14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11-–15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

HIGH-ORDER FLUX RECONSTRUCTION BASED ON IMMERSED BOUNDARY METHOD

Jiaqing Kou[1,2,*], Saumitra Joshi[1,2], Aurelio Hurtado-de-Mendoza[1,2], Kunal Puri[1], Charles Hirsch[1] and Esteban Ferrer[2]

[1] NUMECA International S.A., Chaussee de la Hulpe 187, Brussels, B-1170, Belgium

[2] ETSIAE-UPM-School of Aeronautics, Universidad Politécnica de Madrid, Plaza Cardenal Cisneros 3, Madrid E-28040 Spain

[*] Email address: jiaqing.kou@numeca.be

Key words: Flux Reconstruction, Immersed Boundary Method, High-Order Numerical Method, Volume Penalization

Abstract. In the last decade, high-order methods for Computational Fluid Dynamics (CFD) are becoming attractive for unsteady scale-resolving-simulations in industrial CFD applications, due to their advantages of low numerical dissipation, high efficiency on modern architectures and quasi mesh-independence. However, the generation of body-fitted mesh for high-order methods is still a significant bottleneck and often determines the overall quality of the solution. To avoid the complexity of mesh generation, the present work combines the numerical advantages of the high-order Flux Reconstruction (FR) method and the simplicity of the mesh generation based on Immersed Boundary Method (IBM) that allows solving flow past obstacles on a non body-fitted mesh. The volume penalization method is selected for its ease of implementation and robustness. The proposed method is validated by several test cases, including flow past a cylinder and NACA0012 airfoil for static and moving boundaries. Good agreement with body-fitted simulation is reported.

1 INTRODUCTION

It is well recognized within the CFD community that high-order numerical methods have better accuracy than low-order schemes with comparable computational cost. Additionally, the paradigm of high-order methods has the advantage that it introduces the polynomial order as a new flexibility, which leads to the possibility of improving the accuracy locally in regions of interest. Therefore, high-order numerical methods, including discontinuous Galerkin (DG) [9], flux reconstruction (FR) [10] and spectral difference (SD) [13] [25], have attracted lots of attention in these years. However, there are still some issues that limit the application of high-order method for real engineering problems. One of them is the exploration of high-order method for moving boundary and complex geometry, where complicated grid generation techniques are required. A viable solution to this problem is to extend the immersed boundary method (IBM) in the framework of high-order methods.

IBM is introduced by Peskin in the 1970s [17]. Since its introduction, IBM has been applied to different problems, in order to solve the flow around solid body on a simple, non-conforming grid. IBM mimics

the existence of solid boundary through proper numerical treatment to the background flow grid. It can handle complex geometries with large structure deformations or moving boundaries very efficiently. So far, there have been a lot of different IBM methods developed, as reviewed by Mittal and Iaccarino [16], Sotiropoulos and Yang [22], as well as Griffith and Neelesh [8]. Generally, IBM can be classified into two groups: 1) cutting the elements by the boundary, and solve the equation in the same way as those for body-fitted mesh; 2) adding source terms to the governing equations. The first group, usually known as the 'cut-cell' approach, guarantees the strict global and local conservation of mass and momentum [16]. The main challenge for the cut-cell method is the cutting approach for three-dimensional flows and complex geometry, since it can result in complex or very small cut-cells that require efficient discretization schemes to deal with.

In the second group of IBM approach, the background mesh remains the same for both static and moving obstacles, but the IBM forcing terms are added to related grid nodes (or solution points in high-order method) to reflect the effect of boundary. The governing equations are discretized and solved as usual with the new IBM source term. Most of IBM approaches can be classified into this group, including conventional IBM, ghost cell, direct forcing and volume penalization method. Among all these methods, volume penalization has shown advantages in terms of robustness, simplicity and rigorous theoretical foundation. Compared with other IBM methods, the extension to moving boundary problems is straightforward. The basic idea of volume penalization method is to model complex solid bodies as porous media with a permeability approaching zero, i.e., $\eta \to 0$, through penalizing the velocity (or diffusivity) of grid points immersed in the solid.

The volume penalization method for the Navier-Stokes equations was first proposed by Arquis and Caltagirone [3] to simulate the natural convection flow inside a fluid–porous cavity. Rigorous proofs of the convergence property is then given by Angot et al. [2] and Carbou and Fabrie [6], where imposing Dirichlet boundary conditions to the Navier–Stokes equations was studied. It was proved that, as the penalization parameter η approaches 0, the solution of the penalized Navier-Stokes equations will converge to the solution of the same equations with no-slip boundary conditions. For general boundary conditions refer to [19] [11] and [20]. Applications of compressible flows have been shown in [12] [5] [1]. A review of volume penalization method for numerical simulation of complex flows is given by Schneider [21].

The high-order method adopted in the current study is based on flux reconstruction (FR) approach proposed by Huynh [10]. It provides a differential framework for discontinuous finite element schemes, which unifies several other high-order methods [24]. Combined with IBM implementation, the proposed method will handle complex geometries with large structure deformation or moving bodies very efficiently, and the near-wall resolution can be improved through local polynomial refinement. In the following sections, the methodology and test cases will be introduced, and finally conclusions will be drawn.

2 METHODOLOGY

2.1 Flux Reconstruction

The governing equations for a compressible viscous fluid based on Navier-Stokes equations are focused and solved, written as

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = \frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} + \frac{\partial \boldsymbol{F}_z}{\partial z} = 0\,, \tag{1}$$

where $\boldsymbol{U}$ denotes the vector of conserved variables $\boldsymbol{U} = (\rho, \rho u, \rho v, \rho w, E)^T$. ρ is the density, u, v, and w are the velocity components and E is the total energy. The flux vectors are $\boldsymbol{F}_x$, $\boldsymbol{F}_y$, $\boldsymbol{F}_z$ containing the inviscid and viscous fluxes.

To achieve FR, the domain is firstly discretized by non overlapping elements. For each element, N_p internal degrees of freedom and N_f points on the element interface are defined. A multi-dimensional Lagrange polynomial or a modal expansion in a chosen polynomial basis is chosen to describe the solution based on the internal degree of freedom. Here the solution is represented in the nodal basis as follows:

$$\boldsymbol{U}^{\delta \mathrm{D}}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \boldsymbol{U}_i^{\delta \mathrm{D}} I_i^P(\boldsymbol{\xi}) \tag{2}$$

where $\boldsymbol{\xi}$ is the coordinate in the reference space. δD indicates that the solution is discrete (δ) and discontinuous within the element. I_i^P refers to the nodal basis function defined at each solution point with polynomials of degree P, and $\boldsymbol{U}_i^{\delta \mathrm{D}}$ is the solution at the ith solution point. The discontinuous flux $\boldsymbol{F}^{\delta \mathrm{D}}$ is also expressed in terms of the basis polynomial. The key component in FR is the correction function which leverages the discontinuous values to be continuous values. The resulting continuous solution fluxes are represented as:

$$\boldsymbol{U}^{\delta \mathrm{C}}(\boldsymbol{\xi}) = \boldsymbol{U}^{\delta \mathrm{D}}(\boldsymbol{\xi}) + (\boldsymbol{U}^{Comm}(\boldsymbol{\xi}_{flux}) - \boldsymbol{L}_f \boldsymbol{U}^{\delta \mathrm{D}})\boldsymbol{h}(\boldsymbol{\xi}) \tag{3}$$

$$\boldsymbol{F}^{\delta \mathrm{C}}(\boldsymbol{\xi}) = \boldsymbol{F}^{\delta \mathrm{D}}(\boldsymbol{\xi}) + (\boldsymbol{F}^{Int}(\boldsymbol{\xi}_{flux}) - \boldsymbol{L}_f \boldsymbol{F}^{\delta \mathrm{D}}) \cdot \vec{n}\boldsymbol{h}(\boldsymbol{\xi}) \tag{4}$$

where $\boldsymbol{U}^{Comm}$ and $\boldsymbol{F}^{Int}$ denote the common solution value and the numerical interaction flux at the flux point, shared by elements of both sides. $\boldsymbol{L}_f$ is the interpolation operator to obtain values at the flux point. $\boldsymbol{h}$ serves as a "lifting" operator [26] that will transfer the correction values to the quadrature points within the element. Consequently, the second term in these equations are defined as the correction terms (with superscript Corr). For a given interface, the common solution and interaction flux defined on the flux point, shared by both left and right elements, are obtained by the Local Discontinuous Galerkin (LDG) formulation for the viscous terms and the Rusanov flux for the inviscid terms. The gradient and divergence operators for the scheme can be expressed as [26]:

$$\tilde{\nabla}\boldsymbol{U} = \sum_{i=1}^{N_p} \boldsymbol{U}_i^{\delta \mathrm{D}} \tilde{\nabla} I_i^P(\boldsymbol{\xi}) + \sum_{f=1}^{N_{face}} \sum_{j=1}^{N_f} \tilde{\nabla} C_{f,j}^{P+1}(\boldsymbol{\xi}) \cdot \boldsymbol{U}_{f,j}^{\delta \mathrm{Corr}} \tag{5}$$

$$\tilde{\nabla} \cdot \boldsymbol{F} = \sum_{k=1}^{\mathcal{D}} \sum_{i=1}^{N_p} \tilde{\nabla}_k I_i^P(\boldsymbol{\xi}) \cdot \boldsymbol{F}_{i,k}^{\delta \mathrm{D}} + \sum_{k=1}^{\mathcal{D}} \sum_{f=1}^{N_{face}} \sum_{j=1}^{N_f} \tilde{\nabla}_k C_{f,j}^{P+1}(\boldsymbol{\xi}) \cdot \boldsymbol{F}_{f,j,k}^{\delta \mathrm{Corr}} \tag{6}$$

where $\mathcal{D}$ refers to the space dimension, N_{face} refers to number of faces for each element. $\tilde{\nabla}$ is the discrete gradient in the reference space. The function $C_{f,j}^{P+1}$ is the correction function, which is of polynomial order $P+1$. Note that the resulting gradient and flux should be transformed to the physical space. With the divergence of flux, the solution can be solved with efficient time-integration schemes. Here we use explicit time integration using the the third-order TVD Runge-Kutta scheme. Since the IBM approach is

explored, the only boundary condition considered for the proposed method is the characteristic condition for the far field.

2.2 Volume Penalization

Here the volume penalization method is introduced. The effect of solid bodies is imposed by introducing forcing terms to the governing equations, while the resulting equations are discretized and solved as usual. A mask function which distinguishes between the fluid region Ω_f and solid region Ω_s is firstly defined:

$$\chi(\boldsymbol{x},t) = \begin{cases} 1, \text{ if } \boldsymbol{x} \in \Omega_s \\ 0, \text{ otherwise} \end{cases} \tag{7}$$

This mask function is used to determine whether the IBM force should be imposed [1] [5]. The Navier-Stokes equations with IBM can be written as follows:

$$\frac{\partial \boldsymbol{U}}{\partial t} = \mathbf{RHS} + \chi \boldsymbol{S}(\boldsymbol{U}) \tag{8}$$

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

$$\mathbf{RHS} = -(\frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{F}_y}{\partial y} + \frac{\partial \boldsymbol{F}_z}{\partial z}). \tag{9}$$

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:

$$\boldsymbol{S}(\boldsymbol{U}) = \frac{1}{\eta} \times \begin{pmatrix} 0 \\ \rho u_s - \rho u \\ \rho v_s - \rho v \\ \rho w_s - \rho w \\ \frac{\rho}{2}(u_s^2 + v_s^2 + w_s^2) - \frac{\rho}{2}(u^2 + v^2 + w^2) \end{pmatrix} \tag{10}$$

where η denotes the penalization parameter for IBM. A velocities with subscript s are known velocities to be imposed. The penalization terms proposed were used in [1] for compressible Navier-Stokes equations. Generally, the penalization parameter η should be sufficiently small to ensure accuracy, but not too small to avoid stiffness issues of the IBM source term. In practice, the explicit time step Δt is suggested to be the penalization parameter [7]. In particular, $\mathbf{u}_s = (0,0,0)^T$ will be imposed for no-slip boundary condition.

The governing equation Eq.8 is then marched in time by the time integration method. Since a source term is introduced, the original equations can be separated into two parts, i.e., right hand side term and the source term. Based on operator splitting approach, each part can be solved separately and then combined together to form the solution. Therefore, second-order Strang splitting [23] is used to separate

the equation into two parts. At time step n, the solution is separated into three steps:

$$step\,1: \frac{\boldsymbol{U}_1 - \boldsymbol{U}^n}{\Delta t_1} = \boldsymbol{S}(\boldsymbol{U}_1), \Delta t_1 = \Delta t/2, \boldsymbol{U}_{1,0} = \boldsymbol{U}_n \tag{11}$$

$$step\,2: \frac{\boldsymbol{U}_2 - \boldsymbol{U}_1}{\Delta t_2} = \mathbf{RHS}(\boldsymbol{U}_2), \Delta t_2 = \Delta t, \boldsymbol{U}_{2,0} = \boldsymbol{U}_1 \tag{12}$$

$$step\,3: \frac{\boldsymbol{U}^{n+1} - \boldsymbol{U}_2}{\Delta t_3} = \boldsymbol{S}(\boldsymbol{U}^{n+1}), \Delta t_3 = \Delta t/2, \boldsymbol{U}_0^{n+1} = \boldsymbol{U}_2 \tag{13}$$

Both parts can be solved in explicit or implicit time integration method. Currently, in step 2, we use the third-order TVD Runge-Kutta method to perform explicit time marching. For the first and the third step, two approaches, including explicit and implicit forcing, can be chosen, where the implicit forcing will better handle the stiff IBM source terms for small penalization parameter.

Explicit forcing : The explicit formulation is simply given as:

$$\frac{\boldsymbol{U}_1 - \boldsymbol{U}^n}{\Delta t_1} = \boldsymbol{S}(\boldsymbol{U}^n), \tag{14}$$

$$\boldsymbol{U}_1 = \boldsymbol{U}^n + \Delta t_1 \cdot \boldsymbol{S}(\boldsymbol{U}^n). \tag{15}$$

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

$$\frac{\boldsymbol{U}_1 - \boldsymbol{U}^n}{\Delta t_1} = \boldsymbol{S}(\boldsymbol{U}^n) + \frac{\partial \boldsymbol{S}(\boldsymbol{U}^n)}{\partial \boldsymbol{U}}(\boldsymbol{U}_1 - \boldsymbol{U}^n) \tag{16}$$

2.3 Implementation Details for IBM

From an implementation point of view, two additional aspects should be considered: 1) Boundary representation and mask function, 2) Surface data reconstruction. Both will be discussed in this subsection.

An appropriate boundary representation is needed for general geometries where analytical shape functions cannot be found, in order to obtain the mask function χ, and to compute aerodynamic coefficients from solution points near the surface. In this work, 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 [15, 14].

With this representation, a method to get mask function for general shapes is then developed. This method is based on the ray casting method for the 'point in polygon' (PIP) problem. It generates a ray starting from the point and going in any fixed direction. If the point is inside / outside the polygon, the ray will intersect the edges an odd / even number of times. This is implemented based on the nearest neighbor search algorithm. It firstly generate a bounding box that are formed by coordinates of the rectangular border that fully encloses the solid body. If the point is in the bounding box, then it identifies the number of intercept points in a particular physical direction, in order to determine whether the current solution point is inside or outside the boundary.

To get the surface quantity for each surface point (i.e., IB point), an interpolation from data of surrounding solution points is needed. Here, the interpolation method described in [4] is used. This method is based on the inverse distance between the IB point and the interpolation point selected from several nearest solution points. In particular, the Inverse Distance Weight at Interpolation Point (IDW-IP) method [4] is used for interpolation in the present study.

3 TEST CASES

The proposed method is tested by different cases. Here we will show three groups of test cases, including flow past a cylinder at Reynolds number 40, flow past an airfoil at Reynolds number 5000, and flow past a static or moving cylinder at Reynolds number 100.

3.1 Flow past a cylinder at Reynolds number 40

As the first example, we consider the benchmark case of flow past a static cylinder at Reynolds number 40. To test the basic implementation, a body conforming mesh is generated. The body-fitted quad mesh for the cylinder is first generated. In addition, three layers of grid inside the cylinder surface is added, which contain the cells that need to be penalized by volume penalization. The computational grid is shown in Figure 1, with 3480 quad elements. The simulation is performed with $P = 2$, and two penalization parameters, i.e., $1e-4$ and $1e-6$, are considered. The results on surface quantities, including pressure, density and momentum, are compared in Figure 2. As shown in the figure, good agreement for pressure and density is seen, while some error exists for the surface momentum (velocity). As penalization parameter becomes smaller, a slightly better result can be obtained. However, to further improve the result, one has to locally refine the resolution near the wall, by either increasing the polynomial orders or refining the mesh locally.

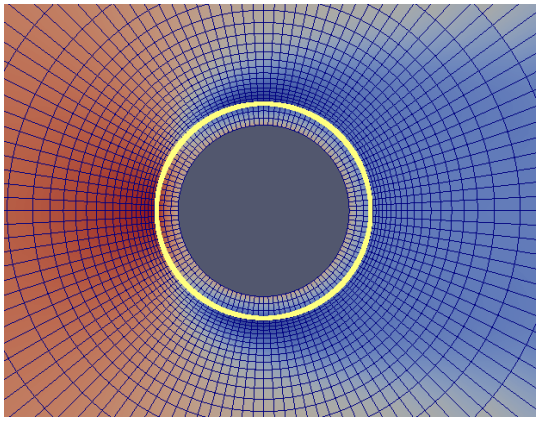


Figure 1: Computational mesh for flow past a cylinder at Reynolds number 40. The mesh is body conforming where the surface aligns with the cell interface. Yellow line is the actual boundary and the elements inside the yellow line are penalized to satisfy the boundary condition.

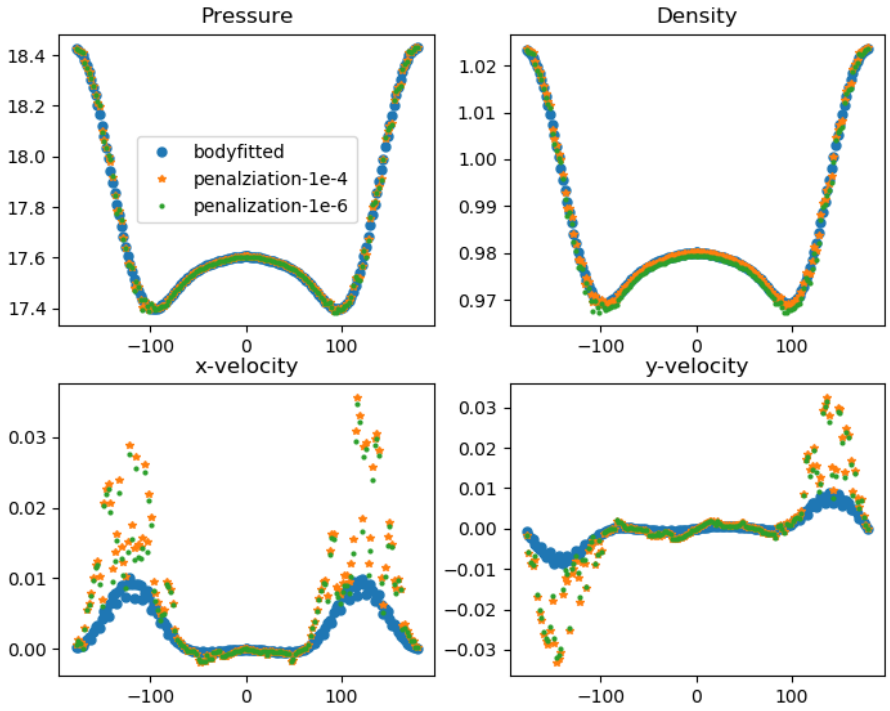
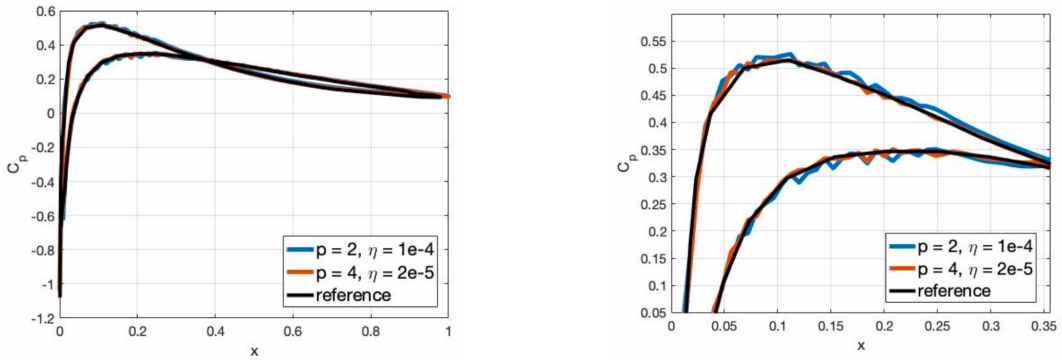


Figure 2: Comparison of IBM simulation for two penalization parameters and the reference case from body-fitted simulation.

3.2 Flow past a NACA0012 airfoil at Reynolds number 5000

The flow over an airfoil is the second test case considered for the present method. Simulation of NACA0012 airfoil at Mach number 0.5 and Reynolds number 5000 is chosen, with an angle of attack 2 degree. The background mesh is a rectangular computation domain defined as $x \in [-30c, 50c]$ 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 total number of 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.

We performed the numerical simulation under two sets of parameters. The first case is simulated with polynomial order $P = 2$ and a constant time step $1e-4$, while the second case is simulated with polynomial order $P = 4$ and time step $2e-5$. The penalization parameter η is chosen to be equal to the time step. Comparison of surface distribution in pressure and friction coefficients are shown in 3 and 4, respectively. As shown in the Figure 3, both simulations give very good prediction on C_p, while when $P = 4$ better accuracy can be seen in Figure 3b. The prediction of surface skin friction coefficient is more difficult, since it involves the reconstruction of gradient. From Figure 4, it is also noticed that as the resolution near the wall increases, C_f is better predicted. As mentioned before, the accuracy can be



(a) Overall view.

(b) Zoomed-in view.

Figure 3: Comparison of surface pressure coefficient distribution with different polynomial orders.

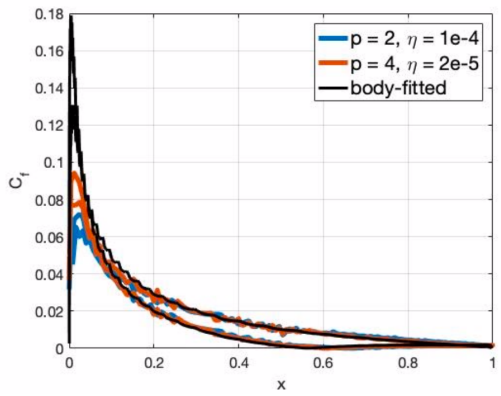


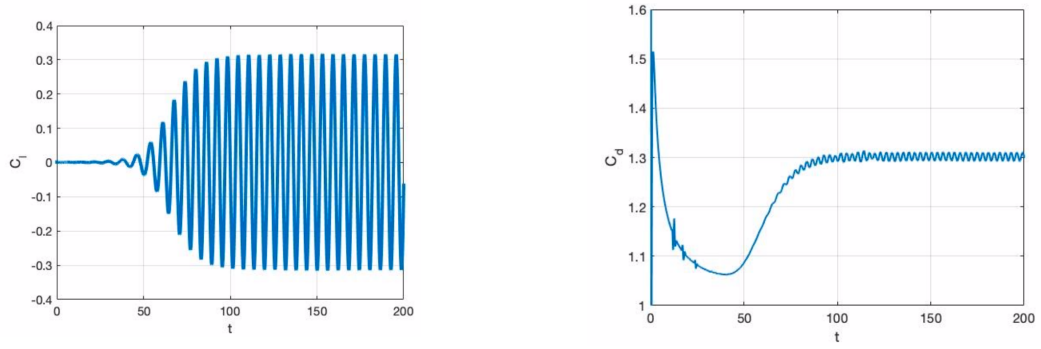
Figure 4: Comparison of surface friction coefficient distribution with different polynomial orders.

further improved by locally refining the resolution near the wall.

3.3 Flow past a static or moving cylinder at Reynolds number 100

As the last test case, unsteady simulation for flow past a cylinder at Reynolds number 100 is focused. The background mesh is a rectangular computation domain defined as $x \in [-30D, 50D]$ and $y \in [-30D, 30D]$, where D is the diameter of the cylinder. In the square region $x \in [-D, D]$ and $y \in [-D, D]$, uniform grid with mesh size $0.015D$ is used. The total number of elements are 377×667. Note that wake region is relatively refined compared with the airfoil case, in order to capture the wake flow structure for periodic vortex shedding. This case is simulated with polynomial order $P = 1$ and a constant time step $2e-4$, where the penalization parameter is also set to $2e-4$.

Firstly, flow past a static cylinder is simulated. The temporal evolution of lift and drag coefficients is shown in Figure 5. From the results, it gives three main quantities of interest: mean drag coefficient 1.30, root mean square of lift coefficient 0.315, vortex shedding frequency $f_s = 0.1667$ (defined by Strouhal number). Compared with body-fitted simulation [18], very good agreement is obtained. In addition, a



(a) Lift coefficient.

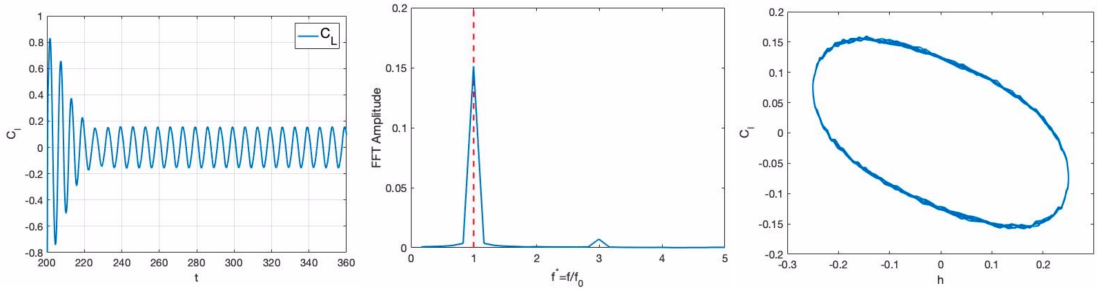
(b) Drag coefficient.

Figure 5: Temporal evolution of aerodynamic coefficients for flow past a cylinder at Reynolds number 100.

moving cylinder is considered, where the mask function becomes time-dependent and the nonzero velocities for solid solution points are imposed according to motion functions. Plunging plunging motion in the transverse direction is simulated, with a sinusoidal motion with amplitude $A = 0.25D$. The frequency of harmonic motion is defined as the ratio between forced motion frequency f_0 and the vortex shedding frequency f_s. Results at frequency ratios 0.9 and 1.5 are shown in Figure 6 and Figure 7, respectively. These two cases are representative since the first case exhibits lock-in phenomenon while the second case does not. When lock-in occurs, the vortex shedding frequency (obtained by the frequency of lift coefficient, f) diverges from the value of static cylinder (f_s) at the same Reynolds number, and it locks on the frequency of the forced oscillation ($f/f_0 \approx 1$). To look at the lock-in phenomenon in the lift force, Fast Fourier Transformation (FFT) is applied to the response of lift coefficient, and the spectrum of FFT amplitude versus the frequency ratio f/f_0 is shown. From FFT spectrum, it is obvious that the most important frequency for both cases is the forced motion frequency. However, for the first case, there is only one dominant frequency $f/f_0 = 1$ while the second case has two, $f/f_0 = 1$ and $f/f_0 = 0.67$ (corresponding to f_s for static cylinder). This validates that the first is the lock-in case since the vortex shedding is the same as forced motion frequency, and the signal is purely periodic. However, in the second case, a beating behavior where the signal is not periodic over two successive cycles of oscillation but over several ones. These phenomena are well captured by the present method. The hysteresis loops of the aerodynamic response are also in good agreement with the reference data [18].

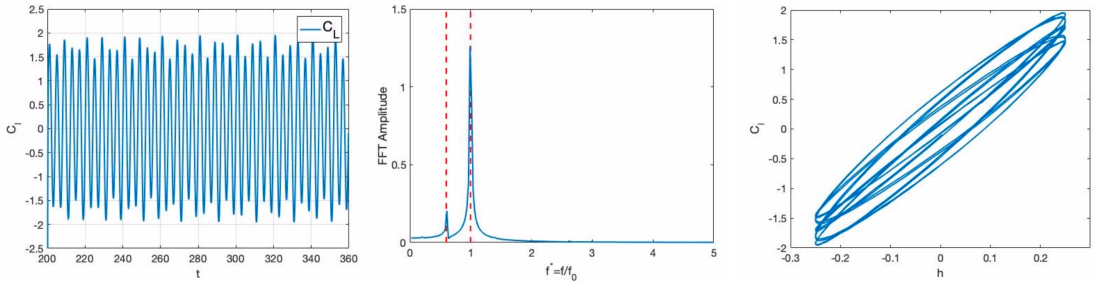
4 CONCLUSIONS

Developing high-order CFD methods for body-fitted mesh has been an active research area in these years. To overcome the complexity involved in mesh generation, it is also important to test the performance of high-order methods with non body-fitted mesh, based on immersed boundary method. In this paper, an immersed boundary method based on volume penalization for high-order flux reconstruction scheme is proposed. Numerical treatment of the stiff source term, based on the splitting method, is also discussed. Through some benchmark test cases, reasonable performance of the proposed method is shown, indicating possible engineering applications of high-order methods to simulate complex geometries with moving boundary.



(a) Lift coefficient. (b) FFT of lift coefficient. (c) Hysteresis loop.

Figure 6: Plunging motion in transverse direction (frequency ratio = 0.9, $A = 0.25D$).



(a) Lift coefficient. (b) FFT of lift coefficient. (c) Hysteresis loop.

Figure 7: Plunging motion in transverse direction (frequency ratio = 1.5, $A = 0.25D$).

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.

REFERENCES

[1] Remi Abgrall, Héloïse Beaugendre, and Cécile Dobrzynski. An immersed boundary method using unstructured anisotropic mesh adaptation combined with level-sets and penalization techniques. *Journal of Computational Physics*, 257:83–101, 2014.

[2] Philippe Angot, Charles-Henri Bruneau, and Pierre Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81(4):497–520, 1999.

[3] E Arquis and JP Caltagirone. Sur les conditions hydrodynamiques au voisinage d'une interface milieu fluide-milieu poreux: applicationa la convection naturelle. *CR Acad. Sci. Paris II*, 299:1–4, 1984.

[4] Anand Bharadwaj S and Santanu Ghosh. Data reconstruction at surface in immersed-boundary methods. *Computers & Fluids*, 196:104236, 2020.

[5] Eric Brown-Dymkoski, Nurlybek Kasimov, and Oleg V Vasilyev. A characteristic based volume

penalization method for general evolution problems applied to compressible viscous flows. *Journal of Computational Physics*, 262:344–357, 2014.

[6] Gilles Carbou and Pierre Fabrie. Boundary layer for a penalization method for viscous incompressible flow. *Advances in Differential equations*, 8(12):1453–1480, 2003.

[7] Thomas Engels, Dmitry Kolomenskiy, Kai Schneider, and Jörn Sesterhenn. Numerical simulation of fluid–structure interaction with the volume penalization method. *Journal of Computational Physics*, 281:96–115, 2015.

[8] Boyce E Griffith and Neelesh A Patankar. Immersed methods for fluid–structure interaction. *Annual Review of Fluid Mechanics*, 52:421–448, 2020.

[9] Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[10] Hung T Huynh. A flux reconstruction approach to high-order schemes including discontinuous galerkin methods. In *18th AIAA Computational Fluid Dynamics Conference*, page 4079, 2007.

[11] Benjamin Kadoch, Dmitry Kolomenskiy, Philippe Angot, and Kai Schneider. A volume penalization method for incompressible flows and scalar advection–diffusion with moving obstacles. *Journal of Computational Physics*, 231(12):4365–4383, 2012.

[12] Qianlong Liu and Oleg V Vasilyev. A brinkman penalization method for compressible flows in complex geometries. *Journal of Computational Physics*, 227(2):946–966, 2007.

[13] Yen Liu, Marcel Vinokur, and Zhi Jian Wang. Spectral difference method for unstructured grids i: basic formulation. *Journal of Computational Physics*, 216(2):780–801, 2006.

[14] Haoxiang Luo, Hu Dai, Paulo JSA Ferreira de Sousa, and Bo Yin. On the numerical oscillation of the direct-forcing immersed-boundary method for moving boundaries. *Computers & Fluids*, 56:61–76, 2012.

[15] Rajat Mittal, Haibo Dong, Meliha Bozkurttas, FM Najjar, Abel Vargas, and Alfred Von Loebbecke. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of computational physics*, 227(10):4825–4852, 2008.

[16] Rajat Mittal and Gianluca Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[17] Charles S Peskin. Flow patterns around heart valves: a numerical method. *Journal of computational physics*, 10(2):252–271, 1972.

[18] Antoine Placzek, Jean-François Sigrist, and Aziz Hamdouni. Numerical simulation of an oscillating cylinder in a cross-flow at low reynolds number: Forced and free oscillations. *Computers & Fluids*, 38(1):80–100, 2009.

[19] Isabelle Ramière, Philippe Angot, and Michel Belliard. A general fictitious domain method with immersed jumps and multilevel nested structured meshes. *Journal of Computational Physics*, 225(2):1347–1387, 2007.

[20] Teluo Sakurai, Katsunori Yoshimatsu, Naoya Okamoto, and Kai Schneider. Volume penalization for inhomogeneous neumann boundary conditions modeling scalar flux in complicated geometry.

Journal of Computational Physics, 390:452–469, 2019.

[21] Kai Schneider. Immersed boundary methods for numerical simulation of confined fluid and plasma turbulence in complex geometries: a review. *arXiv preprint arXiv:1508.04593*, 2015.

[22] Fotis Sotiropoulos and Xiaolei Yang. Immersed boundary methods for simulating fluid–structure interaction. *Progress in Aerospace Sciences*, 65:1–21, 2014.

[23] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM journal on numerical analysis*, 5(3):506–517, 1968.

[24] Peter E Vincent, Patrice Castonguay, and Antony Jameson. A new class of high-order energy stable flux reconstruction schemes. *Journal of Scientific Computing*, 47(1):50–72, 2011.

[25] Zhi Jian Wang, Yen Liu, Georg May, and Antony Jameson. Spectral difference method for unstructured grids ii: extension to the euler equations. *Journal of Scientific Computing*, 32(1):45–71, 2007.

[26] David M Williams, Patrice Castonguay, Peter E Vincent, and Antony Jameson. Energy stable flux reconstruction schemes for advection–diffusion problems on triangles. *Journal of Computational Physics*, 250:53–76, 2013.