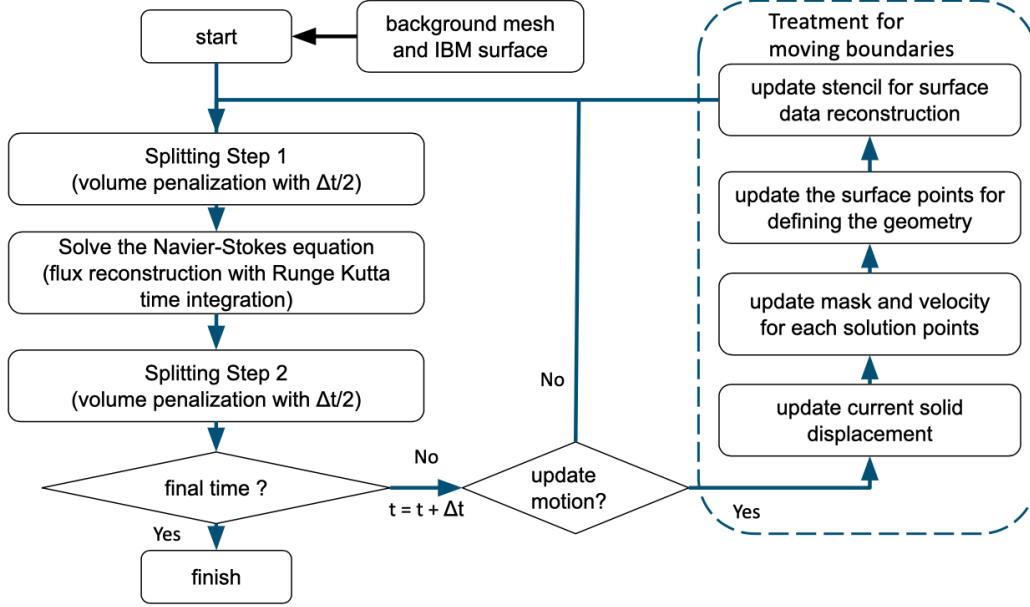


Figure 6: Flow chart of the proposed method.

(2) At each time step, perform the first splitting step, Runge-Kutta explicit time integration, and the second splitting step accordingly.

(3) When the solid body is moving, additional update operations are required after every time step (or every several time steps). The solid displacement is first updated based on the prescribed motion function. This displacement is used to update the mask and the velocity for each solution point, based on the method in Sec. 4.3. To evaluate the aerodynamic coefficients under moving boundaries, the surface IB points are updated, along with its normal vector. With the updated position of IB points, data reconstruction also needs to be reset.

4.6. Discussions on error estimate and selection of penalization parameter

One of the advantages of penalization method over other IBM approaches is that rigorous proofs of the convergence have been given [37, 50]. Therefore, the numerical error introduced from the penalization term can be controlled a-priori [54]. The error of the numerical solution of the penalized problem corresponding to the original problem includes two parts [57], the penalization error and the discretization error:

$$\|u^{exact} - u_\eta^N\| \leq \|u^{exact} - u_\eta\| + \|u_\eta - u_\eta^N\| \quad (44)$$

where u^{exact} is the exact analytical solution of the governing equations, u_η and u_η^N are the exact and numerical solution of the penalized equations. $\|\cdot\|$ is the norm used for quantifying the error, e.g., L_1 , L_2 or L_∞ norm. The first part of error is the penalization error depending on the penalization parameter [42]:

$$\|u^{exact} - u_\eta\| \propto \eta^\alpha \quad (45)$$

It should be emphasized again that the physical interpretation of volume penalization is that the solid obstacle is assumed to be a porous medium with sufficiently small permeability η , thus the velocity of the

surrounding fluid tends to be zero and vanishes at the interface. Therefore, the convergence for the solution of penalization method to the exact solution requires the error norm to approach zero for small penalization parameter limit, i.e., $\lim_{\eta \rightarrow 0} \|u^{exact} - u_\eta\| \rightarrow 0$. In order to achieve this, we need $\alpha > 0$. Fortunately, theories based on rigorous mathematics have been proposed to validate that volume penalization method satisfies this requirement. From Angot et al. [37] and Carbou and Fabrie [50], the volume penalization gives $\alpha = \frac{1}{2}$, indicating the penalization error has a decay rate of $\mathcal{O}(\sqrt{\eta})$ for Dirichlet boundary condition. For Neumann boundary condition, $\mathcal{O}(\eta)$ can be obtained [71].

The discretization error refers to the error between the exact solution and the numerical solution of the penalized equations. With consistent discretization and a stable numerical scheme, the discretization error usually follows ($\beta > 0$):

$$\|u_\eta - u_\eta^N\| \propto N^{-\beta}. \quad (46)$$

However, as pointed out by Schneider et al. [47] [42], the discretization error is not only determined by the numerical scheme, but also limited by the regularity of the solution. Regularity is characterized by the smoothness of the exact solution u_η at the boundary of the penalized problem. Therefore, the order of convergence β becomes the minimum order between the numerical scheme and the regularity of the exact penalized solution. For high-order method, the error of the numerical scheme can be reduced by performing mesh refinement (h-refinement) or increasing polynomial order (p-refinement). However, the low regularity of the solution near the wall for the penalized equation still remains a limitation to the present method.

From the error estimate, it is suggested to use a very small penalization parameter η to minimize the penalization error. However, small η will lead to very stiff source term, thus causing stability issues. This is the motivation of using splitting approach, as discussed in Sec. 4.1. In practice, the penalization parameter is dependent on the numerical resolution, where smaller penalization parameter requires a smaller time step and finer resolution near the wall. When the penalization parameter is treated explicitly, from the linear stability analysis [55], the time step for explicit time integration must be smaller than the penalization parameter $\Delta t < \eta$. This condition can be relaxed by using the Strang splitting method in the present study, and can be further relaxed with the implicit forcing approach mentioned in Sec. 4.1. Also in [72], the intimate coupling between Δt and η is observed, indicating the error will saturate when $\Delta t \approx \eta$. This explains why it is usually suggested to use $\Delta t = \eta$ in the volume penalization method. An interpretation is that the penalty term acts as a strong damping term with order η on the velocity, which has to be resolved by the time discretization scheme [57]. From a practical point of view, one can select the maximum Δt which allows $\Delta t = \eta$ to maintain both computational efficiency and accuracy. Note that in high-order methods, the time step Δt is determined by the Courant-Friedrichs-Levy (CFL) condition, which scales as the inverse of the spatial order squared [73]. Therefore a practical guideline is to fix $\Delta t = \eta$ first and determine Δt according to stability criterion.

5. Test Cases

In this section, the proposed IBM based on volume penalization for high-order flux reconstruction framework is validated by different test cases with increasing complexity. Analysis of one-dimensional equation is firstly performed to study the convergence of the method. Simulation of flow past static obstacles in

two-dimension and three-dimension is then shown. Finally, the capability of treating moving boundaries is validated.

5.1. One-dimensional advection-diffusion equation

The convergence behavior of the proposed method is tested in this subsection. A one-dimensional advection-diffusion is considered as the test case:

$$\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} + \frac{\chi}{\eta}(u - u_b) = 0, x \in [0, 1] \quad (47)$$

The advection speed and diffusivity coefficient are chosen as $c = 0.1$ and $\nu = 0.01$, respectively. The initial condition and the homogeneous Dirichlet boundary conditions to be imposed are shown as follow:

$$u(x, 0) = \exp(5x)\sin(\pi x), u(0, t) = 0, u(1, t) = 0. \quad (48)$$

The analytical solution of this problem is given by:

$$u(x, t) = \exp(5x - t(0.01\pi^2 + 0.25))\sin(\pi x) \quad (49)$$

We choose a constant time step $1e-7$ to ensure time accuracy. The penalization parameter $\eta = \Delta t$. For the diffusion term, the β and τ used for LDG scheme are 0.5 and 0.1. We march the solution in time to $t_{max} = 0.01$, based on the third-order TVD Runge-Kutta scheme. The convergence with respect to penalization parameter is firstly tested, as shown in 7. The computational domain is discretized by 80 elements, and 2 additional elements are extended to both sides to impose the penalized boundary conditions. The solution points in the solid are penalized by the values at two boundaries, i.e., homogeneous boundary conditions $u = 0$, which are known to be the exact boundary condition we want to impose at the interface. The error of the simulation is quantified by the L_2 error norm between the analytical solution u and the approximated solution u^N within the whole fluid region. The convergence plot is given in Fig. 7. It can be seen that as the penalization parameter η approaches zero, the accuracy limit of the penalization method will be reached. The convergence rate of $\mathcal{O}(\sqrt{\eta})$ is recovered at large η , which agrees with the theory of penalization method for Dirichlet problem in the continuous setting [37] [50]. In addition, as P increases, a smaller η is required to get the best accuracy. When P is larger than 1, a super-convergence larger than the theoretical limit is also observed, where the convergence rate becomes $\mathcal{O}(\eta^{-1})$. This is different from numerical tests of low-order scheme, where $\mathcal{O}(\eta^{-0.5})$ is observed [74] [57].

The convergence with respect to the spatial resolution is then studied. To test the numerical convergence in detail, we consider two situations: 1) when the analytical solution in the solid is known (e.g., Eq. (49) is known a-priori), solutions at each solution point are penalized by its analytical solution respectively; 2) when the analytical solution inside the solid is not known, but only the values at the boundary are known (e.g., imposing the homogeneous Dirichlet boundary condition in Eq. (48) for all solid solution points). In this latter case, we penalize all the solution points immersed in the solid with $u = 0$. The L_2 error norm in the fluid is used to quantify the error of numerical scheme. It is obvious that the latter case is more realistic since in practice the behavior of the solutions inside the solid is unpredictable, therefore they should also adopt the boundary condition at the interface.

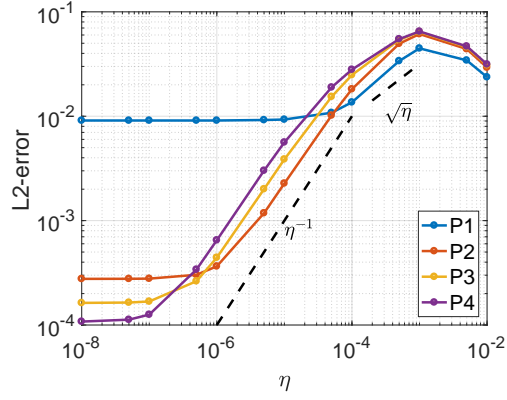


Figure 7: Convergence with penalization parameter η for penalized advection-diffusion equation. P is the polynomial order.

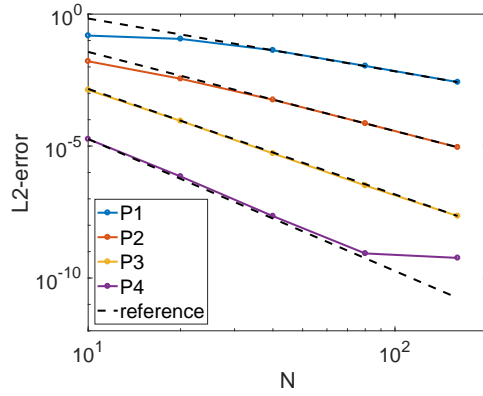
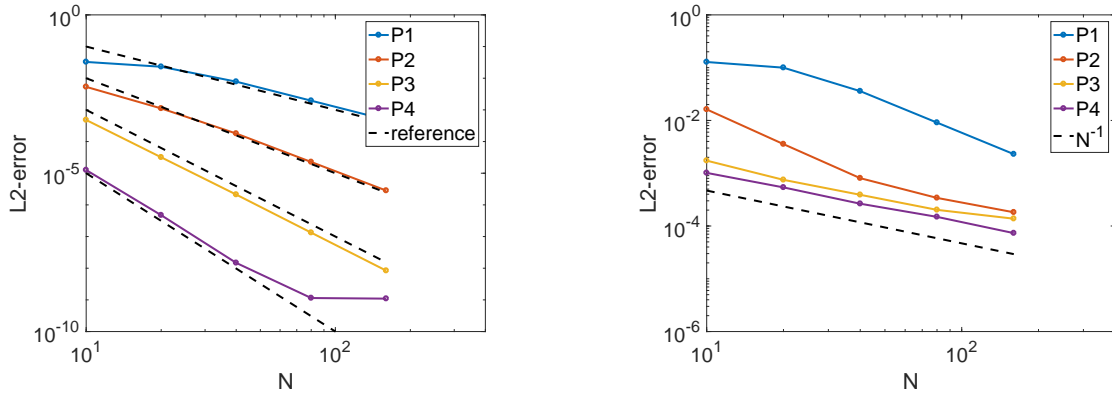


Figure 8: Convergence with the number of elements for penalized advection-diffusion equation. Solid solution points are penalized by the analytical solution. P is the polynomial order. Reference lines give the expected convergence rate of flux reconstruction scheme $N^{-(P+1)}$.

The convergence plot of the first case is shown in Fig. 8. As shown in the figure, when the analytical solutions for all solid points are used in the volume penalization method, an order of convergence $N^{-(P+1)}$ is recovered. This is the exact convergence rate of standard FR scheme. This indicates that the volume penalization method itself does not deteriorate the high-order convergence from high-order schemes. Convergence plot of the second case, which is more realistic, is shown in Fig. 9. From Fig. 9a, in the flow region far from the wall $x \in [0.2, 0.8]$, the theoretical convergence rate $N^{-(P+1)}$ can still be recovered. However, for the global fluid domain $x \in [0, 1]$, the convergence rate is reduced to approximately $\mathcal{O}(N^{-1})$ across all polynomial orders, as indicated in Fig. 9b. This is because when homogeneous Dirichlet boundary conditions are imposed for all solution points, the smoothness (or regularity) of the gradient across boundary cannot be guaranteed. As explained in Section 4.6, this low regularity of the solution near the wall will limit the overall convergence [47] [60]. Such limitation of convergence rate was also observed [54] [55] and discussed [42] [57] in previous works. The regularity can be improved by using larger penalization parameter, but this will affect the accuracy near the wall [47]. In addition, due to the fact that in the flow region far from the wall, high-order accuracy can be recovered, it is preferable to perform local refinement near the wall to increase the accuracy, without costing to many additional degrees of freedom. Compared with the first case, it can be concluded that once the exact solution of the original equations in the solid is known, the regularity near the wall is well kept, thus better global convergence rate is recovered. Therefore, the error will only come from the numerical scheme, where the convergence of standard flux reconstruction method in the global region is retained. This case helps to investigate only the influence of spatial and temporal discretizations, and can also be used to test the correctness of code implementation with penalization method. This also directs the future development of the present method, where some strategies can be sought to improve the regularity near the wall, thus improving the overall convergence.



(a) Order of convergence for the computational domain far from boundary $x \in [0.2, 0.8]$ (b) Order of convergence for the whole computational domain including boundary $x \in [0, 1]$

Figure 9: Convergence with number of elements for penalized advection-diffusion equation. Solid solution points are penalized by the homogeneous Dirichlet condition. P is the polynomial order. Reference lines give the expected convergence rate of flux reconstruction scheme $N^{-(P+1)}$.

5.2. Flow past a cylinder

The flow past a cylinder is a standard test case for IBM simulation. Therefore, in the present study, it is chosen as the first test case to investigate the proposed method for the Navier-Stokes equations. The Reynolds and Mach numbers are set to 40 and 0.2, where the flow remains steady. The no-slip adiabatic

wall boundary condition is considered for the wall, therefore we impose $u_s = 0$ and $v_s = 0$ in the solid. Here three sets of mesh are considered. The size of rectangular computational domain is $x \in [-30D, 50D]$ and $y \in [-30D, 30D]$, where D is the diameter of the cylinder and is set to 1. In the square region $x \in [-D, D]$ and $y \in [-D, D]$, uniform grid is used. The uniform mesh size h is chosen as $0.03D$ (Mesh 1), $0.015D$ (Mesh 2) and $0.01D$ (Mesh 3), respectively. Mesh 1 is illustrated in Fig. 10. The number of grid points are 187×178 , 325×313 and 410×400 . Thanks to the inner degree of freedom given by high-order methods, the mesh size is relatively coarse compared with existing works based on low-order methods. For example, in a recent IBM work based on finite difference method [75], grid with size 800×320 is used for the same case. With a decent mesh, the present method allows to increase the resolution based on high-order polynomial approximation. We choose the polynomial order $P = 3$ for Mesh 1, $P = 2$ for Mesh 2 and Mesh 3. The explicit time steps used for these cases are $7e-5$, $1e-4$, and $5e-5$, respectively. The penalty parameter is chosen to be equal to the time step. The characteristic boundary conditions are imposed to all the far field boundaries.

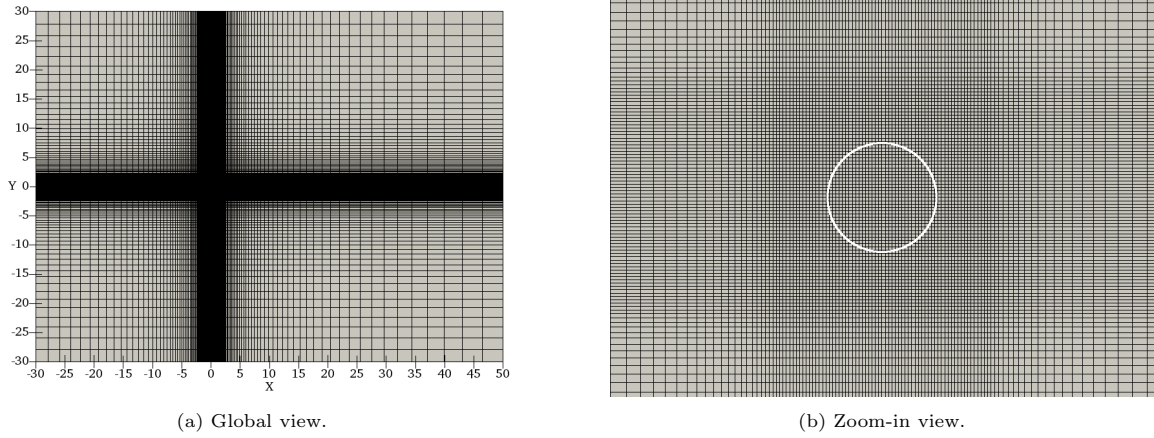


Figure 10: The computational mesh (Mesh 1, locally uniform grid size $h = 0.03$)

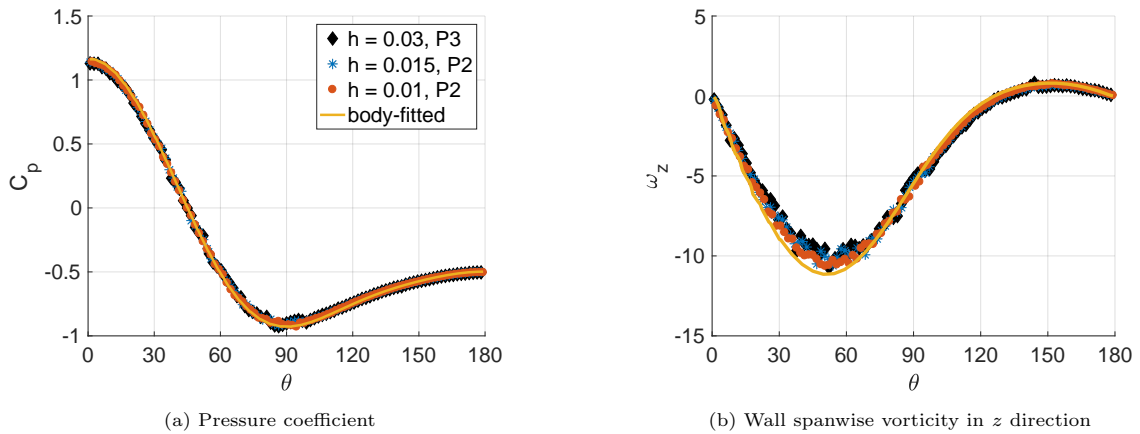


Figure 11: Comparison of variables for flow past a cylinder at $Re = 40$ and $M = 0.2$. h and P are locally uniform grid size and the polynomial order, respectively.

The pressure coefficient C_p and wall spanwise vorticity ω_z are compared in Fig. 11. The results are also compared with those from body-fitted simulation with the same solver. A detailed comparison of quantity

Table 1: comparison of reattachment length, separation angle and drag coefficient for flow past a cylinder

Case	L/D	θ	C_d
Dennis and Chang [76]	2.35	53.8	1.52
Fornberg [77]	2.24	55.6	1.50
Choi et al. [70]	2.21	53.6	1.49
body-fitted	2.24	51.0	1.53
Mesh 1	2.30	51.2	1.50
Mesh 2	2.30	52.0	1.51
Mesh 3	2.27	52.0	1.52

of interest is given in Table 1. All results give good agreement for the pressure coefficient distribution. However, the prediction of spanwise vorticity is not as good as C_p . This is mainly due to the difficulty in predicting the gradient values on the surface, since the present method only impose the constraint to the surface velocity. To solve this, better refined resolution or other treatment to impose the gradient condition can be a good choice for future works. But from Table 1, good agreement with the literature is shown, and as we increase the resolution, the results also become closer to the reference data. A typical flow snapshot is shown in Fig. 12.

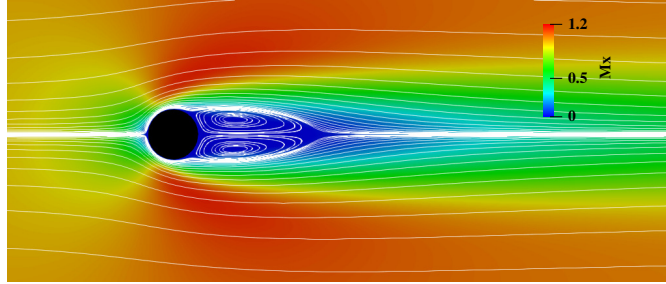


Figure 12: The x -momentum field and the streamlines for flow past a cylinder at $Re = 40$ and $M = 0.2$.

In addition, we also want to highlight the advantage of high-order method in performing local p-refinement near the wall. Local refinement of the polynomial order comes from the flexibility of high-order framework, where the mesh remains unchanged but the resolution inside the element can be improved. This will keep good accuracy with a much reduced degree of freedom, thus reducing the overall computational cost. Although it is not the main focus of the present work, we also investigated the efficacy of local p-refinement near the wall. The p-refinement is implemented based on the mortar method [78], where a mortar element is introduced to the element interface. The common and interaction fluxes are computed on the mortar element and are projected back to each neighboring face. We tested this approach based on the first mesh, where the global order $P = 1$ is used for elements in the farfield. Local p-refinement near the wall is performed to improve the polynomial order locally to $P = 3$. The distribution of polynomial order across each element is shown in Fig. 13a. To implement local p-refinement, we firstly measure the distance between each cell center and the surface. When this distance is smaller than $0.25D$, the polynomial order of that element is increased to $P = 3$. A buffer layer with $P = 2$ is also included between $P = 1$ and $P = 3$ cells. From the results in Fig. 13b, it is clear that with local p-refinement, we can produce very close results compared with globally high-order computation. The total degree of freedom, defined by the total number of solution points, is

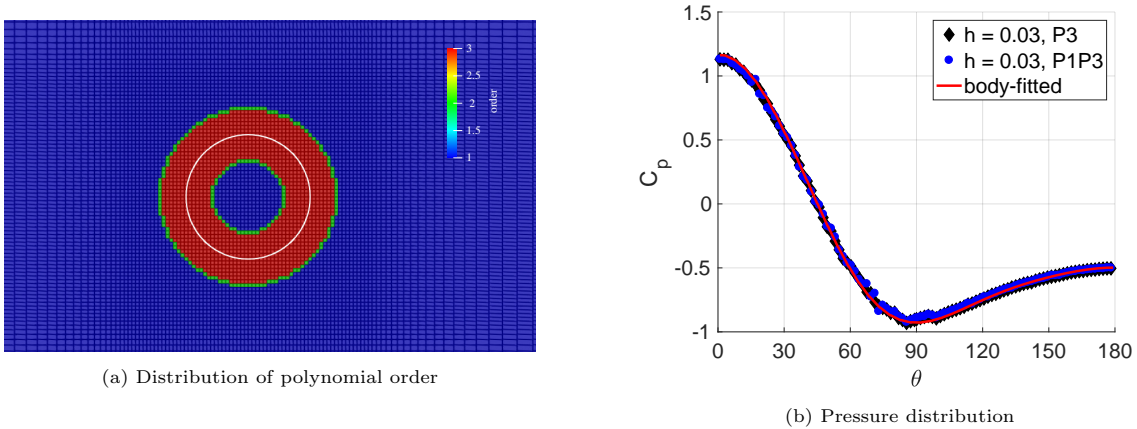


Figure 13: Comparison of variables for flow past a cylinder at $Re = 40$ and $M = 0.2$. The local p-refinement is considered near the wall. The white curve represents the solid boundary. h and P are locally uniform grid size and the polynomial order, respectively. $P1P3$ refers to the local p-refinement case with global order 1 and local order 3 near the wall.

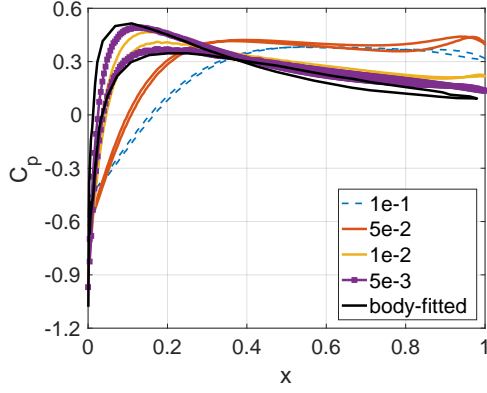
148296 for the p-refinement case. This is nearly one quarter of the degree of freedom 532576 required for
the test case with $P = 3$ cells globally. This indicates the advantage of combining high-order methods with
IBM. The p-adaptation framework [79, 80] with proper adaptation strategy can be further considered to
increase the resolution on any flow region of interest.

5.3. Flow past a NACA0012 airfoil

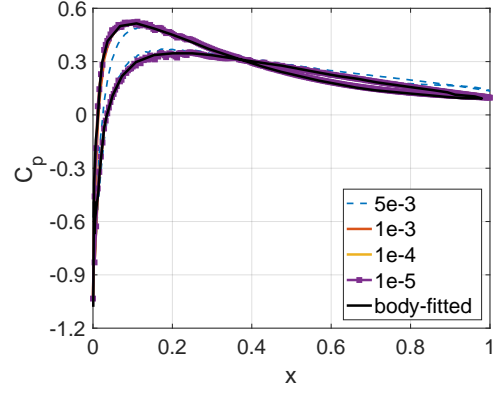
The flow over an airfoil is tested to evaluate the present method for a configuration with higher Reynolds
number and a non-zero angle of attack. The benchmark of NACA0012 airfoil at Mach number 0.5 and
Reynolds number 5000 is chosen, with an angle of attack 2 degree. The size of rectangular computational
domain is $x \in [-30c, 50c]$ and $y \in [-30c, 30c]$, where c is the chord length of the airfoil. In the square region
 $x \in [-c, c]$ and $y \in [-c, c]$, uniform and square grid with mesh size $0.004c$ is used. The number of mesh
elements are 897×697 . The characteristic boundary conditions are imposed to all the far field boundaries.
The reference data is obtained from a body-fitted simulation of the same solver.

The influence of penalization parameter, is firstly studied for this case, as shown in Fig. 14. Here for all
the cases, the polynomial order is selected as $P = 2$ with a constant time step $1e - 4$. As the penalization
parameter decreases, the results become more accurate. This is consistent with the error estimate that the
penalization parameter should be small enough to ensure good accuracy. In addition, the smoothness of
the solution becomes worse as penalization parameter decreases. It can be seen that when the penalization
parameter is smaller than $\Delta = 1e - 4$, the solution accuracy does not seem to improve anymore, but becomes
slightly more oscillation, as shown in the $\eta = 1e - 5$ case. This highlights the argument that the error saturates
around $\eta \approx \Delta t$ [72] [57].

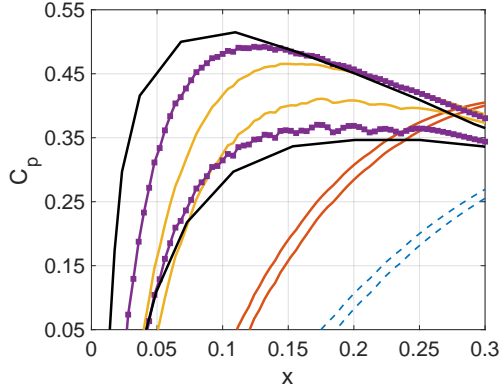
To compare the solution with refined resolution, two more test cases are simulated. For the same mesh
used, the polynomial order is increased to $P = 4$, with time step and penalization parameter set as $2e - 5$. In
addition, a locally refined mesh with size $0.001c$ is also generated, with mesh size 1341×365 . The simulation
with $P = 2$, $dt = 1e - 5$ and $\eta = 1e - 5$ is also added. As shown in the Fig. 15, all simulations give very good
prediction on C_p . The prediction of surface skin friction coefficient is more difficult, since it involves the
reconstruction of gradient. From Fig. 15, we can also see that as the resolution near the wall increases, C_f



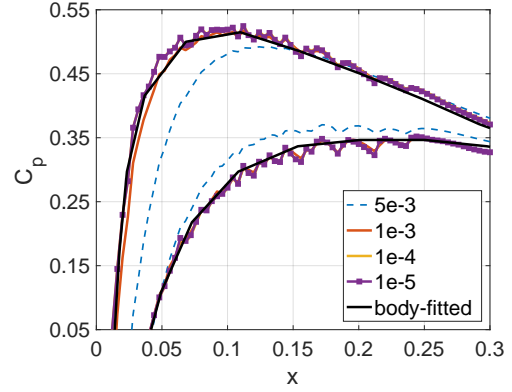
(a) Larger penalization parameter



(b) Smaller penalization parameter



(c) Larger penalization parameter, zoom-in view



(d) Smaller penalization parameter, zoom-in view

Figure 14: Influence of penalization parameter η for flow past a NACA0012 airfoil at $Re = 5000$ and $M = 0.5$, illustrated by C_p distribution. h and P are locally uniform grid size and the polynomial order, respectively.

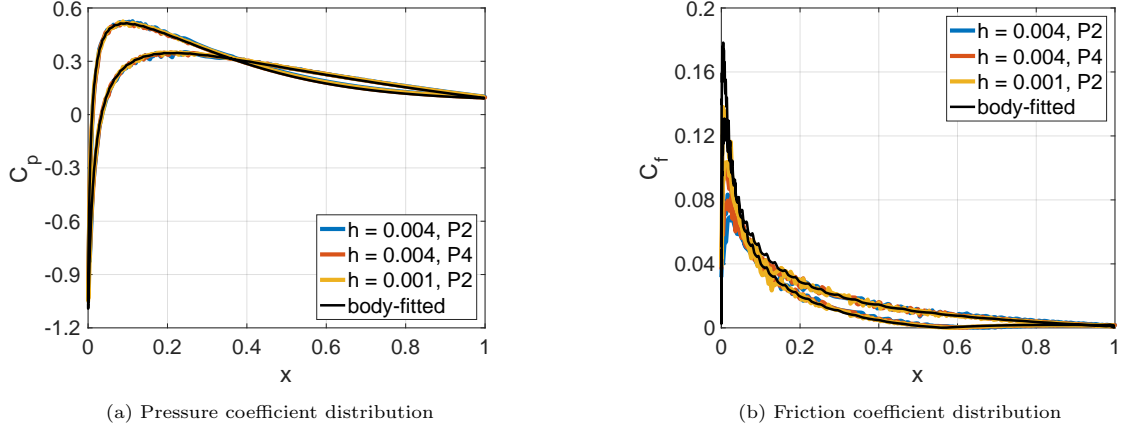


Figure 15: Comparison of pressure and friction coefficients with different h and p resolutions, for flow past a NACA0012 airfoil at $Re = 5000$ and $M = 0.5$. h and P are locally uniform grid size and the polynomial order, respectively.

is better predicted. Note that the increase in resolution can be achieved by increasing either the polynomial order or the number of elements locally. Note that for low-order schemes, we need a resolution about mesh size $5e - 4c$ to have a comparable prediction on friction coefficient [69].

5.4. Flow past a sphere

In order to test the proposed method in simulating three-dimensional flows, the flow over a sphere at Reynolds number 100 and Mach number 0.2 is chosen. To reduce the overall computational cost, we generate the mesh for a quarter of the whole domain. The size of domain is $x \in [-30D, 30D]$, $y \in [0, 30D]$ and $z \in [0, 30D]$, where D is the diameter of the sphere. The symmetric plane is considered as boundary condition on $y = 0$ and $z = 0$ plane, while characteristic boundary conditions are imposed to all the other boundaries. To ensure sufficient resolution near the wall, uniform grid with size $0.03D$ is used in region $x \in [-0.6D, 0.6D]$, $y \in [0, 0.6D]$ and $z \in [0, 0.6D]$. This results in a total element number of $129 \times 55 \times 55$. The polynomial order is set to 2. The time step and penalization parameter are chosen as $5e - 5$.

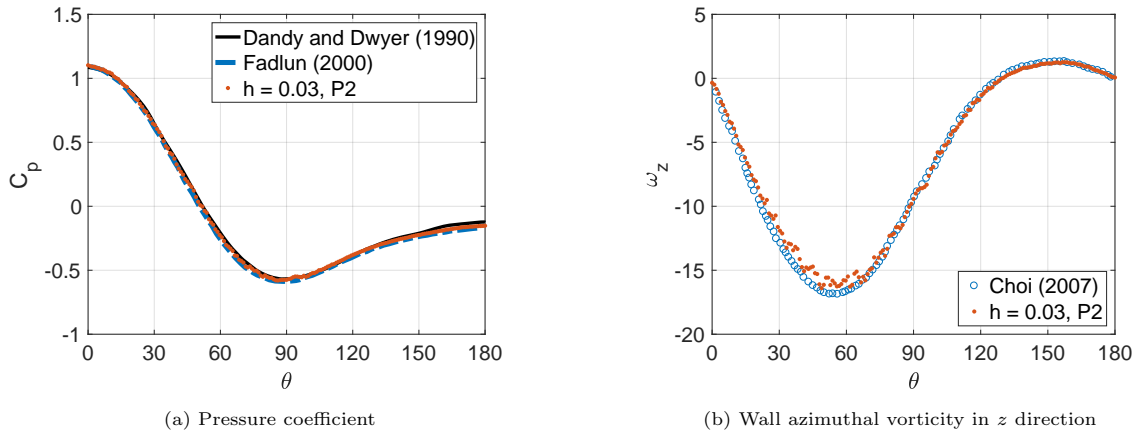


Figure 16: Comparison of variables for flow past a sphere at $Re = 100$ and $M = 0.2$. h and P are locally uniform grid size and the polynomial order, respectively. Reference data are taken from Dandy and Dwyer (1990) [81], Fadlun (2000) [35] and Choi (2007) [70].

The surface quantities from the symmetrical plane $z = 0$ are used for comparison. The resulting pressure coefficient distribution is compared with other studies from [81] and [35], which are shown in Fig. 16. From the comparison of pressure coefficient, all results nearly collapse, thus indicating the good accuracy from the present method. The comparison of wall azimuthal vorticity is shown with the IBM results from Choi et al. [70]. A good agreement is also observed, except slight oscillation around the separation position. As shown in previous study, this can be reduced by further refining the resolution near the wall. The drag coefficient is about 1.06, which is only about 2% error compared with exist results (1.08 from [82] and 1.09 from [83]).

5.5. Flow past a pitching and plunging airfoil

In this subsection, a moving airfoil with combined plunging and pitching motion is simulated with the proposed method. This configuration has been extensively considered in flapping wings [84] and FSI analysis [85]. The present case is taken from Case 2 of the 5th International Workshop on High-Order CFD Methods [86].

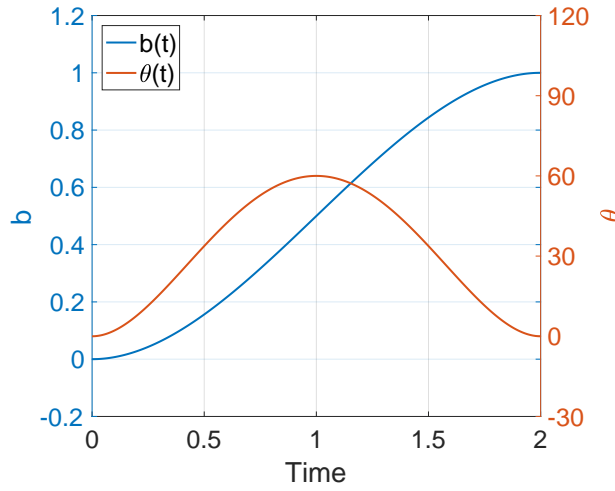


Figure 17: Evolution of motion evolution with time. b and θ are plunging displacement and pitching angle, respectively.

The Mach number is 0.2 and the Reynolds number is 1000. The size of rectangular computational domain is $x \in [-30c, 60c]$ and $y \in [-30c, 30c]$, where c is the chord length of the airfoil. In the rectangular region $x \in [-0.8c, 0.8c]$ and $y \in [-0.3c, 1.3c]$, uniform and square grid with size $0.005c$ is used. This results in a total number of 652×466 elements. The characteristic boundary conditions are imposed to the far field. The rotation axis is placed at $1/3$ chord length at airfoil centerline. The simulation is performed across polynomial orders 1 to 3, with time step set to $6.0e - 5$, $4.0e - 5$ and $2.5e - 5$, respectively. As usual, the penalization parameter is the same as time step for all simulations. The plunging and pitching motions are defined by the following equation:

$$\begin{cases} b(t) = t^3(-8t^3 + 51t^2 - 111t + 84)/16 \\ \theta(t) = (80\pi/180)t^2(t^2 - 4t + 4) \end{cases} \quad (50)$$

The displacement versus time is shown in Fig. 17. The airfoil keeps moving upward, with an impulse type pitching motion. The comparison of force coefficient in x and y directions is shown in Fig. 18. A good

agreement with the reference data produced from body-fitted simulation [86] is observed, where all IBM
simulations are quite close thus is not distinguishable from the figure.

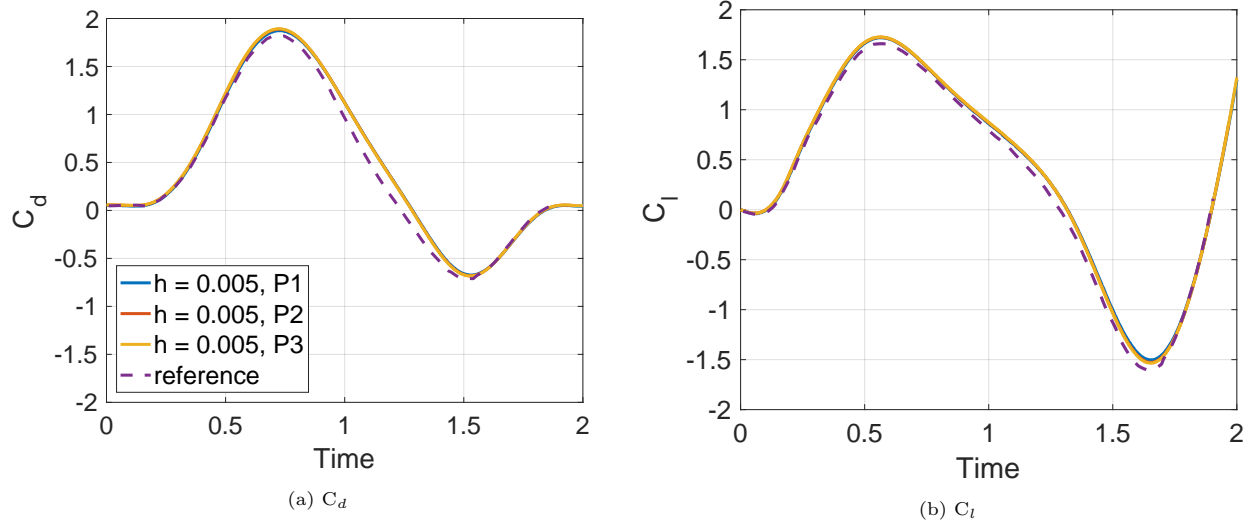


Figure 18: Force comparison of combined plunging and pitching motion. h and P are locally uniform grid size and the polynomial order, respectively.

In order to see how the solution converges as we increase the polynomial order, a zoom-in view of force
evolution is shown in Fig. 19. It can be seen that, as the polynomial order increases, the result will approach
 $P = 3$ simulation. The smoothness of solution is also improved as we increase the polynomial order. This
is in consistent with the high-order framework where the higher polynomial order leads to better resolution
and better accuracy. The flow fields based on $P = 2$ simulation is visualized in Fig. 20. From this figure,
the vortex shedding pattern for such energy extraction process can be accurately described.

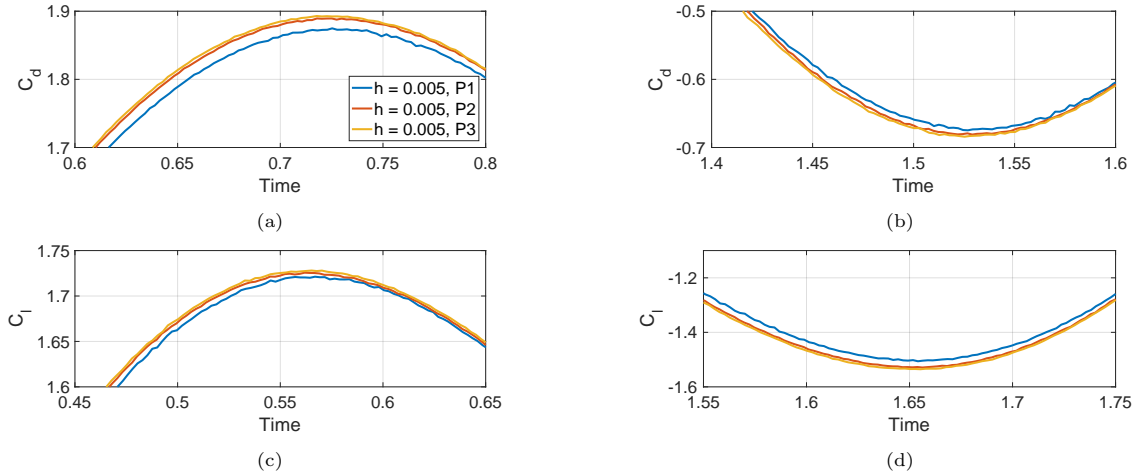
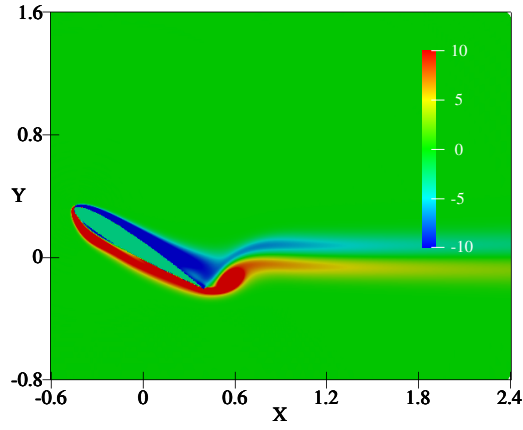
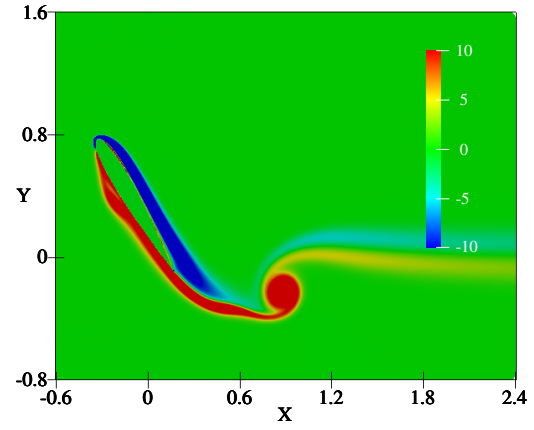


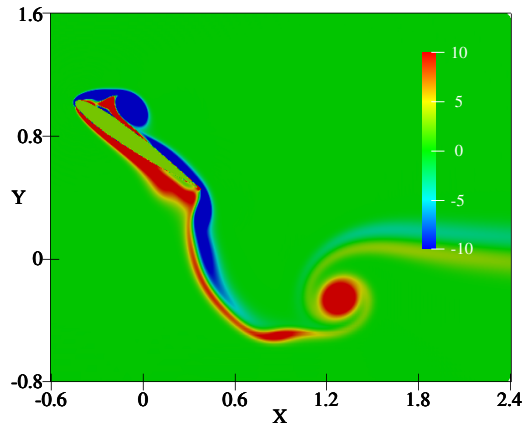
Figure 19: Detailed comparison of force across different polynomial orders. h and P are locally uniform grid size and the polynomial order, respectively.



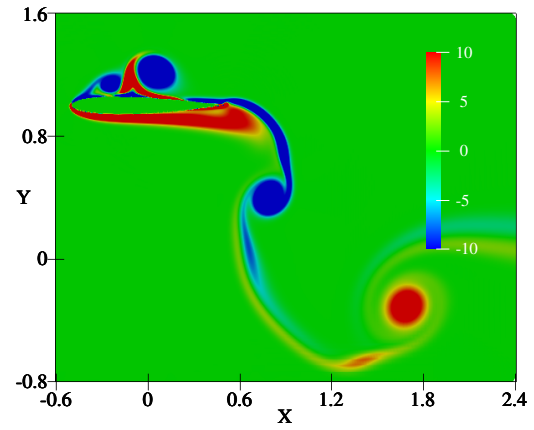
(a) Time = 0.5



(b) Time = 1.0



(c) Time = 1.5



(d) Time = 2.0

Figure 20: Flow snapshots at four time instants for combined pitching and plunging motion of the airfoil. The locally uniform grid size is $h = 0.005$ and the polynomial order is $P = 2$.

6. Conclusions

An immersed boundary method based on volume penalization for flux reconstruction method is proposed. It comes from the aspiration to use high-order DG type methods to efficiently handle the complex geometries on Cartesian grid and to alleviate the sometimes difficult meshing procedure. FR approach is selected due to its capability of recovering existing high-order schemes. The volume penalization method, which is proven to show convergence with low penalization parameter limit, is introduced as the IBM approach. Our implementation is general and does not assume knowing the analytical shape of the geometry, which enables generalisation to complex three-dimensional simulations. In order to overcome the numerical stability issue coming from stiff IBM source term, the splitting scheme with explicit and implicit forcing methods are proposed. Efficient and accurate methods for reconstructing flow quantities near the wall are used to get the surface pressure and friction distribution. Through the one-dimensional advection-diffusion equation, the convergence behavior is investigated. It highlights the importance of increasing resolution near the wall and the use of small penalization parameter to impose boundary condition properly. The accuracy of the proposed method has been validated through different real flow cases including flow past static cylinder, airfoil and sphere, as well as unsteady flow past a moving airfoil. Good agreement with body-fitted simulation and existing literature is shown, indicating the validity of the proposed method. In addition, the high accuracy of the flux reconstruction method provided when using high polynomials is evident in the results, and particularly when computing surface functionals (e.g. lift, drag), in both static and moving geometries. Further work will extend and enhance the present method to compute complex three-dimensional turbulent flows in the FSI context.

Conflict of Interest

The work presented in this paper does not have any conflict of interest with other organizations.

ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement (MSCA ITN-EID-GA ASIMIA No 813605).

Appendix A. Analytical mask function for different geometries

Although the present work has developed a method to calculate mask function for arbitrary geometries, analytical mask functions for the geometries involved in the present study are also given for completeness. For a cylinder, a sphere, and a NACA0012 airfoil with unit length, whose center point is located at $(0, 0, 0)$, the mask function is given as follows:

For cylinder:

$$\chi(\mathbf{x}, t) = \begin{cases} 1, & \text{if } x^2 + y^2 \leq 0.5^2 \\ 0, & \text{otherwise} \end{cases} . \quad (\text{A.1})$$

For sphere:

$$\chi(\mathbf{x}, t) = \begin{cases} 1, & \text{if } x^2 + y^2 + z^2 \leq 0.5^2 \\ 0, & \text{otherwise} \end{cases} . \quad (\text{A.2})$$

For NACA0012 airfoil:

$$\chi(\mathbf{x}, t) = \begin{cases} 1, & \text{if } y^2 - (a_1(a_2\sqrt{(x+0.5)} + a_3(x+0.5) + a_4(x+0.5)^2 + a_5(x+0.5)^3 + a_6(x+0.5)^4))^2 \leq 0 \\ 0, & \text{otherwise} \end{cases} \quad (\text{A.3})$$

the coefficients for NACA0012 airfoils are $a_1 = 0.594689181$, $a_2 = 0.298222773$, $a_3 = -0.127125232$, $a_4 = -0.357907906$, $a_5 = 0.291984971$, and $a_6 = -0.105174606$.

Appendix B. Smoothing the mask function

As mentioned in [55, 57], smoothing the mask function enables to avoid spurious oscillations of the hydrodynamic forces on moving obstacles, and also helps for de-aliasing of the penalization term. In order to test how smoothing mask function helps with the present solver, we make a test on simulating the NACA0012 airfoil for the same case considered in Sec. 5.3. Following [55], we use a Gaussian function to smooth the sharp mask function χ_{sharp} :

$$\chi_{smooth} = [1 - \exp(-(x_{dist}/\delta)^2)] \cdot \chi_{sharp} \quad (\text{B.1})$$

where x_{dist} refers to the distance of solution point to the surface, δ is the width of the smoothing function. As an example, the 1D smooth mask function with different width parameters are shown in Fig. B.21. As the solution point is approaching the surface, less penalization is imposed to this point.

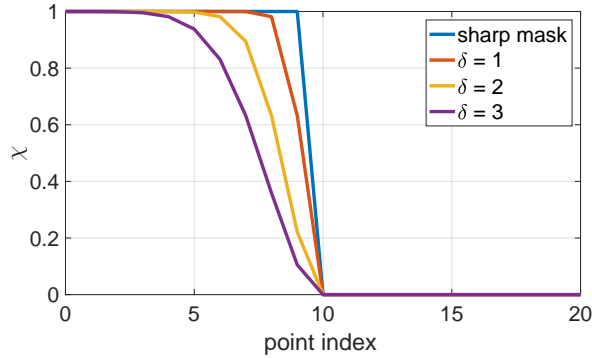


Figure B.21: Smooth mask function with different width.

We select the $P = 2$ test case, with penalization parameter $1e - 5$. Two width parameters, $h/3$ and $h/8$ are considered. The results are shown in Fig. B.22. It is observed that with the smooth mask function, the pressure distribution becomes less oscillatory, while some error will also be introduced near the leading edge of the skin friction coefficient. This is related to the smaller source term near the wall, where the accuracy may be reduced but the smoothness of the solution can be improved. We can also see that as a larger width parameter is selected, the accuracy of the result becomes worse. This comparison shows that smoothing the mask function will help to reduce oscillations near the wall, but will not lead to any gain in accuracy. Therefore in the present work, we keep using a sharp mask function for all simulations. However, different smoothing strategies are still worth testing to see if an optimal smoothing approach can be found for the present approach.

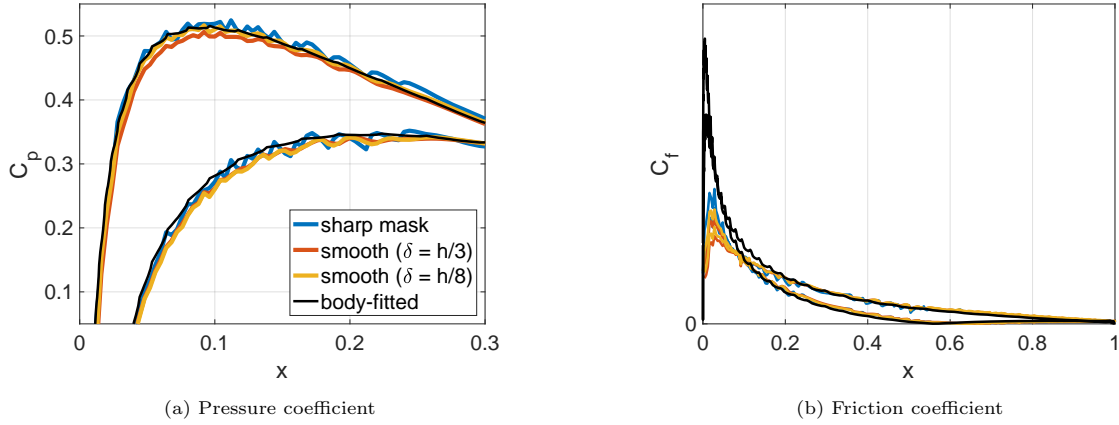


Figure B.22: Comparison of using smooth mask function for flow past a NACA0012 airfoil at $Re = 5000$, $M = 0.5$ and angle of attack 2 degree.

Appendix C. Comparison of data reconstruction methods

As discussed in Sec. 4.4, for the present method, it is straightforward to employ the high-order polynomial interpolation to get the quantities of interest for surface marker points. However, basic analysis on the interpolation stencil shows that such operation will involve the solution points immersed solid, with nonphysical values and gradients. A comparison of data reconstruction methods is made here to show the difference between the polynomial interpolation and the IDW-IP method used in the present study. We take the third result from Sec. 5.2, where flow past a cylinder at Reynolds number 40 is simulated. The third mesh with locally uniform grid size $0.01D$ is selected and the polynomial order is set to 2. We take the final flow field, and reconstruct the surface quantities based on both methods. The comparison of pressure coefficient and wall spanwise vorticity is shown in Fig. C.23. It is clearly seen that the pressure coefficient from high-order polynomial interpolation shows very large oscillation, while results from IDW-IP are smooth and agree well with body-fitted simulation. The under-prediction of wall spanwise vorticity from high-order polynomial interpolation is evident in Fig. C.23b, where the result is not only oscillatory but also does not fit the body-fitted simulation very well. This highlights the weakness of using high-order polynomial to interpolate flow quantities on the surface.

References

- [1] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, et al., High-order cfd methods: current status and perspective, *International Journal for Numerical Methods in Fluids* 72 (2013) 811–45.
- [2] H. Huynh, Z. J. Wang, P. E. Vincent, High-order methods for computational fluid dynamics: A brief review of compact differential formulations on unstructured grids, *Computers & fluids* 98 (2014) 209–20.
- [3] J. S. Hesthaven, T. Warburton, *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*, Springer Science & Business Media, 2007.
- [4] H. T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous galerkin methods, in: *18th AIAA Computational Fluid Dynamics Conference*, 2007, p. 4079.

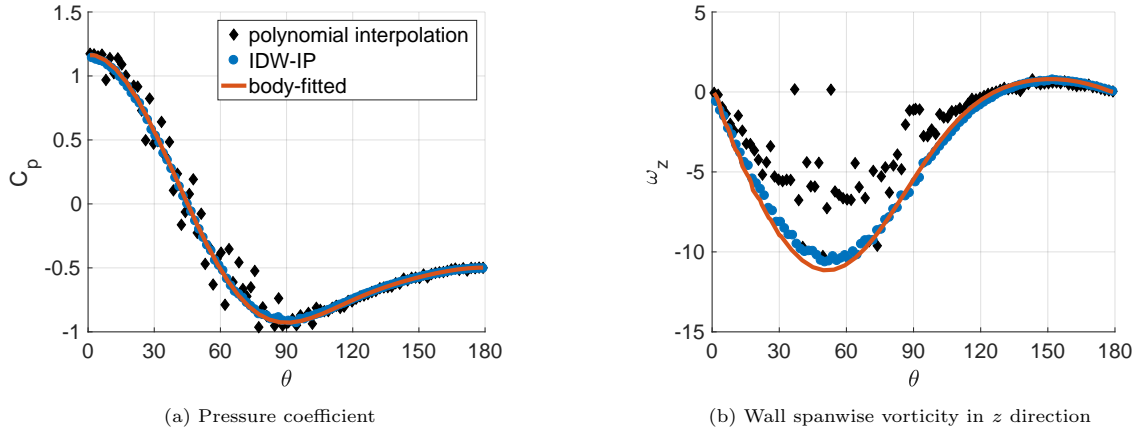


Figure C.23: Comparison of data reconstruction method for flow past a cylinder at $Re = 40$ and $M = 0.2$.

- [5] Y. Liu, M. Vinokur, Z. J. Wang, Spectral difference method for unstructured grids i: basic formulation, *Journal of Computational Physics* 216 (2006) 780–801.
- [6] Z. J. Wang, Y. Liu, G. May, A. Jameson, Spectral difference method for unstructured grids ii: extension to the euler equations, *Journal of Scientific Computing* 32 (2007) 45–71.
- [7] A. J. Lew, G. C. Buscaglia, A discontinuous-galerkin-based immersed boundary method, *International Journal for Numerical Methods in Engineering* 76 (2008) 427–54. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2312>. doi:10.1002/nme.2312. arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2312>.
- [8] A. J. Lew, M. Negri, Optimal convergence of a discontinuous-galerkin-based immersed boundary method, *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique* 45 (2011) 651–74. URL: http://www.numdam.org/item/M2AN_2011__45_4_651_0. doi:10.1051/m2an/2010069.
- [9] R. Rangarajan, A. Lew, G. C. Buscaglia, A discontinuous-galerkin-based immersed boundary method with non-homogeneous boundary conditions and its application to elasticity, *Computer Methods in Applied Mechanics and Engineering* 198 (2009) 1513–34. URL: <http://www.sciencedirect.com/science/article/pii/S0045782509000413>. doi:<https://doi.org/10.1016/j.cma.2009.01.018>.
- [10] P. Bastian, C. Engwer, An unfitted finite element method using discontinuous galerkin, *International journal for numerical methods in engineering* 79 (2009) 1557–76.
- [11] G. Brandstetter, S. Govindjee, A high-order immersed boundary discontinuous-galerkin method for poisson’s equation with discontinuous coefficients and singular sources, *International Journal for Numerical Methods in Engineering* 101 (2015) 847–69.
- [12] H. Dong, B. Wang, Z. Xie, L.-L. Wang, An unfitted hybridizable discontinuous Galerkin method for the Poisson interface problem and its error analysis, *IMA Journal of Numerical Analysis* 37 (2016) 444–76. URL: <https://doi.org/10.1093/imanum/drv071>. doi:10.1093/imanum/drv071. arXiv:<https://academic.oup.com/imajna/article-pdf/37/1/444/9633654/drv071.pdf>.

- [13] H. L. N. THANH, Immersed Hybridizable Discontinuous Galerkin Method for Multi-Viscosity Incompressible Navier-Stokes Flows on Irregular Domains, Ph.D. thesis, National University of Singapore, 2010.
- [14] K. J. Fidkowski, D. L. Darmofal, A triangular cut-cell adaptive method for high-order discretizations of the compressible navier-stokes equations, *Journal of Computational Physics* 225 (2007) 1653–72.
- [15] B. Müller, S. Krämer-Eis, F. Kummer, M. Oberlack, A high-order discontinuous galerkin method for compressible flows with immersed boundaries, *International Journal for Numerical Methods in Engineering* 110 (2017) 3–30.
- [16] M. Geisenhofer, F. Kummer, B. Müller, A discontinuous galerkin immersed boundary solver for compressible flows: Adaptive local time stepping for artificial viscosity-based shock-capturing on cut cells, *International Journal for Numerical Methods in Fluids* 91 (2019) 448–72.
- [17] S. Schoeder, S. Sticko, G. Kreiss, M. Kronbichler, High-order cut discontinuous galerkin methods with local time stepping for acoustics, *International Journal for Numerical Methods in Engineering* 121 (2020) 2979–3003. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6343>. doi:10.1002/nme.6343. arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.6343>.
- [18] P. Brady, D. Livescu, Foundations for high-order, conservative cut-cell methods: Stable discretizations on degenerate meshes, *Journal of Computational Physics* (2020) 109794. URL: <http://www.sciencedirect.com/science/article/pii/S0021999120305684>. doi:<https://doi.org/10.1016/j.jcp.2020.109794>.
- [19] K. Schaal, A. Bauer, P. Chandrashekar, R. Pakmor, C. Klingenberg, V. Springel, Astrophysical hydrodynamics with a high-order discontinuous Galerkin scheme and adaptive mesh refinement, *Monthly Notices of the Royal Astronomical Society* 453 (2015) 4278–300. URL: <https://doi.org/10.1093/mnras/stv1859>. doi:10.1093/mnras/stv1859. arXiv:<https://academic.oup.com/mnras/article-pdf/453/4/4278/8034398/stv1859.pdf>.
- [20] X.-J. Zhang, Y.-S. Zhu, K. Yan, Y.-Y. Zhang, New immersed boundary method on the adaptive cartesian grid applied to the local discontinuous galerkin method, *Chinese Journal of Mechanical Engineering* 31 (2018) 22.
- [21] A. C. Kirby, D. J. Mavriplis, Gpu-accelerated discontinuous galerkin methods: 30x speedup on 345 billion unknowns, 2020. arXiv:2006.15698.
- [22] H. T. Huynh, A reconstruction approach to high-order schemes including discontinuous galerkin for diffusion, in: 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, 2009, p. 403.
- [23] P. E. Vincent, P. Castonguay, A. Jameson, A new class of high-order energy stable flux reconstruction schemes, *Journal of Scientific Computing* 47 (2011) 50–72.
- [24] C. S. Peskin, Flow patterns around heart valves: a numerical method, *Journal of computational physics* 10 (1972) 252–71.

- [25] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, *Appl. Mech. Rev.* 56 (2003) 331–47.
- [26] J. Shao, C. Shu, Y.-T. Chew, Development of an immersed boundary-phase field-lattice boltzmann method for neumann boundary condition to study contact line dynamics, *Journal of Computational Physics* 234 (2013) 8–32.
- [27] W.-X. Huang, S. J. Shin, H. J. Sung, Simulation of flexible filaments in a uniform flow by the immersed boundary method, *Journal of computational physics* 226 (2007) 2206–28.
- [28] L. Wang, F.-B. Tian, J. C. Lai, An immersed boundary method for fluid–structure–acoustics interactions involving large deformations and complex geometries, *Journal of Fluids and Structures* 95 (2020) 102993.
- [29] H. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface cartesian grid method for simulating flows with complex moving boundaries, *Journal of computational physics* 174 (2001) 345–80.
- [30] T. Ye, R. Mittal, H. Udaykumar, W. Shyy, An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries, *Journal of computational physics* 156 (1999) 209–40.
- [31] F. Örley, V. Pasquariello, S. Hickel, N. A. Adams, Cut-element based immersed boundary method for moving geometries in compressible liquid flows with cavitation, *Journal of Computational Physics* 283 (2015) 1–22.
- [32] S. Majumdar, G. Iaccarino, P. Durbin, Rans solvers with adaptive structured boundary non-conforming grids, *Annual Research Briefs* 1 (2001).
- [33] Y.-H. Tseng, J. H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, *Journal of computational physics* 192 (2003) 593–623.
- [34] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, *Journal of computational physics* 105 (1993) 354–66.
- [35] E. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, *Journal of computational physics* 161 (2000) 35–60.
- [36] H. Luo, H. Dai, P. J. F. de Sousa, B. Yin, On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries, *Computers & Fluids* 56 (2012) 61–76.
- [37] P. Angot, C.-H. Bruneau, P. Fabrie, A penalization method to take into account obstacles in incompressible viscous flows, *Numerische Mathematik* 81 (1999) 497–520.
- [38] Z. Li, M.-C. Lai, The immersed interface method for the navier–stokes equations with singular forces, *Journal of Computational Physics* 171 (2001) 822–42.
- [39] D. Z. Huang, D. De Santis, C. Farhat, A family of position-and orientation-independent embedded boundary methods for viscous flow and fluid–structure interaction problems, *Journal of Computational Physics* 365 (2018) 74–104.

- [40] R. Mittal, G. Iaccarino, Immersed boundary methods, *Annu. Rev. Fluid Mech.* 37 (2005) 239–61.
- [41] F. Sotiropoulos, X. Yang, Immersed boundary methods for simulating fluid–structure interaction, *Progress in Aerospace Sciences* 65 (2014) 1–21.
- [42] K. Schneider, Immersed boundary methods for numerical simulation of confined fluid and plasma turbulence in complex geometries: a review, *arXiv preprint arXiv:1508.04593* (2015).
- [43] B. E. Griffith, N. A. Patankar, Immersed methods for fluid–structure interaction, *Annual Review of Fluid Mechanics* 52 (2020) 421–48.
- [44] K. Zhou, S. Balachandar, An analysis of the spatio-temporal resolution of the immersed boundary method with direct forcing, *Journal of Computational Physics* (2020) 109862.
- [45] A. Vreman, Immersed boundary and overset grid methods assessed for stokes flow due to an oscillating sphere, *Journal of Computational Physics* (2020) 109783.
- [46] R. Abgrall, H. Beaugendre, C. Dobrzynski, An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques, *Journal of Computational Physics* 257 (2014) 83–101.
- [47] B. Kadoch, D. Kolomenskiy, P. Angot, K. Schneider, A volume penalization method for incompressible flows and scalar advection–diffusion with moving obstacles, *Journal of Computational Physics* 231 (2012) 4365–83.
- [48] R. Courant, Variational methods for the solution of problems of equilibrium and vibrations, Technical Report, 1943. doi:10.1090/S0002-9904-1943-07818-4.
- [49] E. Arquis, J. Caltagirone, Sur les conditions hydrodynamiques au voisinage d’une interface milieu fluide-milieu poreux: application à la convection naturelle, *CR Acad. Sci. Paris II* 299 (1984) 1–4.
- [50] G. Carbou, P. Fabrie, Boundary layer for a penalization method for viscous incompressible flow, *Advances in Differential equations* 8 (2003) 1453–80.
- [51] I. Ramière, P. Angot, M. Belliard, A general fictitious domain method with immersed jumps and multilevel nested structured meshes, *Journal of Computational Physics* 225 (2007) 1347–87.
- [52] T. Sakurai, K. Yoshimatsu, N. Okamoto, K. Schneider, Volume penalization for inhomogeneous neumann boundary conditions modeling scalar flux in complicated geometry, *Journal of Computational Physics* 390 (2019) 452–69.
- [53] Q. Liu, O. V. Vasilyev, A brinkman penalization method for compressible flows in complex geometries, *Journal of Computational Physics* 227 (2007) 946–66.
- [54] E. Brown-Dymkoski, N. Kasimov, O. V. Vasilyev, A characteristic based volume penalization method for general evolution problems applied to compressible viscous flows, *Journal of Computational Physics* 262 (2014) 344–57.
- [55] D. Kolomenskiy, K. Schneider, A fourier spectral method for the navier–stokes equations with volume penalization for moving solid obstacles, *Journal of Computational Physics* 228 (2009) 5687–709.

- [56] P. Horgue, M. Prat, M. Quintard, A penalization technique applied to the “volume-of-fluid” method: Wettability condition on immersed boundaries, *Computers & Fluids* 100 (2014) 255–66.
- [57] T. Engels, D. Kolomenskiy, K. Schneider, J. Sesterhenn, Numerical simulation of fluid–structure interaction with the volume penalization method, *Journal of Computational Physics* 281 (2015) 96–115.
- [58] X. Cui, X. Yao, Z. Wang, M. Liu, A coupled volume penalization-thermal lattice boltzmann method for thermal flows, *International Journal of Heat and Mass Transfer* 127 (2018) 253–66.
- [59] M. Specklin, Y. Delauré, A sharp immersed boundary method based on penalization and its application to moving boundaries and turbulent rotating flows, *European Journal of Mechanics-B/Fluids* 70 (2018) 130–47.
- [60] A. Piquet, O. Roussel, A. Hadjadj, A comparative study of brinkman penalization and direct-forcing immersed boundary methods for compressible viscous flows, *Computers & Fluids* 136 (2016) 272–84.
- [61] H. Viviand, Conservative forms of gas dynamic equations, *La Recherche Aérospatiale* 1974 (1974) 65–8.
- [62] M. Vinokur, Conservation equations of gasdynamics in curvilinear coordinate systems, *Journal of Computational Physics* 14 (1974) 105–25.
- [63] D. M. Williams, P. Castonguay, P. E. Vincent, A. Jameson, Energy stable flux reconstruction schemes for advection–diffusion problems on triangles, *Journal of Computational Physics* 250 (2013) 53–76.
- [64] G. Mengaldo, D. De Grazia, F. Witherden, A. Farrington, P. Vincent, S. Sherwin, J. Peiro, A guide to the implementation of boundary conditions in compact high-order methods for compressible aerodynamics, in: 7th AIAA Theoretical Fluid Mechanics Conference, 2014, p. 2923.
- [65] G. Strang, On the construction and comparison of difference schemes, *SIAM journal on numerical analysis* 5 (1968) 506–17.
- [66] R. Mittal, H. Dong, M. Bozkurtas, F. Najjar, A. Vargas, A. Von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, *Journal of computational physics* 227 (2008) 4825–52.
- [67] K. Hormann, A. Agathos, The point in polygon problem for arbitrary polygons, *Computational geometry* 20 (2001) 131–44.
- [68] J. Yang, F. Stern, A non-iterative direct forcing immersed boundary method for strongly-coupled fluid–solid interactions, *Journal of Computational Physics* 295 (2015) 779–804.
- [69] A. Bharadwaj S, S. Ghosh, Data reconstruction at surface in immersed-boundary methods, *Computers & Fluids* 196 (2020) 104236.
- [70] J.-I. Choi, R. C. Oberoi, J. R. Edwards, J. A. Rosati, An immersed boundary method for complex incompressible flows, *Journal of Computational Physics* 224 (2007) 757–84.
- [71] D. Kolomenskiy, K. Schneider, et al., Analysis and discretization of the volume penalized laplace operator with neumann boundary conditions, *Applied Numerical Mathematics* 95 (2015) 238–49.

- [72] C. Jause Labert, Simulation numerique d'écoulements turbulents en rotation, confinement et forçage a l'aide d'une methode de penalisation, Ph.D. thesis, Ecully, Ecole centrale de Lyon, 2012.
- [73] J. S. Hesthaven, D. Gottlieb, Stable spectral methods for conservation laws on triangles with unstructured grids, *Computer methods in applied mechanics and engineering* 175 (1999) 361–81.
- [74] D. Kolomenskiy, K. Schneider, et al., Approximation of the laplace and stokes operators with dirichlet boundary conditions through volume penalization: a spectral viewpoint, *Numerische Mathematik* 128 (2014) 301–38.
- [75] F. De Vanna, F. Picano, E. Benini, A sharp-interface immersed boundary method for moving objects in compressible viscous flows, *Computers & Fluids* (2020) 104415.
- [76] S. Dennis, G.-Z. Chang, Numerical solutions for steady flow past a circular cylinder at reynolds numbers up to 100, *Journal of Fluid Mechanics* 42 (1970) 471–89.
- [77] B. Fornberg, A numerical study of steady viscous flow past a circular cylinder, *Journal of Fluid Mechanics* 98 (1980) 819–55.
- [78] D. A. Kopriva, A conservative staggered-grid chebyshev multidomain method for compressible flows. ii. a semi-structured method, *Journal of computational physics* 128 (1996) 475–88.
- [79] M. Kompenhans, G. Rubio, E. Ferrer, E. Valero, Comparisons of p-adaptation strategies based on truncation-and discretisation-errors for high order discontinuous galerkin methods, *Computers & Fluids* 139 (2016) 36–46.
- [80] A. M. Rueda-Ramírez, J. Manzanero, E. Ferrer, G. Rubio, E. Valero, A p-multigrid strategy with anisotropic p-adaptation based on truncation errors for high-order discontinuous galerkin methods, *Journal of Computational Physics* 378 (2019) 209–33.
- [81] D. S. Dandy, H. A. Dwyer, A sphere in shear flow at finite reynolds number: effect of shear on particle lift, drag, and heat transfer, *Journal of Fluid Mechanics* 216 (1990) 381–410.
- [82] T. Johnson, V. Patel, Flow past a sphere up to a reynolds number of 300, *Journal of Fluid Mechanics* 378 (1999) 19–70.
- [83] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, *Journal of computational physics* 171 (2001) 132–50.
- [84] M. F. Platzer, K. D. Jones, J. Young, J. C. Lai, Flapping wing aerodynamics: progress and challenges, *AIAA journal* 46 (2008) 2136–49.
- [85] C. Gao, W. Zhang, X. Li, Y. Liu, J. Quan, Z. Ye, Y. Jiang, Mechanism of frequency lock-in in transonic buffeting flow, *Journal of Fluid Mechanics* 818 (2017) 528.
- [86] P.-O. Persson, C. Fidkowski, Test case cl1—heaving and pitching airfoil, in: *5th International Workshop on High-Order CFD Methods*, 2018.