Computational simulation of the interactions between moving rigid bodies and incompressible two-fluid flows

Amirmahdi Ghasemi, Ashish Pathak, Mehdi Raessi

Department of Mechanical Engineering, University of Massachusetts, Dartmouth, United States

Abstract

We present a two-dimensional computational flow solver for simulation of two-way interactions between moving rigid bodies and two-fluid flows. The fluids are assumed to be incompressible and immiscible. The two-step projection method along with Graphics Processing Unit (GPU) acceleration is employed to solve the flow equations. The fluid–solid interaction is captured by using the fictitious domain method. A consistent mass and momentum scheme is implemented, which allows for simulation of multiphase flows characterized by large density ratios. The evolution of interfaces in the three-phase system is tracked by using the volume-of-fluid method with two scalar functions, representing the solid domain and one of the fluids. A geometrical approach is employed to reconstruct the interfaces in cells containing three phases and capture the intersection of phase interfaces (triple point). The performance and accuracy of the flow solver are assessed through a set of canonical test cases. Then, it is used to simulate the interactions between a free-floating buoy and waves generated by a bottom-hinged paddle in a wave tank.

1. Introduction

Fluid–structure interaction is encountered in many important applications, including artificial heart valves, aircrafts, microfluidic devices, and ocean wave energy converters (WECs). The latter is of particular importance because ocean waves represent a vast untapped source of renewable energy that can contribute towards addressing the increasing demand for energy. However, compared to more mature wind and solar energy harnessing technologies, there is lack of convergence on the best method of energy extraction from ocean waves. This is because the entire process of bringing a new WEC technology from laboratory to its commercial deployment is very costly and time consuming. With rapid increase in computing power in recent years, computational tools have proven to be very successful in cutting down the scope of laboratory tests, especially in early design process of many industries. However, most of the available computational tools in the area of ocean wave energy are based on linearized hydrodynamic equations [1,2]. Hence, they are unable to capture non-linear effects such as wave breaking, turbulence and fluid–structure interactions that have significant impact on the performance of WECs [1,2]. Without capturing these effects the scope of a linear model in design and development of WECs is limited. In this paper, we present a computational tool that accurately predicts the interactions between moving rigid bodies and two-fluid flows by solving the full Navier–Stokes (N–S) equation. Although its immediate application is WECs, the computational tool was developed at a general level without any simplifications and by including all effects (e.g., surface tension) so that it is versatile and can be readily employed in other applications. Three important features of our computational tool are: (a) robustly handling multiphase flows of arbitrarily large density ratios, (b) capturing the interactions of two-fluid flows with moving rigid bodies, and (c) tracking three phase interfaces and resolving triple points: points at which three phase interfaces intersect. The reader is referred to [3–7] for a review on approaches to tracking the interfaces in two-phase flows and to [8] for a review on approaches to handling multiphase flows characterized by large density ratios. Next, we focus on features (b) and (c).

Unlike in approximate approaches to modeling fluid–structure interactions, where effects like viscosity or inertia are neglected, in the direct numerical simulation (DNS) approach, no simplification of such sort is made and the full N–S equations are solved. The DNS approach includes the body-conformal mesh, immersed boundary, and fictitious domain methods (see [9] for a detailed review). In the body-conformal mesh method, the computational domain encompasses the fluid region only and the mesh conforms to
the solid geometry. Unstructured grids have to be often used with
this method to accommodate geometrical complexities that may
arise, especially when the structure is highly deforming or dis-
placed and has a complicated geometry. Furthermore, the com-
putational domain is likely to require frequent remeshing because of
structure displacements, which can become challenging and com-
putationally expensive in many cases. These issues have been mit-
gated to a large extent in advanced body-conformal methods,
including (a) Deforming-Spatial Domain/Stabilized Space–Time
(DSD/SST) [10–12], (b) Arbitary Lagrangian–Eulerian (ALE) [13],
and (c) Fictitious Boundary [14] methods.

The most widely used methods for modeling fluid–structure
interactions are arguably the immersed boundary (IB) and the fic-
titious domain methods. In the IB method, which was introduced
by Peskin [15], a forcing term that represents the fluid–structure
interaction is added to the Navier–Stokes (N–S) equations, which
are solved on the entire computational domain. Based on how the forcing term
is implemented, the IB method can be subcategorized into direct
forcing and discrete forcing. In the discrete forcing approach [16], a
constitutive or model relation is used to calculate the forcing term,
which is then used in the discretized form of the N–S equations. In
the discrete forcing approach, the forcing term is included in the
system of discretized N–S equations as an extra condition applied
only near the immersed boundary. The discrete forcing approach
can be further divided into three classes: discrete direct forcing,
ghost cell and cut cell methods, detailed description of which can
be found in [17–20].

The main attribute of the fictitious domain (FD) method is that
the solid object is treated as a fictitious fluid and included in the
computational domain, on which the N–S equations are solved. The
velocity field in the solid domain, however, is constrained to
impose the solid rigidity condition. Glowinski et al. [21] introduced
distributed Lagrangian multiplier into the N–S equations to impose
the rigidity constraint on moving solid bodies. Many modifications
of the method proposed by Glowinski et al. [21] have been intro-
duced [22–24]. Sharma and Patankar [24] proposed a fast FD meth-
od, in which an intermediate velocity field is first obtained by
solving the N–S equations on the combined fluid–solid domain;
then, conserving the linear and angular momentum, the rigidity
constraint is imposed on the intermediate velocity field. The
computational cost of this method, which has been used in this study,
is very low. The efficacy of the FD method in capturing the interac-
tions of moving solid bodies with single-phase fluid flows has been
demonstrated in various studies, for example in [25–29]. Next, we
briefly review numerical studies that modeled the interaction of
solids with two-fluid flows.

The numerical methods used in previous studies to capture the
interactions between two-fluid flows and moving rigid bodies in-
clude the immersed boundary method (e.g., [30–32]), the cut-cell technique [33],
the vortex level set method [34], an Eulerian–Lagrangian method [35], a technique that treats the solid as a
highly viscous fluid [36], and a method that captures one-way
interaction of two-fluid flows and pre-animated bodies [37]. To
capture the evolution of fluid interfaces in these studies, the level
set method was used in [31,32,34,35,37] and the volume-of-fluid
method was employed in [30,33,36]. However, the three-phase
cells were only resolved in [33], where the cut-cell method was ex-
tended to resolve such cells in two dimensions. Resolving three-
phase cells is of importance, especially when consistent mass and
momentum transport schemes [38,8] are used to simulate multi-
phase flows characterized by large density ratios. Furthermore,
resolving such cells would provide a more versatile computational
tool that can be applied to a wide range of problems. Below, we
present a brief overview of the numerical approaches to recon-
structing interfaces in three-phase (three-material) cells in order
to track their evolution.

The first method to reconstruct interfaces in multi-material
cells was proposed by Noh and Woodward [39], which is known as
the “onion-skin” method and is in the context of the simple line
interface calculation (SLIC) algorithm. In this method, which was
reviewed in [40], the order of reconstruction is known a priori
and each interface can be reconstructed by using any technique
developed for two-material configurations. However, this method
is applicable to the cells that contain layered materials only and
cannot reconstruct interfaces that are intersecting in a cell. Choi
and Bussmann [41] presented an extension to the piecewise-linear
interface calculation (PLIC) scheme, in which interface reconstruc-
tion in three-material cells is performed by minimizing the error in
volume fractions. This method (which is used in this work) can be
applied to both layered materials and intersecting interfaces that
form a triple point in a three-material cell. The order of reconstruc-
tion, however, is known a priori in this two-dimensional (2D)
method. Caboussat et al. [42] also employed a minimization proce-
dure to reconstruct interfaces in three-material cells in 2D prob-
lems. The method can be applied to any configuration of three
interfaces in a cell; however, it is more suitable for Y-shaped
interface configurations than T-shaped ones. Schofield et al. [43]
proposed a material order-independent method that can reconstruc-
tissue interfaces of an arbitrary number of materials. In this method, to
find the relative position of materials in any cell, a particle attrac-
tion model is used, and then interfaces are reconstructed by using a
Voronoi diagram (or power diagram). Later in [44], Schofield et al.
increased the accuracy of the method to second-order, where in-
stead of using a particle attraction model, they employed a piece-
wise linear approximation to the scalar functions representing the
materials to infer the relative locations of the materials in any cell.
Furthermore, a smoothing procedure was applied to the interfaces
in multi-material cells to achieve interface reconstruction with
second-order accuracy. Ahn and Shashkov [45] proposed 3D inter-
face reconstruction methods for multi-material (more than two)
problems. In their methods, the multi-material cells are divided
into sub-cells that contain only one material. They presented
methodologies applicable in the PLIC algorithm as well as the mo-

In this paper, a 2D computational tool that simulates the inter-
action of two-fluid flows with moving rigid bodies is presented.
The two-step projection method in the finite-volume context with
GPU acceleration is used to solve the flow equations. The evolution
of fluid interfaces is tracked by using the volume-of-fluid method.
Furthermore, by using a consistent mass and momentum advection
scheme [38], we achieve a robust numerical solver for interfacial
flows characterized by large density ratios (1000 or higher). The
fictitious domain method [24] is used to capture the interaction of
a moving rigid solid object with two-fluid flows. A geometrical
method [41] is used to reconstruct the interfaces in three-phase
cells and resolve the triple points in order to accurately track phase
interfaces and transport momentum. A set of canonical test cases
are presented to demonstrate the accuracy of the computational
tool. Finally, simulations of waves generated in a tank and their
interaction with a free-floating rigid body are presented.

2. Governing equations

Consider a flow of two incompressible, immiscible and Newto-
nian fluids interacting with a moving rigid solid body. The flow is
governed by the continuity and momentum Eqs. (1) and (2):

$$\nabla \cdot \tilde{V} = 0
$$

(1)

$$\frac{\partial (\rho \tilde{V})}{\partial t} + \nabla \cdot (\rho \tilde{V} \tilde{V}) = -\nabla p + \nabla \cdot \tau + \tilde{F}_b + \tilde{F}_{st} + \tilde{F}_i,
$$

(2)
Here, \( \mathbf{V} \) denotes the velocity vector, \( \rho \) the density, \( p \) the pressure, \( \mathbf{F}_s \) represents any body forces, like gravity, and \( \mathbf{F}_{ST} \) is the surface tension force. \( \mathbf{F}_r \), represents the fluid–solid interaction force, which will be described later. Furthermore, \( \tau \) is the stress tensor, defined as,

\[
\tau = \mu \left[ (\nabla \mathbf{V}) + (\nabla \mathbf{V})^T \right]
\]

where \( \mu \) is the dynamic viscosity.

For a system consisting of three phases, e.g., liquid, solid, and gas, two additional equations are required for tracking the liquid and solid interfaces. Following the approach used in the volume-of-fluid (VOF) method, we define two scalar functions \( f \) and \( \psi \) to represent the liquid and solid phases, respectively:

\[
f(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \text{liquid} \\ 0, & \mathbf{x} \notin \text{liquid} \end{cases}
\]

and

\[
\psi(\mathbf{x}) = \begin{cases} 1, & \mathbf{x} \in \text{solid} \\ 0, & \mathbf{x} \notin \text{solid} \end{cases}
\]

Then, the liquid and solid phases are tracked by solving the following transport equations:

\[
\frac{\partial f}{\partial t} + (\mathbf{V} \cdot \nabla)f = 0
\]

\[
\frac{\partial \psi}{\partial t} + (\mathbf{V} \cdot \nabla)\psi = 0
\]

As it is presented in the next section, a single set of Eqs. (1) and (2) is solved in the entire domain, where the properties \( \rho \) and \( \mu \) are determined by using \( f \) and \( \psi \) as:

\[
\rho = f \rho_l + (1 - f) \rho_s
\]

\[
\mu = f \mu_l + (1 - f) \mu_s
\]

where Eq. (9) is applicable on the fluid portion of the domain only.

3. Numerical method

To solve the flow equations, the two-step projection method [47] was used, in which Eq. (2) is split into predictor and corrector steps:

\[
\frac{\rho^{n+1} \mathbf{V}^{n+1} - \rho^n \mathbf{V}^n}{\Delta t} = - \nabla \cdot (\rho \mathbf{V}^n) + \nabla \cdot (\mu (\nabla \mathbf{V} + (\nabla \mathbf{V})^T)) + \mathbf{F}_g + \mathbf{F}_s
\]

\[
\frac{\rho^{n+1} \mathbf{V}^{n+1} - \rho^{n+1} \mathbf{V}^n}{\Delta t} = - \nabla \rho^{n+1} + \mathbf{F}_{ST}^{n+1}
\]

where the superscripts \( n \) and \( n+1 \) denote the current and next time steps, respectively, and \( \mathbf{V}^{n+1} \) represents an intermediate velocity. By applying the continuity Eq. (1) on Eq. (11) the Poisson equation is obtained:

\[
-\frac{1}{\rho^{n+1}} \nabla^2 \rho^{n+1} = - \frac{1}{\Delta t} \nabla \cdot \mathbf{V}^n - \frac{1}{\rho^{n+1}} \nabla \cdot \mathbf{F}_{ST}^{n+1}
\]

\( \rho^{n+1} \) is obtained by solving Eq. (12) iteratively using an incomplete Cholesky Conjugate Gradient (ICCG) solver. \( \mathbf{V}^{n+1} \) is then evaluated using Eq. (11).

Solving the above Poisson equation on a CPU is computationally expensive and often the bottleneck of the solution procedure, especially in serial flow solvers. To remove this bottleneck, we employ a GPU-accelerated Poisson solver, developed by Codyer et al. [48]. The performance and speedup of the above GPU-accelerated solver was tested and well established in [48]. For example, it was shown that using the GPU-accelerated solver the total runtime in a 2D simulation with 4.2 million grid points is accelerated 81 times, compared to running the same simulation in serial on a CPU [48]. The above comparison was performed on a compute node with two Intel Xeon (quad-core) E5620 2.4 GHz processors, 24 GB of DDR3 ECC 1333 MHz RAM, and a single, ECC enabled, NVIDIA Tesla M2050 GPU that has 3 GB of on-board memory [48].

3.1. Fluid–solid interaction

To capture the interaction of a moving rigid solid body with a two-fluid flow, we implemented the fast fictitious domain method, developed by Sharma and Patankar [24], into our flow solver. In this method, the solid phase is first treated as a fictitious fluid, and the governing equations for the fluid flow are solved in the entire computational domain, including in the solid phase. The velocity proposed by the flow solver for the solid phase, however, does not necessarily satisfy the solid rigidity condition. To enforce the rigidity condition, first, the proposed solid velocity is averaged over the solid domain to obtain the translational and rotational components, \( \mathbf{V}_s \) and \( \omega_s \), respectively:

\[
M_s \mathbf{V}_s = \int_{\text{solid body}} \rho \mathbf{V} \, d\mathbf{v}
\]

\[
I_s \omega_s = \int_{\text{solid body}} (\mathbf{r} \times \rho \mathbf{V}) \, d\mathbf{v}
\]

where \( M_s \) and \( I_s \) are the mass and moment of inertia, respectively, of the solid object, \( \mathbf{r} \) is the volume and \( \mathbf{r} \) denotes the position vector relative to the center of mass of the solid object. Then, the averaged (corrected) solid velocity field, \( \mathbf{V}_s^{\text{solid body}} \), is obtained by adding the two components:

\[
\mathbf{V}_s^{\text{solid body}} = \mathbf{V}_s + \omega_s \times \mathbf{r}
\]

Note that \( \mathbf{V}_s^{\text{solid body}} \) satisfies the rigidity condition and may be different from the velocity field initially proposed by the flow solver for the solid phase. Thus, using the following equation, the fluid–solid interaction force, \( \mathbf{F}_s \) (to be used in Eq. (2)), is computed at each point on the solid body,

\[
\mathbf{F}_{ST}^{n+1} = \frac{\rho_s}{M_s} (\mathbf{V}_s^{n+1} - \mathbf{V}_s^{n})
\]

where \( \mathbf{V}_s^{n+1} \) and \( \mathbf{V}_s^{n+1} \) are the velocity of each point on the solid body before and after averaging.

3.2. Mass and momentum advection

At the beginning of each time step, mass and momentum of all three phases are advected simultaneously using the consistent mass and momentum advection scheme proposed by Bussmann et al. [38]. This scheme establishes a tight coupling between mass and momentum fluxes and provides a robust numerical solver for multiphase flows characterized by large density ratios (1000 or higher). To advect mass (volume) of the phases represented by scalar functions \( f \) or \( \psi \), the mass-conserving VOF method of Youngs [49] is used. This method uses a piecewise-linear interface calculation (PLIC) algorithm, where the interfaces in two-phase cells are reconstructed by using the volume fraction and the normal vector to the interface, which are obtained from \( f \) or \( \psi \). Then, calculating the mass flux, mass and momentum are advected. For consistency, we use the same flux density for both mass and momentum transport, and thereby establish a tight coupling between them [38]. To reconstruct the phase interfaces in cells containing three phases, we follow an extension of the above method, which is described next.
3.3. Treatment of three-phase cells

We employed the geometrical technique of Choi and Bussmann [41] to reconstruct the phase interfaces in three-phase cells and to resolve the triple points: points at which all three phase interfaces intersect. Consider the example shown in Fig. 1(a), where the liquid–gas interface is intersecting with the solid surface and a triple point is formed. Fig. 1(b) shows the magnified view of a 3 × 3 stencil S around a three-phase cell C. Four types of cells can exist in such three-phase systems: (1) cells containing a triple point, (2) cells containing three phases but no triple point, (3) cells containing only two phases, and (4) cells containing only one phase. Type 4 cells do not require reconstruction, and in type 3 cells, the Youngs’ method [49], which was previously discussed, can be readily used to reconstruct the phase interfaces. The geometrical technique of Choi and Bussmann [41] is used to reconstruct phase interfaces in cells of type 1 and 2 by minimizing the error in volume fractions. A brief description of the technique is presented next.

Consider again the three-phase system shown in Fig. 1 and a three-phase cell C depicted in Fig. 1(b). The order of interface reconstruction in cell C is known a priori: the phase interfaces are obtained by first reconstructing the solid surface and then the liquid–gas interface. The order is such because if the phase interfaces intersect in a numerical cell, it is always the liquid–gas interface that will be truncated by the solid surface (note that the solid phase is rigid but moving). The solid surface is reconstructed in cell C on the liquid–gas side; see polygon P shown in red in Fig. 1(b). One side of the polygon P is formed by the reconstructed solid surface line, and its other sides coincide with the boundaries of cell C. In the next step, the line representing the solid surface is extended in both directions, which is denoted by line L in Fig. 1(b). Lines like M that are originating from a point z located on line L and intersecting the polygon P are considered as candidates for the liquid–gas interface in cell C. For a given z, the liquid volume fraction in cell C is used to find the orientation of line M; thus, the orientation of line M is fixed for a given point z. The position of point z itself can vary and its precise location, that corresponds to the true liquid–gas interface location inside cell C, can be found by minimizing the following volume fraction error on a 3 × 3 stencil about cell C:

\[
g(z) = \sum_{ij} \left[ (f_{ij} - f^i_j)^2 + (\alpha_{ij} - \alpha^i_j)^2 \right]
\]

where \(f_{ij}\) and \(\alpha_{ij}\) are the known liquid and gas volume fractions, respectively, and \(f^i_j\) and \(\alpha^i_j\) are the predicted volume fractions. Note that \(\alpha_j = 1 - f_j - \alpha_j\). To minimize the error \(g(z)\) and find the corresponding point \(z\) the Golden Section Search method [50] was used. Following [41], the position of \(z\) ranges nine cell widths on either side of cell C. Since the liquid can lie on either sides of line L, two minima are found, and the absolute minimum of the two is used to ascertain the location of point \(z\) and the orientation of the liquid–gas interface in cell C. The above technique is capable of resolving triple points as well, in which case point \(z\) lies in cell C.

We have used various geometric constructs similar to [51] for determining the position of the liquid–gas interface. Some of the important construct features are briefly described in the Appendix. Although the process of reconstructing interfaces in three-phase cells is more computationally expensive than that in two-phase cells, only a small fraction of cells would contain three phases. Thus, resolving the three-phase cells and triple points does not pose a bottleneck in the overall advection scheme. It should be noted that if one wants to impose a contact angle at the intersection of phase interfaces, then the above error minimization process is not needed in the cell that contains the triple point; the liquid–gas interface can be reconstructed in a straightforward procedure by simply using the liquid volume fraction and the orientation of the liquid–gas interface, dictated by the contact angle. In this work, we did not impose a contact angle.

3.4. Surface tension

Although in certain category of flow problems, for example, ocean wave energy conversion, surface tension effects play a negligible role, incorporating the surface tension force in a flow solver would provide a more versatile computational tool. Here, the surface tension force, \( \bar{F}_{ST} \), is incorporated into Eq. (11) by using the balanced-force algorithm of Francois et al. [52], which imposes an exact balance between the surface tension and pressure forces. Applied only along the fluid interfaces, the surface tension force is formulated as:

\[
\bar{F}_{ST} = \sigma \kappa \nabla \mathbf{f}
\]

(18)

where \(\sigma\) is the coefficient of surface tension between the fluids and \(\kappa\) is the fluid interface curvature, which is calculated via

\[
\kappa = -\nabla \cdot \hat{n}
\]

(19)

and

\[
\hat{n} = \frac{\nabla \mathbf{f}}{||\nabla \mathbf{f}||}
\]

(20)

![Fig. 1. (a) Stencil S containing a triple point in a three-phase system; and (b) geometric constructs to calculate error function g(z) in a 3 × 3 stencil S.](image-url)
4. Results

This section presents a set of test cases to assess the accuracy and performance of the computational tool in problems with three phases. Note that the accuracy of the solver in simulating two-fluid flows (i.e., no fluid–solid interactions) was previously investigated and demonstrated in [53,54]. We begin with test cases with a prescribed velocity field to evaluate the accuracy of resolving the triple point and tracking three phase interfaces. Then, the accuracy of modeling the fluid–solid interaction is assessed. Finally, application of our numerical model in wave energy conversion is presented.

4.1. Translation of a T-shape phase interface

This test assesses the accuracy of resolving three-phase cells, capturing the triple point, and tracking three interfaces. A T-shape interface separating three phases (materials) is initialized in a 1 \times 1 domain, as shown in Fig. 2(a). The triple point is initially located at (0.5, 0.5). In Fig. 2(a), the yellow part (right half) is represented by scalar function \( \varphi \) (see Eq. (5)) and is referred to as “solid” here, while the blue part (lower left quarter) is represented by scalar function \( f \) (see Eq. (4)) and is referred to as “liquid”. The prescribed velocity field is \( (1, -1) \) and the final position of the triple point is \((0.8,0.2)\). The total volume fraction error in each cell \((i,j)\), denoted by \( e_{ij} \), is computed at the end of the translation using

\[
e_{ij} = \left| f_{ij}^{\text{exact}} - f_{ij}^{\text{pred}} \right| + \left| \varphi_{ij}^{\text{exact}} - \varphi_{ij}^{\text{pred}} \right|
\]

(21)

where \( f_{ij}^{\text{exact}} \) and \( \varphi_{ij}^{\text{exact}} \) are the exact liquid and solid volume fractions, respectively, and \( f_{ij}^{\text{pred}} \) and \( \varphi_{ij}^{\text{pred}} \) are the predicted volume fractions at the end of the translation.

Table 1 reports the \( l_1, l_3 \) and \( l_{\infty} \) norms of \( e_{ij} \) at different mesh resolutions. The order of convergence in \( l_1 \) and \( l_3 \) error norms is higher than first order, while the \( l_{\infty} \) norm remains unchanged with mesh refinement. Upon further investigation on the latter, it was found that the error in volume fractions actually originated and propagated onward from a cell, labeled N in Fig. 3, that is located immediately left of the three-phase cell T, shown in Fig. 3. The interface reconstruction in cell N, which contains two only phases, is accomplished by using the \( f \) field, relying on the volume fraction in cell N, which is denoted by \( f_i \), and the unit normal vector \( \hat{n} \) obtained from \( \nabla f \) (see Eq. (20)). As shown in Fig. 3, the volume fraction of cells to the left of cell N is also equal to \( f_i \); but the (liquid) volume fraction is reduced to \( f_i - \Delta \) in cell T, the right neighbor of cell N, because the solid phase truncates a liquid area of size \( \Delta \) (the red square shown in Fig. 3). Thus, using a central scheme to compute \( \nabla f \) in Eq. (20) in order to obtain \( \hat{n} \) yields vector \((0, -1)\) everywhere along the liquid surface, except in cell N (note that the orientation of the liquid interface in cell T is obtained through error minimization, not from \( \nabla f \); see Section 3.3). The error in \( n \) in cell N leads to errors in interface reconstruction and liquid volume advection in cell N, which propagates onward.

The above issue can be resolved by using a smaller stencil for \( \nabla f \) calculation in cell N, or by using auxiliary information, for example, a virtual \( f \) field that extends into the solid phase, which is used only in \( \nabla f \) calculation in cell N. The virtual \( f \) field can be obtained by an extension of the reconstructed liquid interface into the solid phase. We tried the latter in a repeat test, which only affected the interface normal in cell N and ensured that it is vector \((0, -1)\). The volume fraction errors in the repeat test are presented in Table 2. It is seen that by using the above auxiliary information the convergence rate for all three error norms is higher than first order.

4.2. Rotation of a two-phase circle

This test, which is borrowed from [41], assesses the accuracy of resolving two triple points in rotation. Two halves of a circle are initialized as two different phases, while a third phase surrounds the circle (see Fig. 4(a)). Here, the surrounding phase is represented by \( \varphi \) and its interface is reconstructed first, while the blue region (upper half) of the circle is represented by \( f \). The circle, the diameter of which is 0.15, is initially located at \((0.25,0.5)\) in a \( 1 \times 1 \) domain as shown in Fig. 4(a). The circle makes one full rotation around the point \((0.5,0.5)\). Using Eq. (21), the errors associated with the volume fractions are computed at the end of the rotation.
Table 2
The errors associated with the volume fractions in translation of a T-shape interface when the auxiliary information is used near the triple point.

<table>
<thead>
<tr>
<th>Δx</th>
<th>$I_1$</th>
<th>Order</th>
<th>$I_2$</th>
<th>Order</th>
<th>$I_3$</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.00 × 10^{-3}</td>
<td>0.050</td>
<td>2.32</td>
<td>1.63 × 10^{-3}</td>
<td>3.73</td>
<td>9.25 × 10^{-4}</td>
<td>3.08</td>
</tr>
<tr>
<td>2.50 × 10^{-3}</td>
<td>0.010</td>
<td>1.22 × 10^{-6}</td>
<td>1.09 × 10^{-4}</td>
<td>1.63</td>
<td>3.87 × 10^{-5}</td>
<td>1.49</td>
</tr>
<tr>
<td>1.25 × 10^{-3}</td>
<td>0.004</td>
<td>3.93 × 10^{-7}</td>
<td>3.87 × 10^{-5}</td>
<td>1.49</td>
<td>3.87 × 10^{-5}</td>
<td>1.49</td>
</tr>
</tbody>
</table>

Table 3 presents the $I_1$, $I_2$, and $I_3$ norms of the volume fraction error at different grid sizes. Again, the $I_1$ and $I_2$ error norms converge with first-order accuracy. Similar to what was seen in the previous test, the $I_3$ error norm remains almost constant with mesh refinement, which is again because the $f$ field is truncated near the triple point, impacting the calculation of interface normals near that point.

4.3. Free fall of a cylinder in a 2D fluid container

This test case is adopted from the work by Glowinski et al. [21]. A solid cylinder with a diameter of 2.5 mm is positioned 4 cm above the bottom surface of a 2 cm × 6 cm fluid container, along the center line, as shown in Fig. 5. The fluid density is 1000 kg/m³ and the ratio of solid and fluid densities is 1.5. The fluid dynamic viscosity is 10^{-3} Pa s. The cylinder is released at $t = 0$ and free-falls due to gravity ($g = 9.81$ m/s²). Fig. 6(a) and (b) illustrate the vertical position and velocity, respectively, of the cylinder versus time, predicted by a simulation performed with 64 grid points across the cylinder diameter. The figures also include simulation data reported by Glowinski et al. [21]; a very good agreement is seen between the two data sets. The cylinder seems to reach a terminal velocity around 0.81 m/s, when it is 1 cm above the bottom surface.

Using four different mesh resolutions, we also performed a grid convergence study on this test case. The cylinder’s position and velocity obtained at each resolution are presented in Fig. 7(a) and (b), respectively. The figures show convergence in both quantities. Note that, as it was shown above, the results obtained with 64 cells per diameter (CPD = 64) match the results reported by Glowinski et al. [21]. Taking the results obtained at the highest mesh resolution as the benchmark solution, the errors in the cylinder position are presented in Table 4 for each mesh resolution. It is seen that the convergence rate is close to second-order for all three error norms.

4.4. Free fall of a cylinder in a column containing two fluids

Next, we extend the previous test case to include two fluids: the column, which is now 12 cm tall, contains a fluid with a density of 1000 kg/m³ (Fluid 1) resting on top of a denser fluid (Fluid 2) as shown in Fig. 8(a). The density ratio of Fluid 2 to 1 is 1.25. A solid cylinder with a diameter of $D = 2.5$ mm is initially positioned 4 cm above the fluid interface. The density ratio of the solid to Fluid 1 is 1.5. The dynamic viscosity is $10^{-3}$ Pa s in both fluids. A dimensionless time, denoted by $\tau$, is defined as $\tau = t\sqrt{g/D}$, where $t$ is the dimensional time and $g = 9.81$ m/s². Fig. 8 shows snapshots of a simulation performed with 64 cells per diameter of the cylinder. The cylinder, released at $\tau = 0$, is about to impact the fluid interface at $\tau \approx 22$. As seen in Fig. 8(b), the interface deforms due to the pressure induced by the free-falling cylinder. Figs. 8(c) and (d) show a cavity being formed in the wake of the cylinder as it penetrates into Fluid 2. Fig. 8(e) shows a jet that begins to form as the cavity collapses. The jet shoots up above the initial position of the interface as seen in Fig. 8(f).

In addition, we carried out a grid convergence study on this test case, the results of which are shown in Fig. 9. Fig. 9(a) shows the

---

**Table 3**
The errors associated with the volume fractions in rotation of a two-phase circle in a third phase.

<table>
<thead>
<tr>
<th>Δx</th>
<th>$I_1$</th>
<th>Order</th>
<th>$I_2$</th>
<th>Order</th>
<th>$I_3$</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00 × 10^{-3}</td>
<td>0.825</td>
<td>3.76 × 10^{-2}</td>
<td>1.05 × 10^{-2}</td>
<td>1.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.00 × 10^{-3}</td>
<td>0.842</td>
<td>1.55 × 10^{-2}</td>
<td>3.71 × 10^{-3}</td>
<td>1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.50 × 10^{-3}</td>
<td>0.859</td>
<td>8.30 × 10^{-3}</td>
<td>1.72 × 10^{-3}</td>
<td>1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.25 × 10^{-3}</td>
<td>0.970</td>
<td>4.71 × 10^{-3}</td>
<td>7.89 × 10^{-4}</td>
<td>1.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
vertical position of the cylinder in dimensionless time, and Fig. 9(b) depicts its velocity predicted by simulations performed at various number of cells per diameter of the cylinder. The results exhibit convergence at CPD = 64. As seen in Fig. 9(b), the cylinder continuously accelerates while free-falling in the lighter fluid without reaching a terminal velocity, but it attains a terminal velocity in the denser fluid at $\frac{s}{C^2} = 25$. Furthermore, it is clearly seen how the velocity and position curves are affected at the moment when the cylinder enters Fluid 2.

It should be noted that the above 2D problem is more of a numerical test case than a physical one, because the actual physical case involves features such as jet breakup and droplet formation that are inherently 3D, which will not be fully captured by a 2D simulation. An example of such a 3D study is the work by Do-Quang and Amberg [55], in which the impact of a 3D solid sphere on a liquid pool was simulated and the effects of surface wetting on liquid splashing was investigated.

### 4.5. Transient heave oscillation of a floating cylinder

In this test case, which is based on Ito’s experimental work [56], a cylinder with a diameter of $D = 15.24$ cm (i.e., 6 in.) is released from 2.54 cm above its equilibrium position in a water pool and begins to perform heave oscillations (oscillatory vertical displacements). The depth of the water is 1.2 m and the motion of the cylinder is restricted to pure heave. The length of the tank is 25 m, which is sufficiently large to avoid any wave reflection off the walls. The density of the cylinder is 500 kg/m$^3$.

Fig. 10 shows the dimensionless vertical position of the cylinder, $\frac{y}{y_0}$, versus dimensionless time, $t\sqrt{g/D}$, where $y_0 = 2.54$ cm. As seen in Fig. 10, the simulation results converge to the experimental data at the resolution of 64 cells per diameter. Using the
experimental data as the benchmark, the $l_1$, $l_2$ and $l_\infty$ norms of the error in the heave position of the cylinder were calculated, which are presented in Table 5. The convergence rate is about first-order for $l_1$ and $l_2$ while the $l_1$ error norm does not converge as fast. Maximum error was observed in regions where the curve slope is very steep.

4.6. Roll motion of a floating rectangular block

This test case is based on the experiments conducted by Jung et al. \[57\]. A rectangular block with length $L = 0.3$ m and height $H = 0.1$ m is half-submerged into a 0.9 m deep water tank and is constrained to only rotate around its center of mass (roll). The density of the block is 1180 kg/m$^3$, and the tank is 10 m long. The block is initially tilted at an angle of 15°, as shown in Fig. 11, and then released to perform a roll motion.

Fig. 12 shows the roll angle (angle of rotation) in time, reported from the experimental data \[57\] and predicted by simulations performed at various number of cells per diameter (CPD) of the block. It is seen that the simulation results converge at 60 CPL. The period of oscillations is predicted fairly well by the simulations, especially at the early stage of oscillations; but the amplitude of oscillations decays faster in the experiments, which is possibly because the turbulent viscous dissipations and the frictional losses in the 3D experiments are not fully captured in the 2D simulations.

4.7. Wave maker

Next, we focus on the performance of our computational tool in simulations of wave energy conversion devices. The first step is to ensure that the wave generation mechanisms are simulated accurately. Two types of wave makers are presented: (1) piston and (2) bottom-hinged paddle.

4.7.1. Piston wave maker

The characteristics of the waves generated by a piston wave maker were reported by Ursell et al. \[58\], which will be used here to validate the wave generation mechanism. The numerical
domain used in our simulation is $15 \text{m} \times 0.6 \text{m}$. The water depth is denoted by $d$. The width and height of the piston wave maker are $0.1 \text{m}$ and $0.5 \text{m}$, respectively. The initial position of the wave maker is $0.5 \text{m}$ from the left wall as shown in Fig. 13(a). The piston is $0.1 \text{m}$ and $0.5 \text{m}$, respectively. The initial position of the wave maker is denoted by $\psi$; however, the fluid–solid interaction in this problem is “one-way” in that the piston velocity, $V$, is prescribed as

$$V = \frac{S}{T} \left[\frac{2\pi}{T} \sin \left(\frac{2\pi}{T} t\right)\right]$$

where $T$ is time period, and $S$ is the piston stroke. A $5 \text{m}$ long damping zone is on the opposite end of the tank to avoid wave reflection. The fluid viscosity in the damping zone is increased to $10^8$ times of water viscosity. The size of each cell in the uniformly spaced numerical grid is $0.01 \text{m} \times 0.01 \text{m}$.

We simulated three different waves by varying $T, S$ and $d$ as listed in Table 6. Denoting the average wave height and wavelength by $H$ and $L$, respectively, Table 6 presents a comparison between $H/S$ and $2\pi d/L$ predicted by the simulations and the theoretical and experimental values reported by Ursell et al. [58]. The theoretical $H/S$ is from an inviscid flow analysis reported in [58]:

$$\frac{H}{S} = \frac{2\cosh(2k_d d) - 1}{\sinh(2k_d d) + 2k_d d}$$

where $k = 2\pi/L$. As seen in Table 6, in all three cases, the values predicted by the simulations for $H/S$ and $2\pi d/L$ agree very well with the theoretical and experimental values.

Furthermore, to assess the performance of the damping zone in suppressing wave reflection off the tank wall, the damping zone coefficient, $\epsilon$, was computed by using Eq. (24) from [58] in all simulations,

$$\epsilon = \frac{(H_{\text{Max}} - H_{\text{Min}})}{(H_{\text{Max}} + H_{\text{Min}})}$$

where $H_{\text{Max}}$ and $H_{\text{Min}}$ are the maximum and minimum wave heights, respectively. According to Ursell et al. [58], as long as $\epsilon$, remains below 10%, the reflection can be considered negligible because the reflected wave is smaller than 1% of the primary wave. As seen in Table 6, $\epsilon$, is indeed below 10% in all of our simulations, indicating that the damping zone is effective in suppressing wave reflections. Note that the effectiveness of the damping zone was verified in all wave tank simulations, including those presented in the following sections, in a similar manner; however, for brevity the damping coefficient is not reported for those.

Fig. 13 illustrates the development of waves and the wave profiles at various instances of time. The figure corresponds to the parameters listed on the last row of Table 6. The motion of the piston is not visible because the piston stroke is much smaller than the tank length.

### 4.7.2. Bottom-hinged paddle wave maker

Next, a paddle wave maker of $0.5 \text{m}$ height and $0.1 \text{m}$ width is hinged at the bottom in a $0.3 \text{m}$ deep and $15 \text{m}$ long water tank as shown in Fig. 14. The tank includes a $5 \text{m}$ long damping zone on the right end. The angular velocity of the paddle is prescribed as

$$\dot{\theta} = \frac{\Delta \phi}{2} \left[\frac{2\pi}{T} \sin \left(\frac{2\pi}{T} t\right)\right]$$
where \( \Delta \theta \) is the angle that the paddle sweeps out during its motion. Denoting the water depth by \( d \), the stroke, \( S \), is

\[
S = 2d \tan \left( \frac{\Delta \theta}{2} \right)
\]

We consider a case, where \( S = 8.45 \text{ cm} \) and \( T = 0.84 \text{ s} \). This case was previously studied by Finnegan and Goggins [59] and Anbarsooz et al. [60] through numerical simulations. Experimental measurements of this case are also reported in [60]. Additionally, the
theoretical analysis by Ursell et al. [58], in which the flow was assumed inviscid, predicts the ratio of the wave height to the wave maker stroke to be

$$
\frac{H}{S} = \frac{4 \sinh(\kappa d)}{k \sinh(\kappa d) + \cosh(\kappa d) + 1}\left(\frac{k \sinh(\kappa d) + \cosh(\kappa d) + 1}{k \sinh(2\kappa d) + 2k \cosh(2\kappa d)}\right)
$$

(27)

We simulated this case by using a uniform 0.01 m × 0.01 m mesh resolution, and the results are presented in Fig. 14. The wave generation process reaches steady state at around 12 s, and no wave reflection is observed. Furthermore, Table 7 presents \(H/S\) and \(2\pi d/L\) obtained from our numerical simulation, which are compared to the numerical values reported by Finnegan and Goggins [59] as well as the theoretical [58] and experimental [60] values. Our results agree very well with the numerical and experimental data. As discussed in [60], the discrepancy between the theoretical value and the experimental and/or numerical values is because the

<table>
<thead>
<tr>
<th>(T) (s)</th>
<th>(S) (cm)</th>
<th>(d) (m)</th>
<th>(\frac{H}{S})</th>
<th>(\frac{H}{S}_{\text{Num.}})</th>
<th>(\frac{H}{S}_{\text{Exp.}})</th>
<th>(\frac{H}{S}_{\text{Theory}})</th>
<th>(2\pi d/L)</th>
<th>(2\pi d/L_{\text{Num.}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.84</td>
<td>8.45</td>
<td>0.30</td>
<td>0.77</td>
<td>0.76</td>
<td>0.80</td>
<td>0.96</td>
<td>1.80</td>
<td>1.80</td>
</tr>
</tbody>
</table>

Fig. 15. Interaction of a free-floating buoy with waves generated by a bottom-hinged paddle wave maker.

Fig. 16. Surge (frame a) and heave (b) motion of the free-floating buoy in a wave tank, shown in Fig. 15.
error in the theoretically calculated wavelength increases for waves with a high steepness, like in this case.

4.8. Interaction of a free-floating cylindrical buoy with waves

After assessing the performance of our computational tool in simulating wave makers, we now test the interaction of a cylindrical buoy with water waves. Here, we use the bottom-hinged paddle wave maker of Section 4.7.2 with the same stroke and period. The geometry of the tank, the water depth, and the mesh resolution are also similar to those in Section 4.7.2. The buoy is 0.18 m in diameter and is initially located 4.6 m from the wave maker. The density of the buoy is 500 kg/m³. As shown in Fig. 15(a), the buoy is initially half-submerged in the water. The buoy is not constrained and can be displaced in any direction. Note that the wave maker and the buoy are both represented by the same scalar function \( \varphi \) and that the velocity of the paddle wave maker is prescribed (see Section 4.7.2), while the velocity of the buoy is solved for.

Fig. 15 shows the interactions of the buoy with the waves at various times. The buoy remains almost still during an initial period, when the wave train is still traveling towards it. The displacements of the buoy become large when the first wave reaches it. Fig. 16 shows the horizontal and vertical displacements of the buoy (center of mass) in time. It is seen in Fig. 16(a) that the horizontal displacement (i.e., surge) of the buoy is negligible in the first 7 s, when the wave train is still traveling towards the buoy and has not reached it yet. After that initial period, the buoy steadily moves towards the end of the tank and reaches the damping zone at \( t \approx 40 \) s. Fig. 16(b) shows the heave oscillations of the buoy, where it can be seen that the average amplitude of heave oscillations is about 5 cm, which is comparable to the wave height.

5. Conclusion

A computational tool for simulation of two-way interactions between incompressible two-fluid flows and 2D moving rigid bodies was presented. The two-step projection method with GPU acceleration was used to solve the flow equations. The fictitious domain method was used to capture the fluid–solid interaction, in which the rigidity constraint is imposed on the solid domain by conserving the linear and angular momentum. To capture the evolution of phase interfaces in the three-phase system, the volume-of-fluid method with two scalar functions, representing the solid domain and one of the two fluids, was employed. Using a geometrical approach, the interfaces in three-phase cells were reconstructed and the triple points were captured. A consistent mass and momentum transport scheme was used to allow for simulations of multiphase flows of arbitrarily large density ratios. The aim was to develop a versatile computational tool that is applicable to a wide range of problems. The performance and accuracy of the solver were assessed through a set of canonical test cases and the results showed very good agreement with the theoretical, experimental, and other numerical results. The solver was used to simulate the interactions between a free-floating rigid body and waves generated by a bottom-hinged paddle wave maker in a tank.

Acknowledgments

The research support from the National Science Foundation under the CBET Grant Nos. 1236462 and 1336232 is gratefully acknowledged. The simulations presented here were performed on the UMass Dartmouth HPC Cluster, which was purchased through partial support from the NSF (MRI Grant No. CNS-0959382) and AFOSR (DURIP Grant No. FA9550-10-1-0354). AP gratefully acknowledges the Fellowship from the Office of the Associate Provost for Graduate Studies at UMass Dartmouth.

Appendix A

A brief description of the geometrical constructs used in interface reconstruction in three-phase cells is presented here. Consider the three-phase system shown in Fig. 1. Reconstruction of the solid surface provides the end points of the solid interface in cell C, which are denoted by \( X_{ns} \) and \( X_{ni} \) in Fig. 1(b). The points \( X_{ns} \) and \( X_{ni} \) are the vertices of the polygon \( P \). Other vertices of the polygon \( P \) are obtained by identifying the vertices of cell C that lie outside the solid domain. This is done by defining a distance function as,

\[
d = (X_{ns} - X_i) \cdot \hat{n}_i
\]

where \( X_i \) is a vertex of cell C and \( \hat{n}_i \) is the normal vector to the solid surface \((\hat{n} = \nabla \varphi / |\nabla \varphi|)\). If \( d > 0 \), then the vertex \( X_i \) lies outside the solid domain; otherwise, inside. The vertices of the polygon \( P \) need to be ordered before proceeding further. We use the “Quickhull” algorithm [61] to sort the vertices of a polygon in a counter-clockwise direction.

To calculate the orientation of line \( M \) that is originating from a given \( z \), we find the root of a function \( h \) expressed as,

\[
h(x) = V(x) - V_{act}
\]

where \( V(x) \) is the volume (area in 2D) truncated by line \( M \) (the green area inside polygon \( P \) shown in Fig. 1)) and \( V_{act} \) is the actual fluid volume in cell C. As shown in Fig. 1(b), point \( x \) is the intersection point of line \( M \) and the polygon \( P \) that is the furthest to point \( z \). Point \( x \) can lie on any side of the polygon \( P \). To determine the precise location of \( x \), we first find two consecutive vertices of polygon \( P \), say \( r_1 \) and \( r_2 \), between which the function \( h \) changes sign. \( x \) then lies on the line segment joining \( r_1 \) and \( r_2 \).

\[
x = r_1 + (1 - \beta) \cdot (r_2 - r_1)
\]

where \( 0 < \beta < 1 \) (Eq. (29)) now becomes a function of parameter \( \beta \). Using the bisection method to find the root of \( h(x) \), point \( x \) is located. \( V(x) \) in Eq. (29) can be calculated by constructing a second polygon resulting from the intersection of line \( M \) with polygon \( P \). Similar steps that were taken to construct the polygon \( P \) can be employed in the construction of the new polygon. Given point \( x \) on the polygon \( P \), we first find the other point due to intersection of line \( M \) with polygon \( P \). Subsequently, we identify the vertices of the polygon \( P \) which lie on the fluid side of line \( M \). The resultant second polygon is shown by blue area inside cell C in Fig. 1(b). All vertices of the new polygon are sorted in the counter-clockwise direction and used to calculate the polygon area using the following formula,

\[
A = \frac{1}{2} \sum_{i=1}^{n} (X_{i-1}y_i - X_iy_{i-1})
\]

where \( (x_i, y_i) \) are the coordinates of vertex \( i \) of the \( n \)-sided polygon and vertex \( i = n + 1 \) is assumed to coincide with vertex \( i = 1 \).

References


Appendix A


