A three-dimensional immersed boundary method based on an algebraic forcing-point-searching scheme for water impact problems

Bin Yan[a], Wei Bai[b,*], Sheng-Chao Jiang[c], Peiwen Cong[c], Dezhi Ning[c], Ling Qian[b]

[a] *Department of Civil and Environmental Engineering, National University of Singapore, Kent Ridge, Singapore 117576, Singapore*
[b] *Department of Computing and Mathematics, Manchester Metropolitan University, Chester Street, Manchester M1 5GD, UK*
[c] *State key laboratory of Coastal and Offshore Engineering, Dalian University of Technology, Dalian 116024, China*

Abstract

A new three-dimensional immersed boundary method combined with the level set method for the interface capturing is developed to simulate the interaction between the fixed\moving structure and the two-phase fluid flow. The concept of the forcing point searching scheme developed for the two-dimensional situations in Yan et al. (2018) is extended in the present study to three dimensions, where the determination of intersections between the arbitrary body surface and the Cartesian background grid system is the major issue. This problem can be converted to the prediction of triangle-triangle intersection, which was traditionally solved from the geometrical point of view. Here, an algebraic algorithm is adopted for the triangle-triangle intersection, based on which the forcing points can be determined in the three-dimensional immersed boundary method. This algebraic algorithm is robust for any body geometry and easy for implementation. To demonstrate the accuracy and capability of the developed numerical model, three benchmark testing cases for water impact problems are conducted, including dam break over a fixed obstacle, water entry of a wedge and free decay of a bobber. Extensive comparisons with the experimental data and the numerical results obtained by other immersed boundary methods suggest that the developed immersed boundary method is accurate and effective for both fixed and moving bodies with complex geometries.

Keywords: Immersed boundary method, Level set method, Forcing point, Algebraic algorithm, Triangle-Triangle intersection, Water impact

1. Introduction

Fluid flow interaction with structures is a common problem in nature, which can find massive applications in various disciplines. In particular, with the increasing demand of modern society for energy, different types of marine structures have come into use in offshore and ocean engineering. Therefore, understanding the hydrodynamic performance of those structures under wave actions becomes an essential task for the safe design and efficient operation. Due to the geometry and character, some structures can be simplified to a

*Corresponding author
Email address: `w.bai@mmu.ac.uk` (Wei Bai)

two-dimensional (2D) problem. However, the realistic marine structures are complex and certainly three-dimensional (3D), which makes the modelling of 3D structures in ocean waves a very interesting research problem.

In a broad sense, there are two methods to model a structure in the computational domain in the mesh-based approach: body-fitted method and non-conforming method. The recently developed overset method (Ma et al., 2018; Chen et al., 2019) can be considered as a body-fitted method, which requires extensive interpolations to transfer information with the background grid. In the body-fitted method (Yan and Ma, 2007; Yang et al., 2008), curvilinear structured grids or unstructured grids that conform to the body boundary are generated. This method seems straightforward, however, it could require a heavy cost in computational time as well as manpower or even fail when dealing with a complicated body geometry with large displacement. In recent years, the non-conforming method becomes increasingly popular for simulating bodies in fluid-structure interactions. The cut cell method, one non-conforming method, may face difficulty in simulating irregular "interface-cells", which requires different "special treatments". Thus, the extension of cut cell method from 2D to 3D is sometimes prohibited if not impossible, although the limited successful work has been shown in Hu et al. (2013). The basic concept of another non-conforming method, the immersed boundary method (Wu et al., 2013; Wu and Young, 2014) is to add a body force to the momentum equations at certain points around the boundary, without the necessity of performing the mapping procedures, aiming to model the effect of investigated bodies in the flow. By means of this robust method, generation of grids can be greatly simplified. The immersed boundary method was firstly proposed by Peskin (1972) to simulate the cardiac flow with the feedback forcing scheme, which had one disadvantage of requiring small time step. To resolve such problem, Mohd-Yusof (1997) proposed a discrete forcing scheme, which was implemented successfully by Fadlun et al. (2000); Kim et al. (2001); Zhang et al. (2010); Wu (2019).

For the 3D situation, the immersed boundary method was extended by Fadlun et al. (2000) on a staggered grid with volume fraction to determine the forcing point location, which is rather simple but inaccurate. Both Fadlun et al. (2000) and Kim et al. (2001) conformed the background grids to the sphere surface by the use of cylindrical coordinate system. They aimed to enable the forcing points to coincide with the sphere surface manually. In addition, unstructured triangular grid was adopted by Gilmanov and Sotiropoulos (2005) in their 3D immersed boundary method to approximate the sharp body interface, where the nodes near the boundary were utilized to determine whether a given point was an internal or external forcing point, according to the sign of scalar product between the position vector and the normal vector. Based on the work in Kim et al. (2001), Kim and Choi (2006) applied the immersed boundary method to simulate moving bodies via the non-inertial reference frame. As the non-inertial reference frame system was fixed on the body, the procedure for determining forcing terms was the same as that for a stationary case. Similar to Gilmanov and Sotiropoulos (2005), Mittal et al. (2008) also employed the concept of scalar product to determine the forcing points. However, the scalar product method requires many product iterations in the whole domain, which causes the computational effort increase dramatically in the 3D cases.

Borazjani et al. (2008) proposed a curvilinear immersed boundary method for modeling complex 3D rigid bodies represented by unstructured triangular grids. In their work, the point-in-polyhedron identification procedure was used to locate forcing points, which starts by emitting a line from a given point P (to be identified) to a point S outside the polyhedron, and the number of intersections is counted to determine whether P is in the solid phase. Similarly, a ray-tracing procedure was adopted in Roman et al. (2009) to address situations involving multiple or concave immersed boundaries. However, both Borazjani et al. (2008) and Roman et al. (2009) did not show the procedure to handle the special situations when the ray passes through the vertices of a triangular cell. This special situation always causes confusion in the determination of intersection numbers, leading to the inaccurate forcing point positions.

Recently, Seo and Mittal (2011) developed a sharp-interface immersed boundary method with a partial cut-cell based approach to strictly satisfy the boundary condition with geometric conservation. Spurious pressure oscillations can be reduced, but the treatment was always complicated and the simulations were limited to single-phase flows. To suppress the spurious force oscillations in the simulations of moving bodies, a fully-implicit ghost-cell immersed boundary method was developed in Lee and You (2013), based on the work in Mittal et al. (2008), by combining the mass source/sink algorithm and implicit discretizations. Furthermore, Calderer et al. (2014) developed a CURVIB immersed boundary method for calculating the forces on a body by imposing the pressure projection boundary condition (PPBC). To define forcing points, the point-in-polyhedron algorithm in Borazjani et al. (2008) was used to classify all nodes in the computational domain. Again based on Mittal et al. (2008), Nicolaou et al. (2015) proposed a robust immersed boundary method on the curvilinear grids with semi-implicit discretizations to solve the Navier-Stokes equations. However, the force term applied in the solid phase required an iterative approach, which was time-consuming despite the smaller errors at the boundary. In Bihs and Kamath (2017), the approach for the geometry description with a local directional ghost cell method based on Berthelsen and Faltinsen (2008) was proposed. To determine the ghost cell, the extrapolation along the normal direction of the regular or irregular boundary was adopted, which needs to be conducted in the x-, y-, and z-directions, respectively.

For the 2D problem, the authors have developed an efficient and straightforward scheme on the Cartesian background grid to identify forcing points in the immersed boundary method (Yan et al., 2018), which exhibits the advantage of ease of implementation with desirable accuracy. In addition, it locates the u and v forcing points precisely to ensure the accurate boundary enforcement rather than introducing any assumption or approximation in the determination of forcing points. However, the extension from 2D to 3D problem requires tremendous effort. The discussion on past studies shows that approaches to locate 3D forcing points mainly include simplification for special bodies, scalar product of point vector and surface normal vector and ray-tracing scheme. They are all directed at adopting a geometrical view to obtain the positions of intersection and hence determine forcing points. Therefore, they may be more complicated and face more challenges when dealing with complicated 3D geometries. Different to the geometrical view, in the present study an algebraic algorithm for predicting triangle-triangle intersections is developed to locate forcing points. The algebraic algorithm was originally proposed in the discipline of computational graphics (Tropp et al., 2006), with the good features of being robust and efficient. By following the basic concept developed for 2D situations, the main advantages of the immersed boundary method developed in Yan et al. (2018) can be carried over to 3D situations. With the combination of level set method, several water impact problems with complex breaking surface are investigated in the present study. To validate the newly developed 3D immersed boundary method, the present results are compared with other experimental and numerical results extensively for both the fixed and moving 3D structures with complex geometries.

2. Two-phase flow model

To simulate the 3D incompressible viscous flows, a two-phase flow model is adopted in the present study. This model was developed in Archer and Bai (2015) where more details have been discussed, and only the key information is provided in the following. The fluid motion can be described by the Navier-Stokes equations,

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = \frac{1}{\rho}\left(-\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j}\right) + g_i + f_i, \tag{1}$$

and the continuity equation,

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2}$$

where i and j denote the x, y, z directions ($i, j = 1, 2, 3$), u_i is the fluid velocity, x_i is the spatial coordinate, t is the time, p is the pressure, g_i is the gravitational acceleration, ρ is the fluid density and f_i is the momentum forcing component used to enforce the desired boundary condition on an immersed boundary interface in the present study. τ_{ij} are the viscous stress components, which is dependent on the dynamic fluid viscosity μ. Here the Cartesian tensor notation is used.

The level set method is adopted to capture the interface (free water surface) between the two water-air phases. In the level set method, a signed distance function ϕ is defined throughout the domain to measure the shortest distance from the grid cell center to the interface. A positive value of ϕ indicates one phase while a negative value indicates the other. The zero value of level set function represents the interface position. The evolution of the level set function ϕ satisfies the following convective equation,

$$\frac{\partial \phi}{\partial t} + u_i \frac{\partial \phi}{\partial x_i} = 0. \tag{3}$$

As the two-phase flow model needs to consider air and water simultaneously, ρ and μ in the Navier-Stokes equations should be determined by the properties of the local fluid phase. However, the interface between the air and water is sharp, so the sudden change in the values of ρ and μ between two phases could cause instabilities when solving the Navier-Stokes equations. To overcome such problem, a smeared form of Heaviside function H is applied in a small band around the interface to smooth out ρ and μ between two phases. Therefore, ρ and μ can be calculated by

$$\rho(\phi) = \rho_{air} + H(\phi)(\rho_{water} - \rho_{air})$$
$$\mu(\phi) = \mu_{air} + H(\phi)(\mu_{water} - \mu_{air}) \tag{4}$$

where the subscripts *air* and *water* denote the values of air and water, respectively.

The Navier-Stokes equations are solved numerically by the finite difference method on a staggered grid system. The level set equation (Eq. 3) is descritised by a fifth-order scheme to ensure the accuracy of the calculation. For the detailed numerical descritisation and implementation, the reader can refer to Archer and Bai (2015).

To calculate the responses of a 3D moving body in 6 degrees of freedom, the motion equations for the translational and rotational responses can be adopted based on the Newton's second law. As the body only undergoes translational motions in the present study, the motion equations in the translational directions are considered, which take the following general form for a body with the damping and spring system,

$$M\frac{\partial^2 Y^i}{\partial^2 t} + C\frac{\partial Y^i}{\partial t} + KY^i = F^i_{fluid} + F^i_{ext}, \tag{5}$$

where the superscript i denotes the direction as in Eq. 2, Y^i is the displacement in the ith direction, M the mass of the moving body, C the damping coefficient, K the spring stiffness coefficient, F^i_{fluid} the force exerted by the fluid, and F^i_{ext} the external force. In the following cases, C is required to be determined by the numerical simulation, and the spring system is not considered. Furthermore, the structure is only subjected to the force exerted by the fluid F^i_{fluid}, which can be expressed as

$$F^i_{fluid} = \int_\Omega (-pn_i)d\Omega + \int_\Omega (\tau_{ij}n_j)d\Omega, \tag{6}$$

where the first and second parts on the right of the equation represent the pressure force and shear force obtained by the integration over the body surface Ω. Based on the obtained displacement, the velocity of the structure can be predicted by

$$u_i = \frac{\partial Y^i}{\partial t}, \tag{7}$$

where u_i is the velocity component at the ith direction.

3. 3D immersed boundary method

In the present method, the triangle to triangle intersection is solved algebraically that can locate the u, v, and w forcing points respectively. It guarantees the precise body boundary in a relatively coarser grid compared to other methods, such as the simplification for special bodies, a scalar product of point vector and surface normal, and the ray-tracing scheme. The scalar product and ray-tracing scheme require much computational cost for iterations, while the algebraic algorithm adopted here is easy for the speed-up in the matrix solution. Meanwhile, by solving the matrix the accuracy of forcing points can be ensured and the missing forcing points can be avoided.

3.1. Solid geometry description

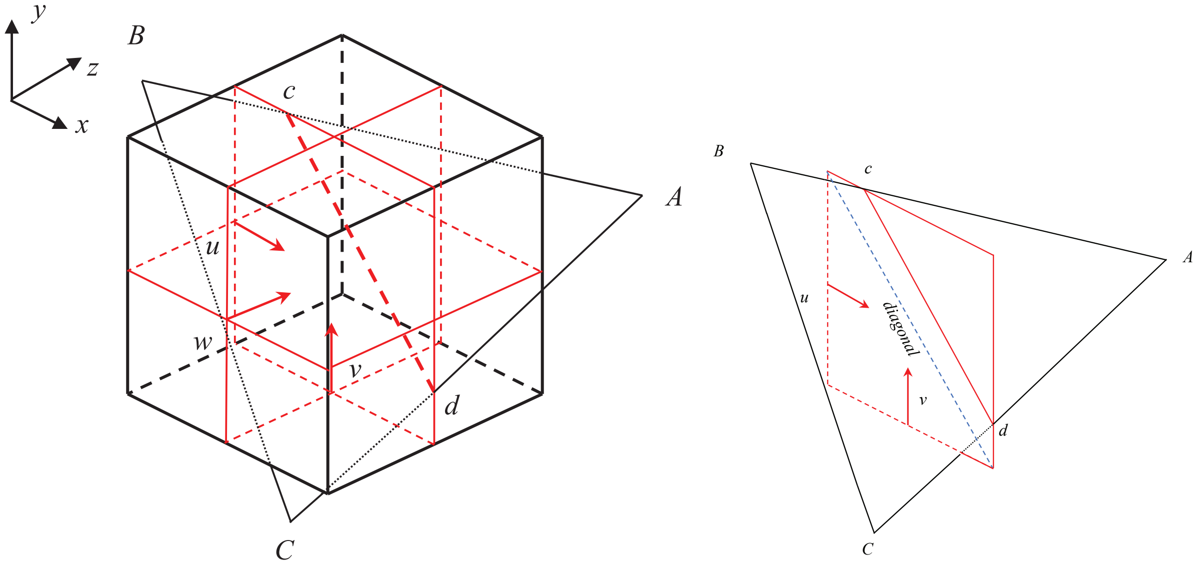
Firstly, structures in the computational domain need to be defined and described in a certain way. In the present study, the surface of a solid body is descritised by a series of unstructured triangular grids. The unstructured triangular grids are generated by means of the open source software "NETGEN" (Schőberl, 1997). NETGEN is an automatic mesh generation tool in both the 2D and 3D spaces, and it can generate triangular or quadrilateral grids, as well as tetrahedral grids. To generate unstructured triangular grids on the curved surface of a 3D solid body, the constructive solid geometries (CSG) or the standard template library (STL) file format is created first and then imported into NETGEN. Once the geometry of the solid body is defined by NETGEN, the information of nodes on the body surface is provided to the present numerical model for the subsequent determination of forcing points near the body surface.

3.2. Triangle-triangle intersection in 3D space

With the description of a solid body, it is possible to find the forcing points. In the present study, the forcing points locate outside the solid phase (surface mesh), which has been demonstrated to give better performance in Yan et al. (2018) for the 2D situation. In order to apply the basic concept of the improved forcing point searching scheme developed for the 2D case in Yan et al. (2018) to the present 3D problem, the key is to project the body surface onto the corresponding plane, so that the 3D problem can be converted to the 2D problem to some extent. To construct the projection area, three bounded planes are adopted, which are shown in Fig. 1(a) as red squares. Those three bounded planes are denoted as the uv, uw, vw planes, respectively, according to the directions of two velocity components that belong to the plane, each of which is also perpendicular to a particular coordinate axis. For the convenience of capturing the intersection between the plane and the body surface, each bounded plane is further divided into two triangles. Therefore, the problem is transferred to the determination of 3D triangle-triangle intersections, before the forcing points can be found.

For the purpose of further demonstration, one triangle on the body surface ($\triangle ABC$ shown in Fig. 1(a)) is taken as an example. To search the forcing points, one needs to find the interaction between $\triangle ABC$ and the uv bounded plane (for example), which is the line segment cd as shown in Fig. 1(a). Fig. 1(b) shows the sketch of $\triangle ABC$, uv bounded plane and intersection line segment cd only. It should be noted that the bounded plane is divided into two triangles by a diagonal line. By using the triangle-triangle intersection algorithm, the intersection line cd can be found. As a result, the dimension of the problem is reduced from three to two in a sense.

As the intersection between the edge of one triangle with another triangle can be described by a linear equation, there are in total six such equations to be solved for the triangle-triangle intersection problem. A few fast representative algorithms (Moller, 1997; Held, 1997; Guigue and Devillers, 2003) were proposed by



(a) (b)

Figure 1: Triangle-triangle interaction in the simulation: (a) overall view (b) $\triangle ABC$ for clear demonstration.

the utilization of the intersection line between the planes supported by the two triangles. In addition to the above-mentioned algorithms to solve the intersection problem geometrically in the conventional view, Tropp et al. (2006) proposed an algebraic approach. As there is a strong relationship among the linear equations, this relationship can be applied to expedite computational efficiency of the solution. In other words, the algebraic algorithm requires fewer arithmetic operations than those geometric algorithms, and the accurate intersection coordinates could be obtained with a much lower effort than before. Thus, we introduce the algebraic algorithm into the development of 3D immersed boundary method.

For completeness, we recapitulate the theory for the intersection between two triangles presented in Tropp et al. (2006) in the following. In this algorithm, Tri A and Tri B are defined as two triangles. The intersection between Tri A and Tri B means that the edges belonging to one triangle intersect the other triangle. With all possible edge being examined, the intersection could be determined. To demonstrate, define $\overrightarrow{p}_1$ and $\overrightarrow{p}_2$ as the edges of Tri B sharing the same vertex P_1, and the edges of Tri A originating from vertices Q_i as $\overrightarrow{q}_i$ $(1 \leq i \leq 3)$ (see Fig. 2). To reside the intersection coordinates between two supporting planes of two triangles, the following set of equations is solved:

$$P_1 + \alpha_1 \times \overrightarrow{p}_1 + \alpha_2 \times \overrightarrow{p}_2 = Q_i + \beta_i \times \overrightarrow{q}_i \tag{8}$$

where α and β_i are the scalar factor to be determined in the following content.

As the intersection point is inside the triangle, Eq. 8 should be solved with considering the inequalities: $0 \leq \beta_i \leq 1$, $\alpha_1\alpha_2 \geq 0$ and $\alpha_1 + \alpha_2 \leq 0$. Six such intersection tests ought to be carried out: three to examine

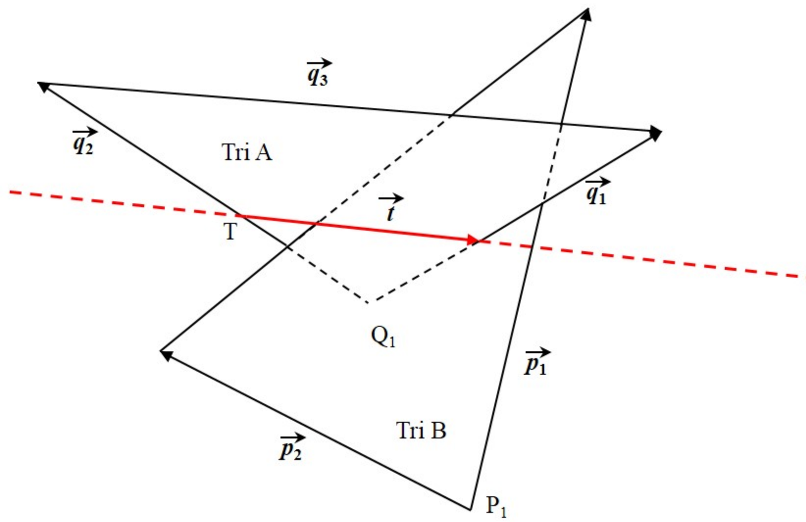


Figure 2: Problem setting for triangle-triangle intersection.

whether Tri A intersects the edges of Tri B, and three to examine whether Tri B intersects the edges of Tri A. By reusing the common elements and utilizing the linearity of matrix operations, the algebraic algorithm is applied only to three equations.

In the beginning, three linear equations can be partially solved to determine the parameters β_i, $1 \leq i \leq 3$, namely the three edges of Tri A intersecting with the supporting plane of Tri B (Eq. 8). The values of β_i can determine whether a fast rejection could occur, concluding no intersection. If such rejection does not happen, the values of β_i are combined with the vertices and edges to construct the intersection line between Tri A and the supporting plane of Tri B (vector $\overrightarrow{t}$ shown in Fig. 2). Thus, the 3D problem is reduced to a planar intersection between the intersection line segment $\overrightarrow{t}$ and Tri B. The flow chart of different stages of this algorithm is shown in Figure 3.

At the first step, Eq. 8 can be rewritten in the matrix form with $r_i = Q_i - P_1$,

$$(\overrightarrow{p}_1 \quad \overrightarrow{p}_2 \quad \overrightarrow{q}_i) \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ -\beta_i \end{pmatrix} = (r_i). \tag{9}$$

Letting the matrix $\mathbf{A}(\overrightarrow{q}_i) = (\overrightarrow{p}_1 \quad \overrightarrow{p}_2 \quad \overrightarrow{q}_i)$, Eq. 9 can be written in the simplified form,

$$\mathbf{A}(\overrightarrow{q}_i)\mathbf{x}_i = r_i \tag{10}$$

where $\mathbf{x}_i = (\alpha_1, \alpha_2, -\beta_i)$. As the intersection point always resides on the edge $\overrightarrow{q}_i$, the legal β_i is subjected to the inequality $0 < \beta_i < 1$. β_i is obtained via the determinants, $\beta_i = -|\mathbf{A}(\overrightarrow{q}_i)|/|\mathbf{A}(r_i)|$. Thus, the equivalent inequality condition becomes $0 < \beta_i|\mathbf{A}(r_i)|^2 < |\mathbf{A}(\overrightarrow{q}_i)|^2$ mathematically. As a result, $0 < -|\mathbf{A}(r_i)||\mathbf{A}(\overrightarrow{q}_i)| < |\mathbf{A}(r_i)|^2$ can be employed to examine whether the division is required.

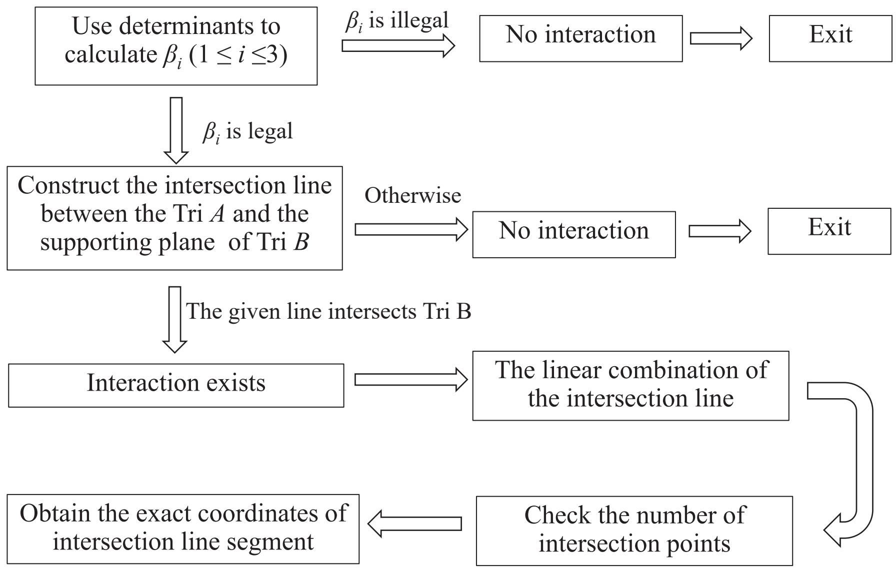


Figure 3: The flow chart for triangle-triangle intersection.

At the second stage, if all the values of β_i are illegal, Tri A is located on one side of the supporting plane of Tri B without any intersection or parallel to the supporting plane of Tri B. This situation concludes no intersection directly. If $|\mathbf{A}(\overrightarrow{q}_i)| = 0$ is always satisfied, Tri A and Tri B are overlapped in one plane. Otherwise, discussion on the common situation continues as follows.

At the third step, the following equations can be utilized to obtain the intersection point T and the line segment $\overrightarrow{t}$. For instance, if the legal parameters (β_1, β_2) exist, T and $\overrightarrow{t}$ can be written as,

$$T = Q_1 + \beta_1 q_1, \overrightarrow{t} = \beta_2 q_2 - \beta_1 q_1. \tag{11}$$

At the stage four, two cases occur when the two triangles intersect each other: 1) there is at least one intersection between the surface of Tri B and the line segment $\overrightarrow{t}$; 2) the line segment $\overrightarrow{t}$ is fully enclosed by Tri B. Thus, the following equations for intersection coordinates shall be solved:

$$\begin{pmatrix} P_1 + \delta_1 \overrightarrow{p}_1 = T + \gamma_1 \overrightarrow{t} \\ P_1 + \delta_2 \overrightarrow{p}_2 = T + \gamma_2 \overrightarrow{t} \\ P_1 + \delta_3 (\overrightarrow{p}_2 - \overrightarrow{p}_1) = T + \gamma_3 \overrightarrow{t} \end{pmatrix}. \tag{12}$$

The legal values of β_i lead to the residence of intersection on the edge $\overrightarrow{p}_i$. Only when $0 < \gamma_i < 1$ the intersection points are on the line segment $\overrightarrow{t}$. Only when both the inequalities ($0 < \gamma_i < 1, 0 < \beta_i < 1$) are satisfied, the intersection exists between the edge $\overrightarrow{p}_i$ and the line segment $\overrightarrow{t}$, with the coordinate of $\mathbf{X}_i$,

$$\mathbf{X}_i = P_i + \delta_i \overrightarrow{p}_i \tag{13}$$

At the final optional fifth stage, the parameters (β_i) and vectors can be combined to obtain the exact intersection coordinates. T and $T + \overrightarrow{t}$ or $\mathbf{X}_i$ and $T + \overrightarrow{t}$ shall be the required endpoints of the line segment.

3.3. Determination of forcing points and forcing terms

After the determination of the intersection line segment (such as cd shown in Fig. 1), the forcing points can be predicted. Here, take the uv plane as an example for the purpose of illustration. In Fig. 4(a), the red solid line is an intersection line segment that has been determined by the algorithm discussed in the last section, and $\mathbf{n}$ is the unit normal vector pointing outwards. If the slope of the line segment is smaller than that of the diagonal line, the line segment is extended to a new grid in the y direction. However, if the line segment has already touched both two vertical sides of the present grid, the extension is not required. It is noted that the direction of extension of the line segment depends on the normal vector direction. The u and v positions closest to the line segment can be calculated by the linear equation of intersection line segment. According to the normal direction, u and v forcing points are located in the same direction of $\mathbf{n}$, shown as the red u and v arrows in Fig. 4(a). Otherwise, the forcing points are located at the black u and v positions if the normal direction is in the opposite way. On the other hand, if the slope of the line segment is larger than that of the diagonal line, the extension of line segment is in the horizontal direction, as shown in Fig. 4(b). Similar to Fig. 4(a), the u and v positions closest to the line segment can be determined. Based on the normal direction, the u and v forcing points are located in the same direction of $\mathbf{n}$, shown as the red u and v arrows in Fig. 4(b). Otherwise, the forcing points are the black u and v if the normal vector direction is opposite. The following equation is aimed to represent such procedure,

$$\text{forcing point position} = \begin{cases} (x_i, y_{j+1}), & \text{if} \quad n_j \geqslant n_i \\ (x_{i+1}, y_j), & \text{if} \quad n_i \geqslant n_j \end{cases} \tag{14}$$

where n_i, n_j are the x-,y- components of the normal vector $\mathbf{n}$.

It is the same to determine the information of u, w forcing points in the uw plane and v, w forcing points in the vw plane, respectively. Based on our developed algorithm for the 2D case in Yan et al. (2018), the forcing points closest to the body surface are selected and stored for the interpolation procedure.

Upon the location of forcing points is determined, the velocity at the forcing point v_f needs to be predicted by the interpolation (more details are also given in Yan et al., 2018). Once the velocity at the forcing point is available, the forcing term component at the forcing point is predicted based on the method described in Mohd-Yusof (1997),

$$f_i = \frac{v_f - v_i^n}{\Delta t} - RHS_i^n. \tag{15}$$

where RHS is the sum of the convective, viscous, pressure gradient and body force terms in the governing equations, and the superscript n denotes the values taken from the previous time step.

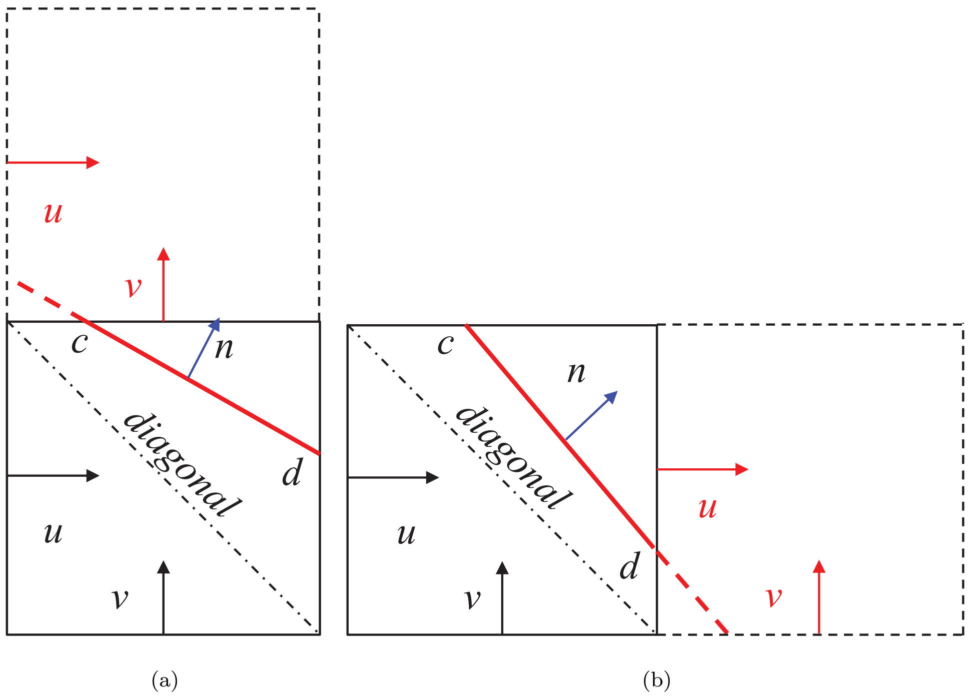


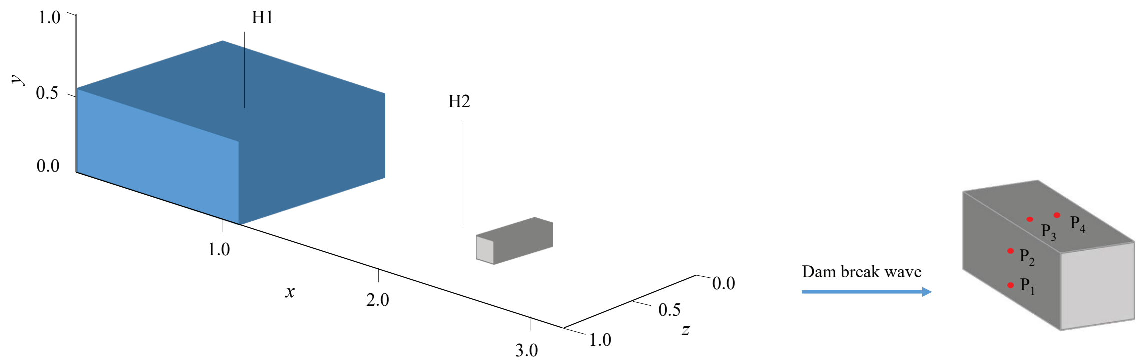
Figure 4: Sketch of determination of forcing points in a 2D plane: (a) slope is smaller than diagonal line; (b) slope is larger than diagonal line.

4. Validations and applications

To validate the effectiveness and accuracy of the proposed 3D immersed boundary method, three cases are conducted: 3D dam break over a cuboid, free fall of a 3D wedge, and free decay of a bobber. It aims to demonstrate the capability of our model to simulate fixed and moving 3D complex bodies in the fluid flow with complicated free surface.

4.1. 3D Dam break over a cuboid

In this case, the numerical model is adopted to simulate a physical experiment carried out in a short tank of 3.22m × 1.0m × 1.0m, at the laboratory of Maritime Research Institute Netherlands (MARIN), which was reported in Kleefsman et al. (2005). To simulate the dam break, a door encloses a water column with 1.22m width and 0.55m height on the right side of the tank. After the door is pulled up, the water column breaks and flows to the other side of the tank. A cuboid is fixed to resemble a scaled model of a deck-house on the deck of a vessel in the tank. Water heights and pressures were measured at the specified positions as shown in Fig. 5. As the behaviour in the middle of the z plane is more concerned, the pressure gauges are all set in the symmetrical plane in the z direction. The coordinates (x, z) of the water gauges H1 and H2 are (0.56, 0.5) and (2.22, 0.5). The coordinates (x, y, z) of the pressure gauges P1 $\sim$ P4 are (2.4, 0.025, 0.5), (2.4, 0.01, 0.5), (2.425, 0.16, 0.5) and (2.45, 0.16, 0.5). The unit of the coordinates is meter.



(a) Fluid domain (b) Pressure position

Figure 5: Sketch of 3D dam break over a cube and measured positions.

The surface of the cuboid needs to be described first. In the present numerical simulation, the cuboid is discretized with 12 vertices and 20 triangular surface elements, as shown in Fig. 6. To test the mesh convergence, three grids around the structure are examined with intervals of 0.04m × 0.04m × 0.04m (coarse mesh), 0.02m × 0.02m × 0.02m (medium mesh) and 0.01m × 0.01m × 0.01m (fine mesh) in the x, y and z directions respectively. In the other area, the uniform grids of 0.04m × 0.04m × 0.04m are generated. The pressures at the gauges P1 and P3 are presented for those three grids in Fig. 7. From the figure, it can be observed that the results for the medium and fine meshes are close to each other, while the result for the coarse mesh deviates much from the other two results. Therefore, the convergent results are achieved when the medium mesh is used.

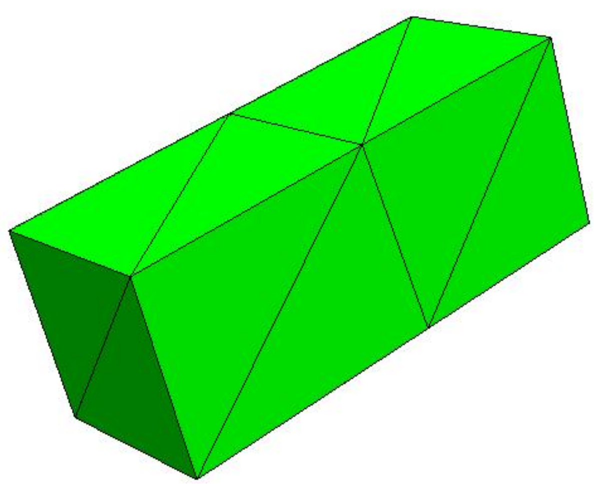
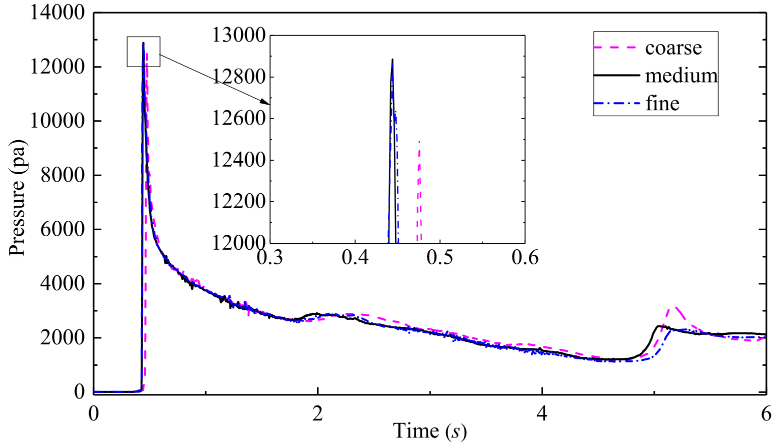
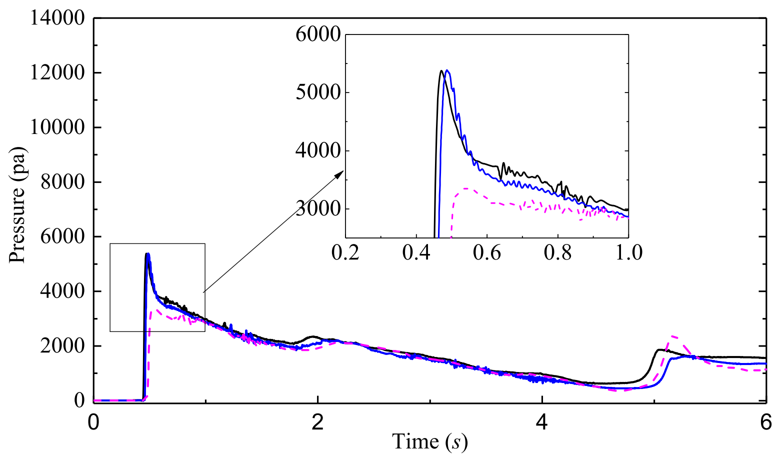


Figure 6: Mesh details of the surface of a cube (12 nodes and 20 surface elements).

Fig. 8 presents a comparison of time histories of water height with the experimental data and other numerical results. The results in Kleefsman et al. (2005) were computed using the model Comflow and Gu



(a) P1

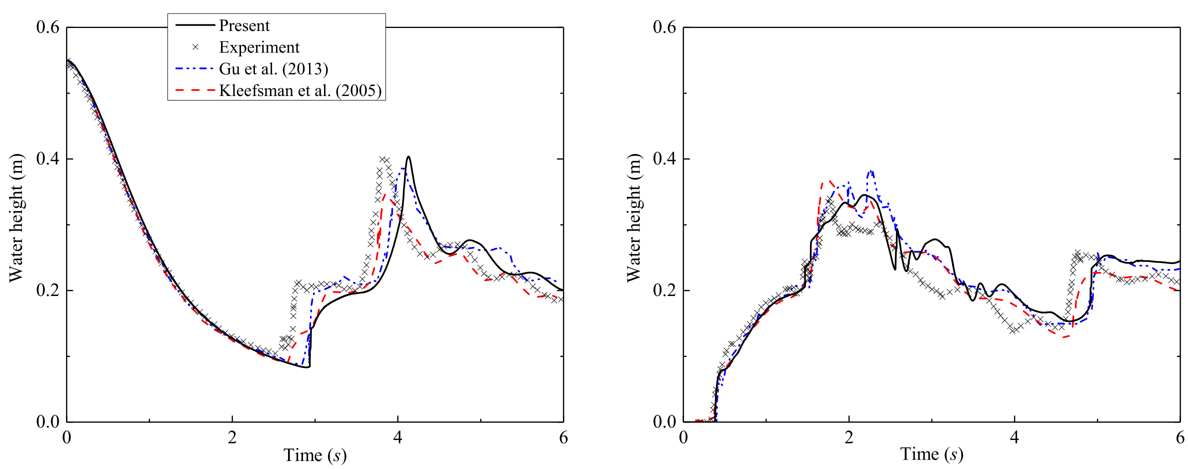


(b) P3

Figure 7: Time history of pressures at P1 and P3 for the mesh convergence test.

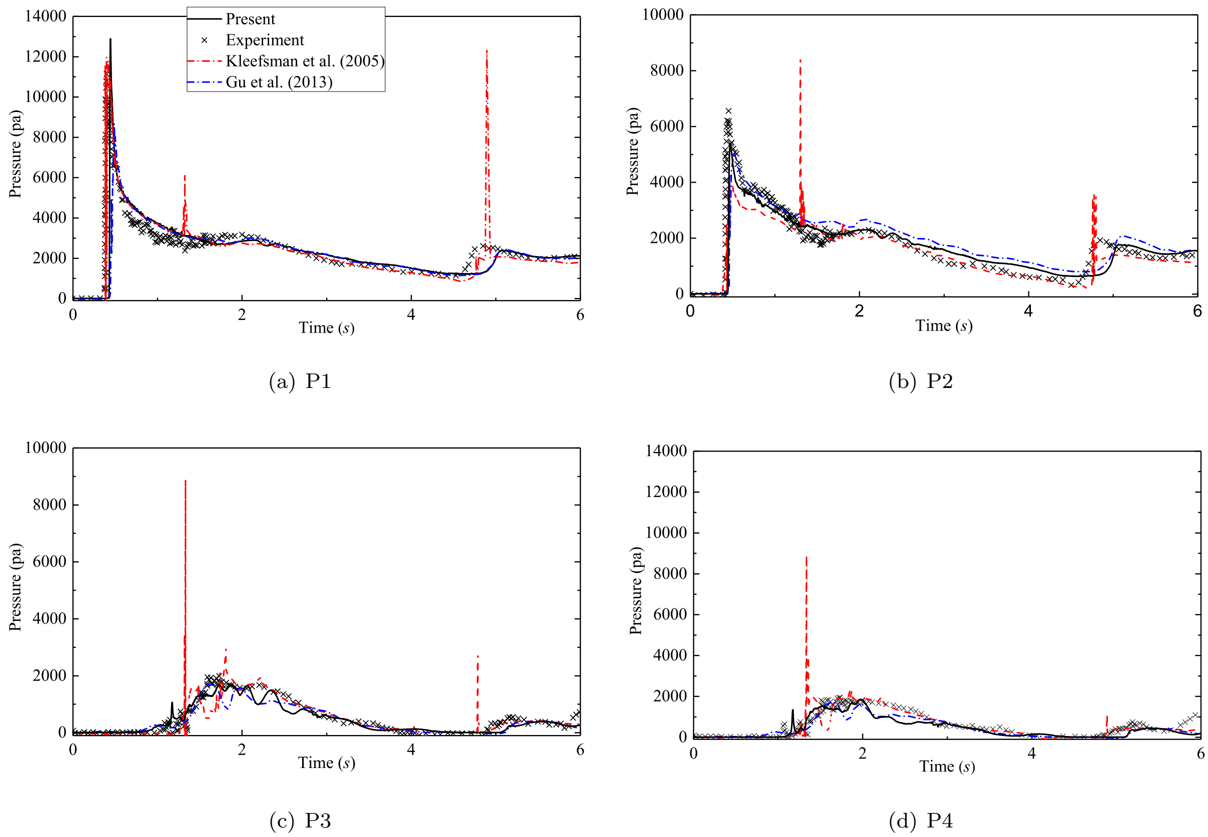
et al. (2013) adopted the partial cell technique. In general, all the three numerical results deviate a little from the experimental one with a phase lag. However, the present results seem to capture the magnitude better, while the numerical results of both Kleefsman et al. (2005) and Gu et al. (2013) underestimate the peak value in Fig. 8(a) and overestimate the peak value in Fig. 8(b). At the wave gauge H1, the present result shows a delay in the peak value, which means the reflected water from both the cuboid and the end wall arrives the wave gauge slightly later. This is probably due to the fact that one disadvantage of the level set method adopted in the present study for the interface capturing lies in the poor guarantee of conservation of mass for the complex water surface with the fluid-structure interaction. The developed 3D immersed boundary method seems to perform well, as the free surface elevation at the gauge H2 which is just in front of the cuboid shows a good agreement with the experimental data.

To further demonstrate the capability, the time histories of pressures at four positions are presented in Fig. 9. It can be seen that there are suspicious spurs in the results of Kleefsman et al. (2005), while the present results and the results of Gu et al. (2013) shows the good feature of stability. Compared to the results of Gu et al. (2013), the present results agree better with the experimental data, as their results underestimate the first peak. Fig. 10 shows the water surface profiles at $t = 0.0$s, 0.5s, 0.75s, 1.0s, 1.25s and 2.0s respectively, with the grid of 0.01m $\times$ 0.01m $\times$ 0.01m. As the water impacts on the cuboid, the water surface exhibits the process of separation, breaking and mixing. The splashing water jets occur between the time instants $t = 0.75$s and $t = 1.25$s. The complicated water surface is generated mainly when the water impacts on the front side of the obstacle and the end wall of the tank, as shown in Figs. 10(c) and 10(d). Fig. 10(e) captures the most chaotic and breaking features of the water surface.



(a) H1 (b) H2

Figure 8: Comparison of water heights in the gauges H1 and H2 with the experimental and other numerical results.



(a) P1

(b) P2

(c) P3

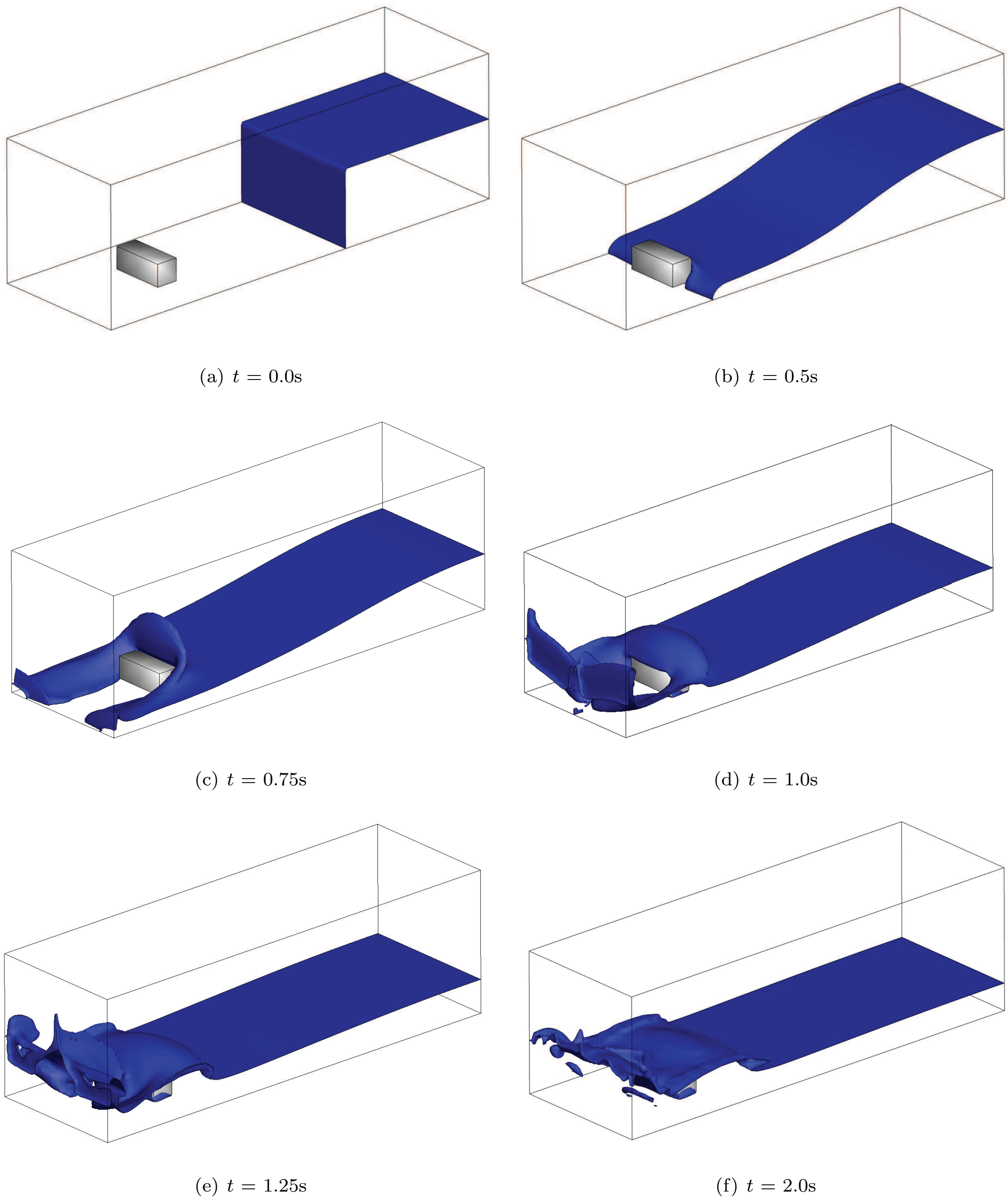
(d) P4

Figure 9: Comparison of time history of pressures at P1, P2, P3, and P4 with the experimental and other numerical results.

4.2. Free falling of a 3D wedge

In this section, a freely falling 3D wedge is investigated, which is more challenging than the dam break past a fixed body studied in the last section, as shown in Fig .11. Free falling of the 3D wedge was investigated experimentally in Yettou et al. (2006), where the position and velocity of the wedge were measured. Calderer et al. (2014) and Bihs and Kamath (2017) also worked on the same problem numerically using the immersed boundary method. The differences between the present and the previous immersed boundary methods mainly lie in the search of forcing points and the implementation of boundary condition.

This symmetric wedge with a 25 degree dead-rise angle weights 94kg, equivalent to a body with the density of 466 kg/m^3. Initially, the wedge falls freely from the position 1.3m above the still water. For simplification, we set the velocity of the wedge as 5.0m/s ($\sqrt{2gs}$, where the distance $s = 1.3$m) at the initial instant of wedge penetrating the still water in the numerical simulation. The dimension of the wedge in the numerical simulations is taken as 1.2m in both the spanwise and longitudinal directions. Correspondingly, the length and width of the channel (fluid domain) are 4.0m and 2.0m. It ensures that there is a 0.4m gap between the wedge and each channel wall in the spanwise direction. The gap in the longitudinal direction is



(a) $t = 0.0$s

(b) $t = 0.5$s

(c) $t = 0.75$s

(d) $t = 1.0$s

(e) $t = 1.25$s

(f) $t = 2.0$s

Figure 10: Water surface profiles at six different time instants for the dam break.

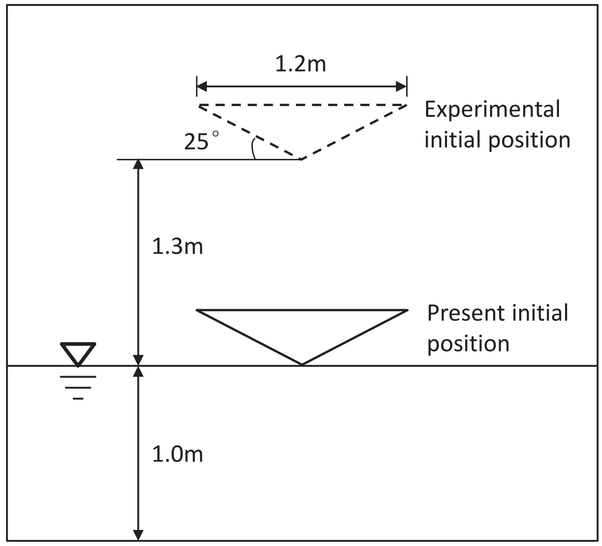


Figure 11: 2D sketch of water entry of the free falling wedge.

1.4m to each channel wall to avoid the influence of walls. The channel length in the present simulations is smaller compared to those in the experiment (Yettou et al., 2006) and other numerical simulations (Calderer et al., 2014, Bihs and Kamath, 2017). However, as the time duration is very short before we terminate the simulation, the wave reflection from the longitudinal walls cannot affect the central fluid area of interest. The uniform grid interval is 0.04m, and it is reduced to 0.01 around the structure in both the horizontal and vertical directions.

The wedge velocity at the initial stage is presented and compared with the experimental and the numerical results in Calderer et al. (2014), as shown in Fig. 12. In Calderer et al. (2014), the combination of a point-in-polyhedron algorithm for defining forcing points and the PPBC correction is adopted. Despite the PPBC improvement in Calderer et al. (2014) made to the standard method of Borazjani et al. (2008), the present numerical results of wedge velocity are still in a better agreement with the experimental data. At this stage, the present results and the experiment data show the evident inflection point around $t = 0.013$s, which is not well captured in the results of Calderer et al. (2014). Before $t = 0.013$s, the numerical results based on the proposed method agree much better with the experimental data than the other published results. In addition, Calderer et al. (2014) used much smaller grid spacings, which can further demonstrate the advantage of the present algebraic algorithm for the forcing point searching.

To extend the time duration in the comparison, both the wedge position and velocity in the whole process are compared with the experimental data and the numerical results in Calderer et al. (2014) and Bihs and Kamath (2017), as shown in Fig. 13. It is noted that the wedge position is recorded at the keel. In Bihs and Kamath (2017), a ray-tracing algorithm was adopted to locate forcing points. According to the comparison, it can be observed that the present numerical results almost override the results of Calderer et al. (2014),

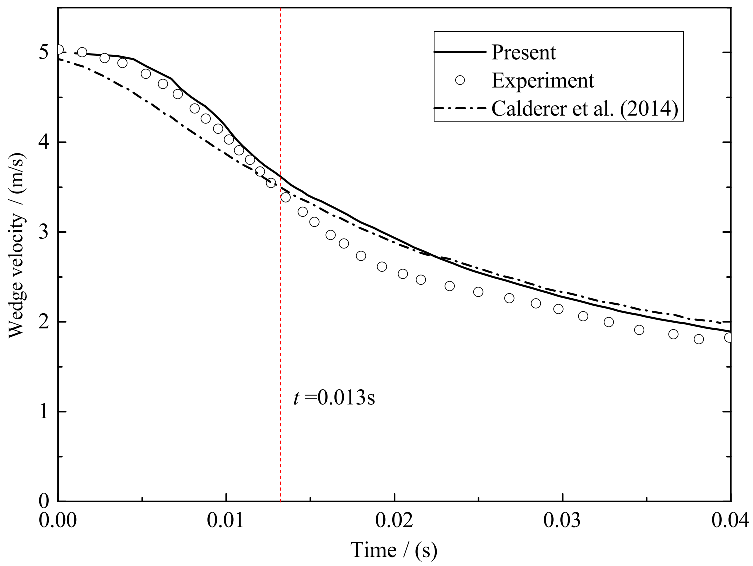
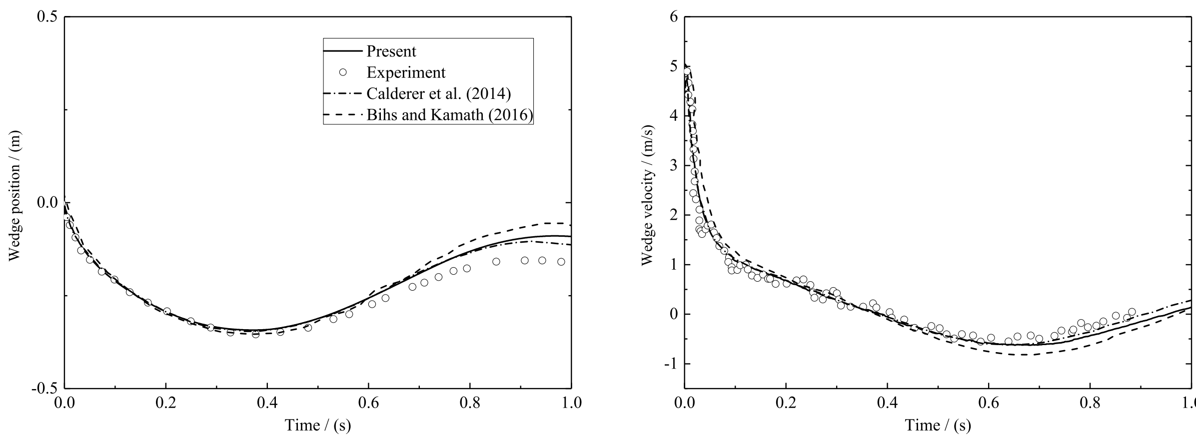


Figure 12: Comparison of the wedge velocity at the initial stage between the present, experimental, and other numerical results.

both of which agree well with the experimental data. However, one should note that much coarser grids are adopted in the present numerical simulations, compared to that in Calderer et al. (2014). Moreover, it is evident that the results for both the wedge position and velocity in Bihs and Kamath (2017) show more discrepancies with the experimental data, in the comparison with the present numerical simulations.



(a) (b)

Figure 13: Comparison of the wedge position and velocity between the present, experimental, and other numerical results: (a) wedge position; (b) wedge velocity.

Fig.14 shows the time history of slamming force on the whole wedge. The peak value of the slamming force is about 23525N, occurring at the time instant $t = 0.013$s. The largest slamming force corresponds to the largest acceleration according to the Newton's law. At this instant, the wedge velocity in Fig.13(b) decreases most sharply. In addition, the water surface profiles at several time instants are given in Fig. 15, where the position of the wedge and the water jet are shown. The wedge penetrates the water initially and floats upwards after $t = 0.5$s. A weak water jet around the wedge appears at $t = 0.7$s. The water surface looks similar to the "shipping-wave" with two wave crests. According to the comparison with the numerical results of the other two immersed boundary methods in Calderer et al. (2014) and Bihs and Kamath (2017), it can be seen that the surface profile is similar in all the studies when the 3D wedge is fully submerged. The surface profile from Bihs and Kamath (2017) looks more complicated than the other two, which seems unreal due to the errors in the predicted wedge velocity and position, as indicated in Fig. 13.

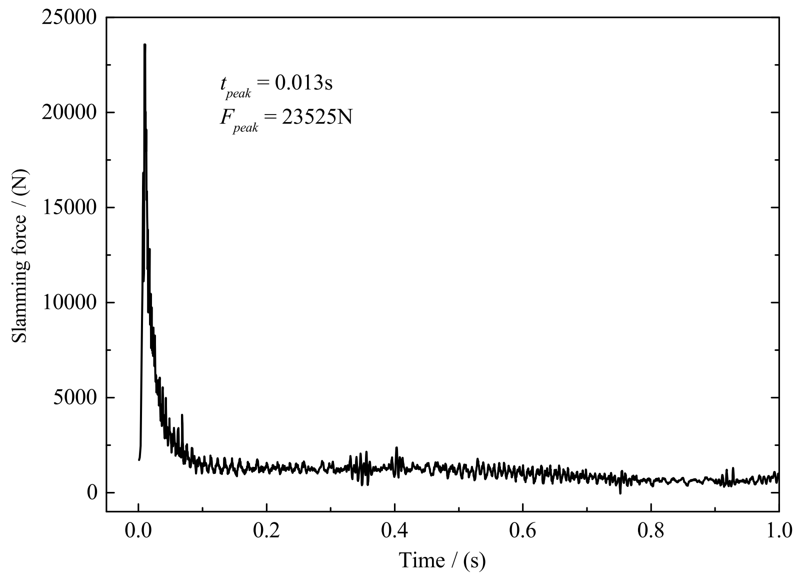
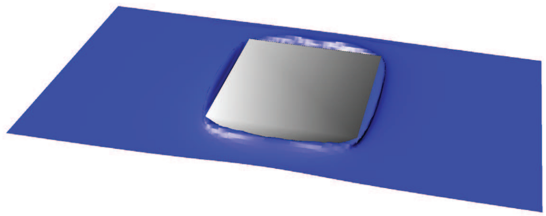


Figure 14: Time history of slamming force on the wedge.

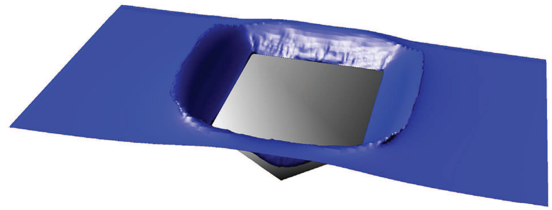
Finally, numerical results of the vector field at the middle vertical section are shown in Fig. 16. It can be seen that around $t = 0.025s$, the velocity magnitude approaches the maximum at about 10m/s, which is consistent with the value reported in Bihs and Kamath (2017). The wedge penetrates the water sharply at $t = 0.2s$, as shown in Fig. 16(c). At the following time instants, the complicated water jet occurs and propagates to the walls. In addition, the vorticity appears and develops around the two tips of the wedge.

4.3. Decay of a bobber

To further demonstrate the capability of the developed 3D immersed boundary method in simulating a complex body geometry, a free decay bobber is considered in this section. In this test, a bobber undergoing oscillatory heavy motions is considered, which is controlled by its own weight m_f and a counterweight m_c. A pulley system is adopted to connect those two weighting systems, as shown in Fig. 17(a). According to the



(a) $t = 0.1s$

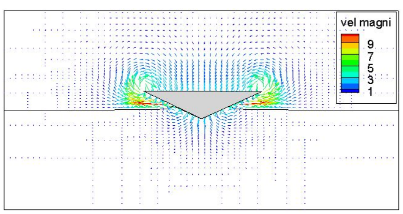


(b) $t = 0.3s$

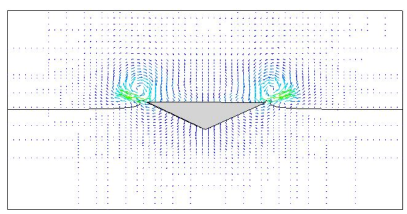
(c) $t = 0.5s$

(d) $t = 0.7s$

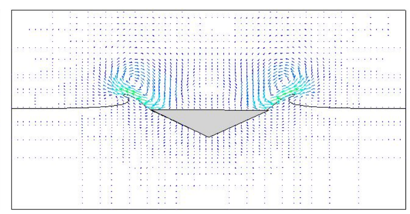
Figure 15: Water surface profiles at four different time instants for the free falling of the wedge.



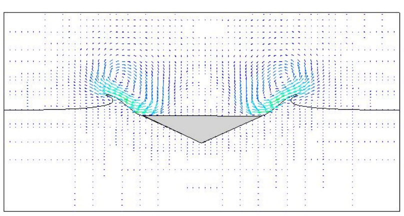
(a) $t = 0.025s$



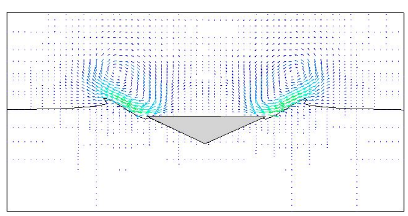
(b) $t = 0.1s$



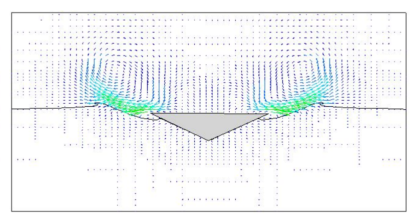
(c) $t = 0.2s$



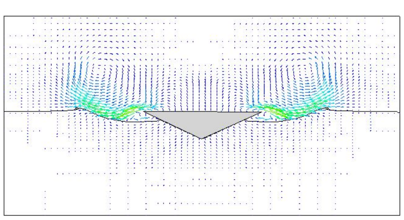
(d) $t = 0.3s$



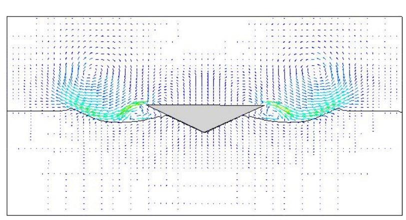
(e) $t = 0.4s$



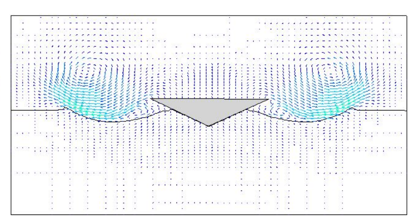
(f) $t = 0.5s$



(g) $t = 0.6s$



(h) $t = 0.7s$

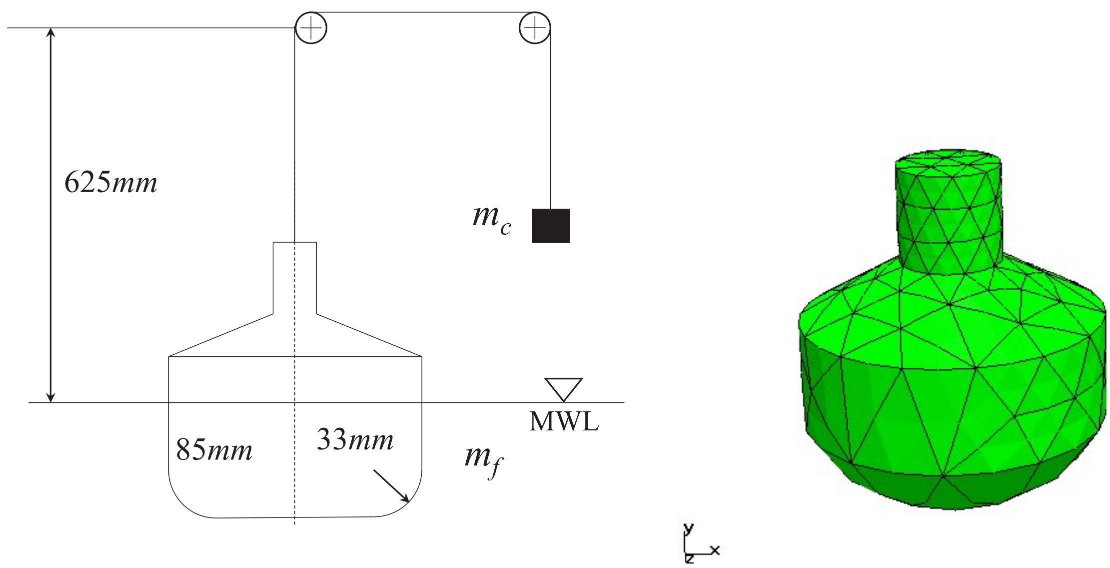


(i) $t = 0.8s$

Figure 16: Vector field at different time instants for the free falling of the wedge.

Newton's second law, the motion of the system is described by the formula $(m_f + m_c)a_y = F_y + (m_f - m_c)g$, where a_y is the acceleration of the whole system in the vertical direction and F_y is the slamming force acting on the bobber. It is noted that the positive direction of force, displacement, acceleration is pointing upwards.

Based on Thomas et al. (2008) and Hu et al. (2013), the geometry of the bobber is the combination of a flat-bottomed cylinder of the radius 74mm, a 30° cone, and a cylinder with the radius 25mm. The flat-bottomed cylinder is unsharpened around the corner with the radius 33mm. The height of the bottom cylinder is 85mm, and the grids to describe the bobber surface are shown in Fig. 17(b). Initially, the bobber is placed 0.01m above the still water (0.5m water depth) before the free motion. The whole fluid domain is 2m × 1m × 2m, with a grid of 0.01m × 0.01m × 0.01m around the bobber. To locate the forcing points, the bobber is discretized by 116 triangular surface elements.



(a) (b)

Figure 17: Sketch of bobber (a) and its surface mesh (b).

In the numerical simulation, m_f and m_c are set as 2.1kg and 1.2kg, respectively. Due to the difference in the weights of bobber and counterweight, the bobber could penetrate the water. In the water, the bobber is subjected to the hydrodynamic force, which leads to the oscillation of the bobber. In Calderer et al. (2014) and Bihs and Kamath (2017), the artificial damping was added into the motion equations for the heave motion by introducing the damping coefficient $C = 0.275$ in Eq. 5, in order to account for the friction of the experimental apparatus. However, as there is no friction of the experimental apparatus in the numerical simulation, the damping coefficient could be different. Thus, different values of damping coefficient are tested in the present study to evaluate the influence of damping, and the numerical results of vertical displacement of the bobber with different damping coefficients are compared in Fig. 18. The larger damping coefficient results in the smaller oscillation amplitude, and for the present case the damping coefficient $C = 1.5$ seems to best fit with the experimental data.

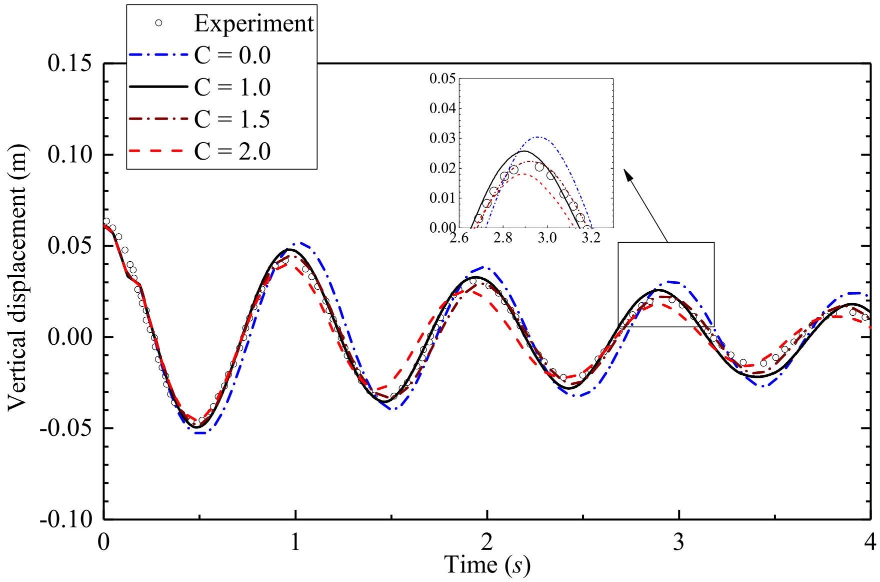


Figure 18: Numerical results of vertical displacement of the free decay bobber under different artificial damping coefficients.

The comparison of vertical displacement is shown in Fig. 19, in which the experimental data in Thomas et al. (2008), the numerical result from Hu et al. (2013) and the present results are included. In the first 4 periods, it shows that the present numerical result obtains a obviously better agreement with the experiment than that in Hu et al. (2013). With further increase of time, both numerical results show some discrepancies with the experiment. However, the present results show no phase lag with the experimental data, while there is an evident delay in the numerical results from Hu et al. (2013). In addition, the numerical result from Hu et al. (2013) are severely over-predicted compared to the present numerical results, as the damping was not considered in Hu et al. (2013).

For a further comparison, the heave force and vertical velocity of the bobber are shown in Fig. 20(a) and Fig. 20(b). It can be seen that the agreement is not very satisfactory both for the phase and amplitude, especially after $t = 2.5$s. It seems that the present numerical results may be more reliable, as a better agreement has been achieved between the present numerical results and the experimental data for the displacement shown in Fig. 19. Finally, the snapshots of water surface closed to the bobber are presented in Fig. 21. The figure shows the oscillation of the bobber and the surface profile around the bobber at different time instants. The free surface is irregular, especially at $t = 0.9$s when the discrete water volume appears. At $t = 0.5$s, there is a step-shape free surface profile due to the bobber geometry. From this case, the present new immersed boundary method is demonstrated to be robust and accurate even for a relatively complex body geometry.

Fig. 22 shows the pressure distribution at six different time instants at the central vertical section of the

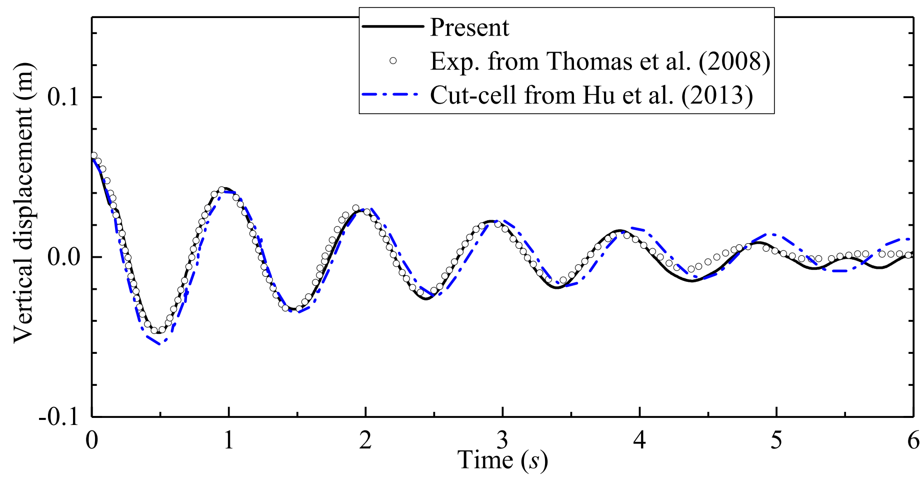
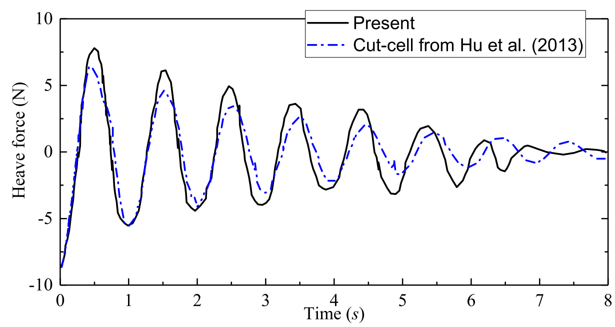
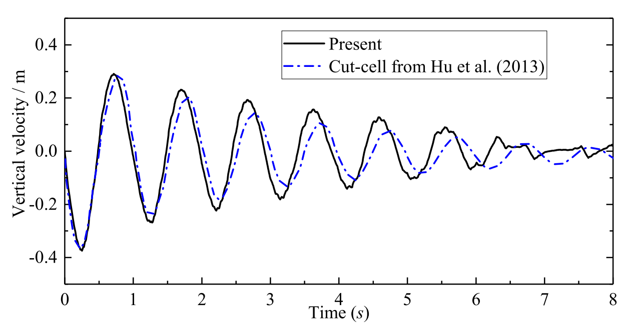


Figure 19: Comparison of time history of vertical displacement with the experimental data and other numerical results.

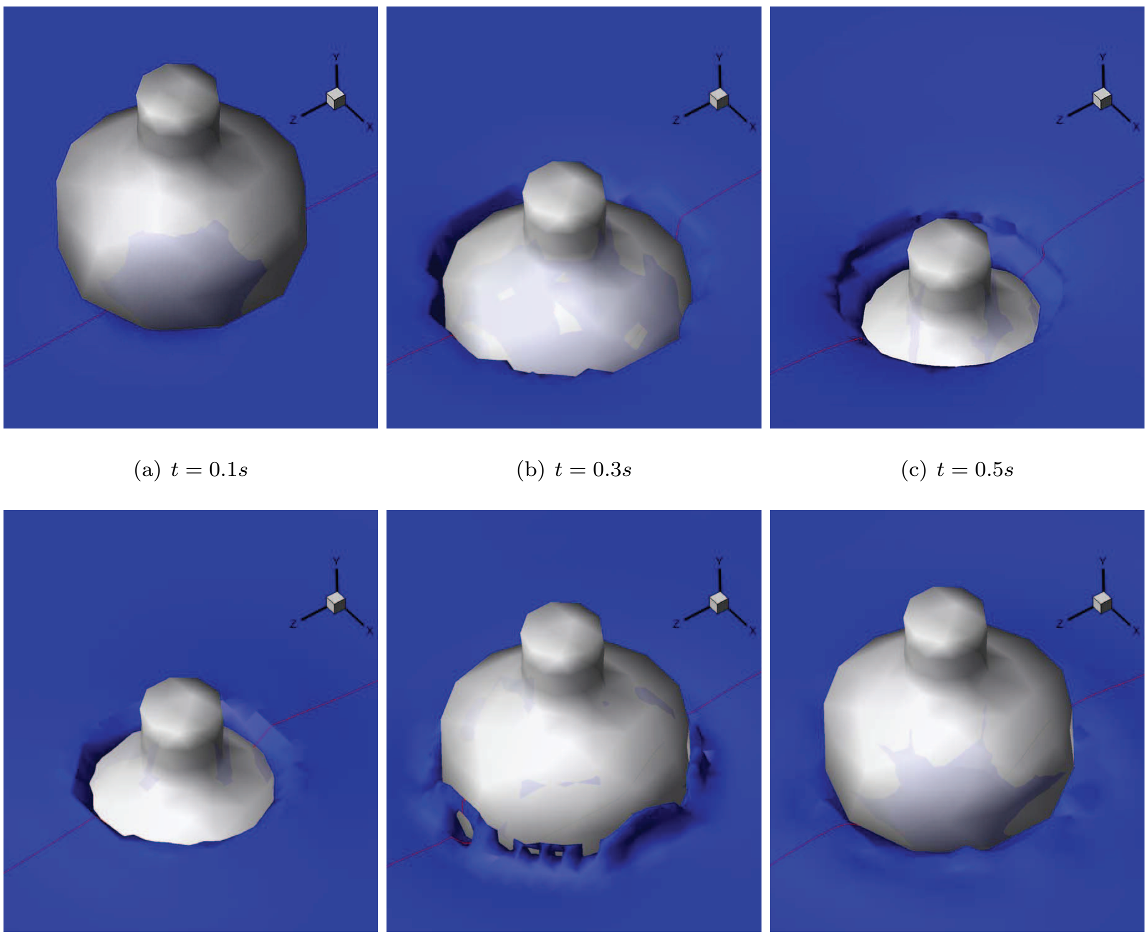


(a)



(b)

Figure 20: Time history of heave force (a) and vertical velocity (b) for the bobber undergoing heave motions.

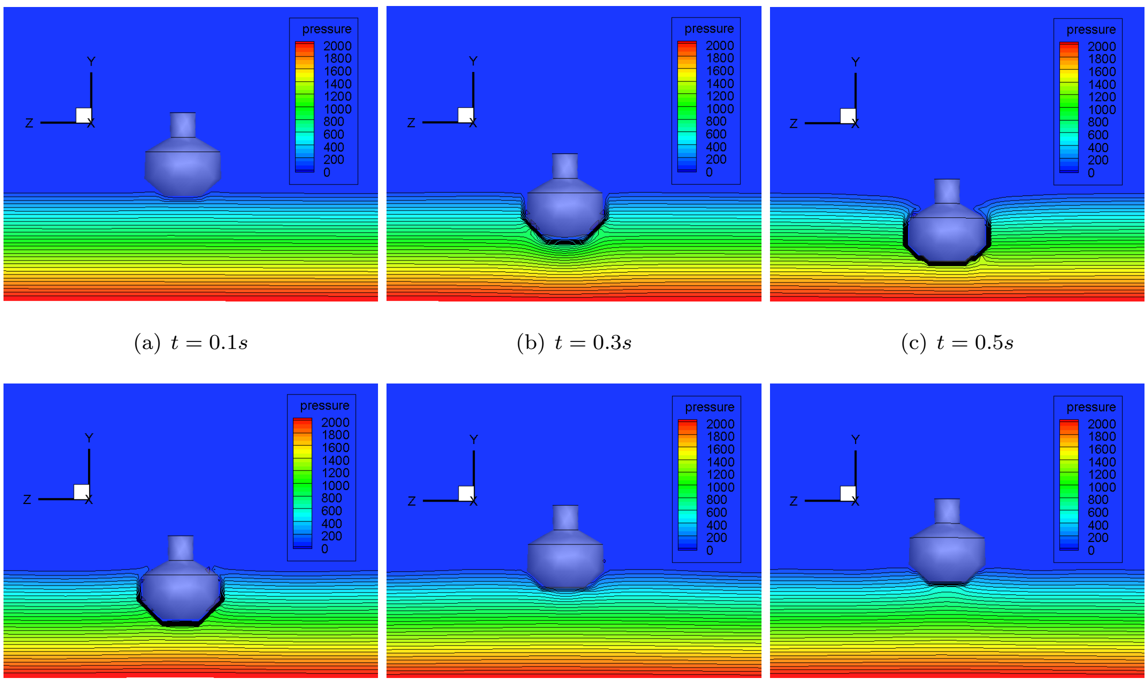


(a) $t = 0.1s$ (b) $t = 0.3s$ (c) $t = 0.5s$

(d) $t = 0.7s$ (e) $t = 0.9s$ (f) $t = 1.1s$

Figure 21: Water surface profiles at different time instants for the bobber decay.

domain. It can be seen that the hydrostatic pressure is dominant in most part of the domain. As the zero pressure is defined on the free surface, the free surface profile is represented by the zero-pressure contour approximately in the figure. Due to the geometry of the bobber, the splashing water jets are not very obvious during the water entry process. However, the water can run onto the shoulder of the bobber in Figs. 22(c) and 22(d). In Figs. 22(a), 22(e) and 22(f), the hydrostatic pressure field is not disturbed much by the bobber motion, while the dynamic pressure is visible around the bobber surface at $t = 0.3$s in Fig. 22(b).



(a) $t = 0.1s$ (b) $t = 0.3s$ (c) $t = 0.5s$

(d) $t = 0.7s$ (e) $t = 0.9s$ (f) $t = 1.1s$

Figure 22: Pressure distribution at different time instants for the bobber decay.

5. Conclusions

In this paper, a new algebraic algorithm is incorporated to the forcing point searching scheme in the 3D immersed boundary method, which differs from the traditional geometrical approaches. The conventional way employs the concept of the geometrical method to locate the forcing points. However, the algebraic algorithm adopted to determine the triangle-triangle intersection is robust, easy to implement and efficient. For the purpose of validation of the developed 3D immersed boundary method, three test cases are carried out, including dam break over an obstacle, free falling of a 3D wedge and free decay of a bobber. In the dam break case, the present immersed boundary method converges fast and can capture the pressure magnitude better than other numerical methods. For the water entry of the wedge, the present results reveal that the overall wedge displacement and velocity obtained by the present immersed boundary method are superior to other two immersed boundary methods. Lastly, after finding the most-fitted damping coefficient by matching the experimental data, for the free decay bobber the developed immersed boundary method can provide the better vertical displacement compared to other numerical method. All the numerical results suggest that the present numerical model is robust and accurate for both fixed and moving bodies with irregular complex geometries.

References

Archer, P. and Bai, W. (2015). A new non-overlapping concept to improve the hybrid particle level set method in multi-phase fluid flows, *Journal of Computational Physics* **282**: 317–333.

Berthelsen, P. A. and Faltinsen, O. M. (2008). A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries, *Journal of Computational Physics* **227**(9): 4354–4397.

Bihs, H. and Kamath, A. (2017). A combined level set/ghost cell immersed boundary representation for floating body simulations, *International Journal for Numerical Methods in Fluids* **83**(12): 905–916.

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

Calderer, A., Kang, S. and Sotiropoulos, F. (2014). Level set immersed boundary method for coupled simulation of air/water interaction with complex floating structures, *Journal of Computational Physics* **277**: 201–227.

Chen, H., Qian, L., Ma, Z. H., Bai, W., Li, Y., Causon, D. M. and Clive, C. G. (2019). Application of an overset mesh based numerical wave tank for modelling realistic free-surface hydrodynamic problems, *Ocean Engineering* **176**: 97–117.

Fadlun, E. A., Verzicco, R., Orlandi, P. and Mohd-Yusof, J. (2000). Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, *Journal of Computational Physics* **61**: 35–60.

Gilmanov, A. and Sotiropoulos, F. (2005). A hybrid cartesian/immersed boundary method for simulating flows with 3d, geometrically complex, moving bodies, *Journal of Computational Physics* **207**(2): 457–492.

Gu, H. B., Causon, D. M., Mingham, C. G. and Qian, L. (2013). Development of a free surface flow solver for the simulation of wave/body interactions, *European Journal of Mechanics - B/Fluids* **38**: 1–17.

Guigue, P. and Devillers, O. (2003). Fast and robust triangle-triangle overlap test using orientation predicates, *Journal of Graphics Tools* **8**(1): 25–32.

Held, M. (1997). ERIT - a collection of efficient and reliable intersection tests, *Journal of Graphics Tools* **2**(24): 25–44.

Hu, Z. Z., Causon, D. M., Mingham, C. G. and Qian, L. (2013). A cartesian cut cell free surface capturing method for 3d water impact problems, *International Journal for Numerical Methods in Fluids* **71**(10): 1238–1259.

Kim, D. and Choi, H. (2006). Immersed boundary method for flow around an arbitrarily moving body, *Journal of Computational Physics* **212**(2): 662–680.

Kim, J., Kim, D. and Choi, H. (2001). An immersed boundary finite-volume method for simulations of flow in complex geometries, *Journal of Computational Physics* **171**: 132–150.

Kleefsman, K. M. T., Fekken, G., Veldman, A. E. P., Iwanowski, B. and Buchner, B. (2005). A Volume-of-Fluid based simulation method for wave impact problems, *Journal of Computational Physics* **206**(1): 363–393.

Lee, J. and You, D. (2013). An implicit ghost-cell immersed boundary method for simulations of moving body problems with control of spurious force oscillations, *Journal of Computational Physics* **233**: 295–314.

Ma, Z. H., Qian, L., Ferrer, P. M., Causon, D. M., Clive, C. G. and Bai, W. (2018). An overset mesh based multiphase flow solver for water entry problems, *Computers and Fluids* **172**: 689–705.

Mittal, R., Dong, H., Bozkurttas, M., Najjar, F. M., Vargas, A. and von Loebbecke, A. (2008). A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, *Journal of Computational Physics* **227**(10): 4825–4852.

Mohd-Yusof, J. (1997). Combined immersed boundary/B-spline method for simulations of flows in complex geometries, *Technical report*, Center Annual Research Briefs, NASA Ames/Stanford University.

Moller, T. (1997). A fast triangle-triangle intersection test, *Journal of Graphics Tools* **2**(2): 25–44.

Nicolaou, L., Jung, S. Y. and Zaki, T. A. (2015). A robust direct-forcing immersed boundary method with enhanced stability for moving body problems in curvilinear coordinates, *Computers and Fluids* **119**: 101–114.

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

Roman, F., Napoli, E., Milici, B. and Armenio, V. (2009). An improved immersed boundary method for curvilinear grids, *Computers and Fluids* **38**(8): 1510–1527.

Schőberl, J. (1997). NETGEN: An advancing front 2D/3D-mesh generator based on abstract rules, *Computing and Visualization in Science* **1**(1): 41–52.

Seo, J. H. and Mittal, R. (2011). A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations, *Journal of Computational Physics* **230**(19): 7347–7363.

Thomas, S., Weller, S. and Stallard, T. (2008). Float response within an array: numerical and experimental comparison, *Proceeding of 2nd International Conference on Ocean Energy*, Brest, France.

Tropp, O., Tal, A. and Shimshoni, I. (2006). A fast triangle to triangle intersection test for collision detection, *Computer Animation and Virtual Worlds* **17**(5): 527–535.

Wu, C.-S. (2019). A modified volume-of-fluid/hybrid cartesian immersed boundary method for simulating free-surface undulation over moving topographies, *Computers and Fluids* **179**: 91–111.

Wu, C.-S. and Young, D.-L. (2014). Simulations of free-surface flows with an embedded object by a coupling partitioned approach, *Computers and Fluids* **89**: 66–77.

Wu, C.-S., Young, D.-L. and C.L, C. (2013). Simulation of wave-structure interaction by hybrid cartesian/immersed boundary and arbitrary lagrangian-eulerian finite-element method, *Journal of Computational Physics* **254**: 155–183.

Yan, B., Bai, W. and Quek, S. T. (2018). An improved immersed boundary method with new forcing point searching scheme for simulation of bodies in free surface flows, *Communications in Computational Physics* **24**(3): 830–859.

Yan, S. and Ma, Q. W. (2007). Numerical simulation of fully nonlinear interaction between steep waves and 2D floating bodies using the QALE-FEM method, *Journal of Computational Physics* **221**: 666 – 692.

Yang, J., Preidikman, S. and Balaras, E. (2008). A strongly coupled embedded-boundary method for fluid-structure interactions of elastically mounted rigid bodies, *Journal of Fluids and Structures* **24**: 167 – 182.

Yettou, E. M., Desrochers, A. and Champoux, Y. (2006). Experimental study on the water impact of a symmetrical wedge, *Fluid Dynamics Research* **38**(1): 47–66.

Zhang, Y., Zou, Q., Greaves, D., Reeve, D., Hunt-Raby, A., Graham, D., Phil, J. and Lv, X. (2010). A level set immersed boundary method for water entry and exit, *Communications in Computational Physics* **8**(2): 265–288.