Fully implicit and accurate treatment of jump conditions for two-phase incompressible Navier–Stokes equation

Hyuntae Cho and Myungjoo Kang

May 29, 2020

Abstract

We present a numerical method for two-phase incompressible Navier–Stokes equation with jump discontinuity in the normal component of the stress tensor and in the material properties. Although the proposed method is only first-order accurate, it does capture discontinuity sharply, not neglecting nor omitting any component of the jump condition. Discontinuities in velocity gradient and pressure are expressed using a linear combination of singular force and tangential derivatives of velocities to handle jump conditions in a fully implicit manner. The linear system for the divergence of the stress tensor is constructed in the framework of the ghost fluid method, and the resulting saddle-point system is solved via an iterative procedure. Numerical results support the inference that the proposed method converges in L^∞ norms even when velocities and pressures are not smooth across the interface and can handle a large density ratio that is likely to appear in a real-world simulation.

1 Introduction

Two-phase incompressible flows are ubiquitous in real life, and thus numerical simulations of these flows are crucial in applications such as oil–water core annular flow, biofluid dynamics, and analysis of gas bubbles rising in water. One popular approach to handling the interface between the two fluids involved is to use a body-fitted mesh. However, in this study, we will narrow our interest to the Cartesian grid method, which is free of mesh generation. The very early work of Peskin [29], which is also known as the immersed boundary method, is a typical example of a Cartesian grid method. It uses numerical δ-functions to handle singular forces on the interface between fluid and solid. This idea was utilized together with the front-tracking method [40, 41] and the level-set method [33, 34] for the numerical simulation of incompressible two-phase flows. However, such an approach using smoothed δ-functions cannot avoid numerical diffusion near the interfaces and eventually encounters interfaces with non-zero thicknesses.

To avoid the smearing out of the interfaces involved in these incompressible two-phase flows, sharp capturing methods have been developed by researchers over the years. For example, the immersed interface method (IIM) [18] has earned a reputation as a second-order finite difference method for elliptic interface problems. The fundamental idea of IIM has been applied to the incompressible Stokes equation [19] and Navier–Stokes equation [17, 22], on the assumption that the viscosities and densities of the two fluids are identical. Later, introducing augmented variables together with the interfaces, Li et al. [21] used IIM to solve Stokes equations with discontinuous viscosities. However, instead of the marker-and-cell (MAC) grid, a collocated grid was used, which caused periodic boundary conditions to be imposed for the pressure. To address this issue, solution methods utilizing the MAC grid have been developed for the two-phase Stokes equation [4, 38]. For Navier–Stokes equations with discontinuous viscosities, IIM has also demonstrated its success in capturing non-smooth velocities and pressures [36, 37], but relatively less work is done when the density is discontinuous across the interface. (For more detailed explanations and applications of IIM, see [20].)

Another famous sharp capturing method is the ghost fluid method (GFM), which was first introduced by Fedkiw et al. [8] for capturing contact discontinuity in compressible flows. Its concept was later employed in solving elliptic interface problems [23]. For incompressible flows [15], techniques by [23] have been applied in approximating viscous terms and solving Poisson's equations of pressure from the projection method [6], resulting in the sharp capture of the surface tension. Whereas the viscous term was discretized explicitly in GFM, Sussman et al. [35] introduced sharp capturing methods that involve semi-implicit treatments of the viscous term, allowing for larger time steps. Even though the details of the implementations by [15] and [35] differ, Lalanne et al. [16] have demonstrated that the two approaches are actually equivalent.

Two pioneering works on simulating incompressible flows in the framework of the level-set/ghost fluid method [15,35] used the projection method to solve for fluid velocity. However, it should be noted that when the projection method is used, the jump conditions of intermediate or predictor velocities differ from those of the original velocities. Nevertheless, the same jump conditions are applied occasionally to simulate two-phase flows. To impose jump conditions accurately, approaches that are alternative to the projection method have been created. One example is the virtual node method,

which was first developed for elliptic interface problems [2, 12] and then extended to incompressible flows [1, 31]. It directly discretizes the Navier–Stokes equation together with the divergence-free condition to obtain a saddle-point linear system of velocity and pressure. On the other hand, Saye has used the gauge method to simulate incompressible flows [30] where the jump conditions were reformulated with auxiliary and gauge variables. Recently, in [39], a modified projection method was developed for a sharp capturing method that uses the quad/octree grid. The projection step was repeated until the corrected velocity and pressure satisfied the jump conditions. The researchers also remarked that many existing numeric methods omit some parts of the jump conditions to simplify implementation.

In this paper, we introduce a new sharp capturing method for two-phase flows, characterized by the accurate and fully implicit treatment of jump conditions. Different from [30, 39], our method shares similarities with the virtual node method. That is, instead of introducing an auxiliary variable, our method discretizes the divergence of the stress tensor directly in the framework of GFM. Expressions for the jump conditions of velocity gradient and pressure using tangential derivatives of velocities are calculated to develop a ghost fluid method for viscous terms and gradients of pressure. We note that the proposed method is only first-order accurate but considers jump conditions accurately and implicitly. We begin in section 2 with equations on two-phase incompressible flows and on the movements of the interfaces. Section 3 explains the details of the proposed method. Section 4 then follows with descriptions and accounts of the numerical experiments.

2   Governing Equations

We consider two incompressible, immiscible, and viscous flows on Ω,

$$
\begin{aligned}
\rho^{\pm}(\mathbf{U} + \mathbf{U} \cdot \nabla \mathbf{U}) &= \mu^{\pm} \triangle \mathbf{U} - \nabla p + \mathbf{f} \text{ on } \Omega^{\pm}, \\
\nabla \cdot \mathbf{U} &= 0 \text{ on } \Omega.
\end{aligned}
\tag{1}
$$

In addition to (1), jump conditions are given at the interface Γ:

$$
\begin{aligned}
[\mathbf{U}] &= 0, \\
[\sigma \mathbf{n}] &= \mathbf{G},
\end{aligned}
\tag{2}
$$

for stress tensor $\sigma = \mu \left(\nabla \mathbf{U} + \nabla \mathbf{U}^T \right) - p\mathbf{I}$. $[\mathbf{V}] = \mathbf{V}^+ - \mathbf{V}^-$ denotes the jump condition along the interface, where the superscripts "+" and "−" refer to $\Omega^{\pm}$. Here, $\mathbf{n}$ is a normal to the interface, and $\mathbf{G}$ is a singular force term across the interface. We are especially interested in the case where $\mathbf{f} = -\rho \mathbf{g}$ and $\mathbf{G} = \beta \kappa \mathbf{n}$, where $\mathbf{g}$ refers to gravity, κ denotes the mean curvature of the interface, and β is a surface tension coefficient.

In this study, the level-set method [27] is used to capture the interface of two different fluids as a zero level-set of the continuous function ϕ. Thus, the interface and two sub-domains at time t can be represented as

$$
\begin{aligned}
\Gamma &= \{\mathbf{x} \in \Omega | \phi(\mathbf{x}, t) = 0\}, \\
\Omega^+ &= \{\mathbf{x} \in \Omega | \phi(\mathbf{x}, t) > 0\}, \\
\Omega^- &= \{\mathbf{x} \in \Omega | \phi(\mathbf{x}, t) < 0\}.
\end{aligned}
\tag{3}
$$

An advantage of the level-set method is that the evolution of the interface Γ with the fluid velocity $\mathbf{U}$ can be formulated as

$$
\phi_t + \mathbf{U} \cdot \nabla \phi = 0.
\tag{4}
$$

Furthermore, geometric quantities of the interfaces, such as the normal $\mathbf{n}$ and curvature κ in (2), are calculated using the level-set function:

$$
\begin{aligned}
\mathbf{n} &= \frac{\nabla \phi}{|\nabla \phi|}, \\
\kappa &= \nabla \cdot \mathbf{n}.
\end{aligned}
\tag{5}
$$

For a more detailed explanation and application of the level-set method, see [10, 26].

3   Numerical Methods

We use a staggered MAC grid [11] for spatial discretization. Pressure p and level-set function ϕ are placed in the cell centers, whereas the values from $\mathbf{U}$ are stored on the cell faces. Figure 1 illustrates the locations of the variables in 2D.

Overall discretization of the proposed method is similar to that of [31]. At the beginning of each time step, ϕ^n is advected to ϕ^{n+1} via the discretization of (4) with third-order total variation diminishing (TVD) Runge-Kutta [32] in time and fifth-order weighted essentially non-oscillatory (WENO) scheme [14] in space. To discretize (4), the velocity
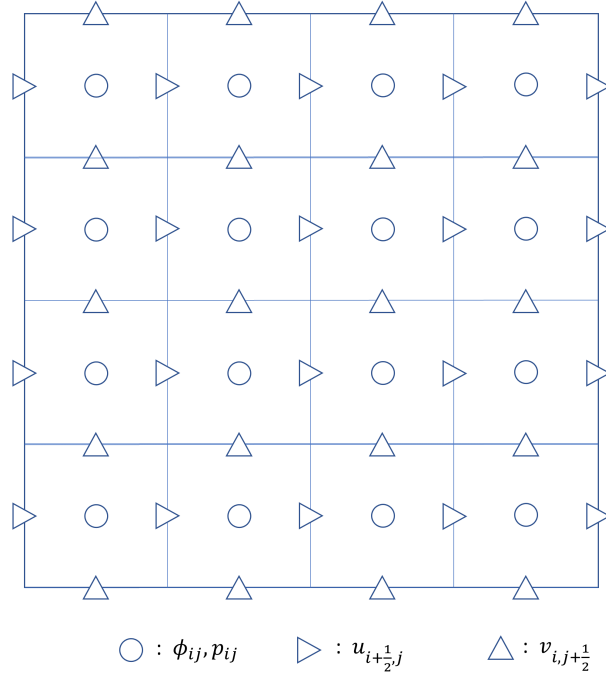
$$\bigcirc : \phi_{ij}, p_{ij} \qquad \triangleright : u_{i+\frac{1}{2},j} \qquad \triangle : v_{i,j+\frac{1}{2}}$$

Figure 1: MAC grid

at cell center needs to be defined, which will be done later. After the advection, ϕ^{n+1} is reinitialized through the solving of the partial differential equation

$$\phi_\xi + \text{sign}(\phi)(|\nabla \phi| - 1) = 0 \qquad (6)$$

for pseudo time ξ. Discretization of (6) follows that of [25], which uses second-order essentially non-oscillatory (ENO) scheme and TVD-RK2 with subcell resolution technique. The ϕ^{n+1} on the cell faces are evaluated via linear interpolation of the value defined at the cell center:

$$\phi^{n+1}_{i+\frac{1}{2},j} = \frac{\phi^{n+1}_{i,j} + \phi^{n+1}_{i+1,j}}{2},$$
$$\phi^{n+1}_{i,j+\frac{1}{2}} = \frac{\phi^{n+1}_{i,j} + \phi^{n+1}_{i,j+1}}{2}.$$

These values will determine if the center of a cell face belongs to either Ω^+ or Ω^-. For the Navier–Stokes equation, we use semi-Lagrangian with backward difference formula to construct a saddle-point system of $\mathbf{U}$ and p:

$$\rho^{n+1} \frac{\mathbf{U}^{n+1} - \mathbf{U}^n_d}{\Delta t} = (\mu \triangle \mathbf{U} - \nabla p + \mathbf{f})^{n+1},$$
$$(\nabla \cdot \mathbf{U})^{n+1} = 0. \qquad (7)$$

Here, $\mathbf{U}^n_d$ is an approximation of $\mathbf{U}^n$ at departure point $\mathbf{x}_d$ traced backward along the characteristic curve $\frac{d\mathbf{x}}{dt} = \mathbf{U}$ from time level t^{n+1}. Geometric quantities (5) that appear in the jump condition are approximated with ϕ^{n+1} using second-order central differences. Because the Navier–Stokes equation is implicitly discretized, whether the grid point belongs to Ω^+ or Ω^- is determined by the sign of ϕ^{n+1} rather than that of ϕ^n.

Discretizing viscous terms and pressure using jump condition (2) is quite a challenging problem. When the viscous term is treated explicitly, the higher truncation error near the interface does not spread out. Furthermore, as the grid is refined, a significant restriction is imposed on the time step. On the other hand, if the viscous term is discretized implicitly using the projection method, the jump condition of $\mathbf{U}$ will differ from the jump condition of the intermediate or predictor velocity. However, this difference is sometimes ignored. Direct discretization of Navier–Stokes equation (7) enables us to treat the viscous term implicitly with accurate jump conditions.

In the following subsections, we will first derive a jump condition equivalent to (2) without neglecting any component. To emphasize the sharp discretization of viscous terms and pressure using the jump condition, a solution method for two-phase steady-state Stokes equations that involve these jump conditions will be presented. Afterward, with some modifications, the numerical algorithm for (7) will be completed.

3

3.1 Derivation of jump conditions

A jump condition equivalent to (2) is first derived through the replacement of normal derivatives with a linear combination of Cartesian and tangential derivatives, as was done for the elliptic interface problem in [5, 7]. The normal and tangent vectors of the interface are then defined as $\mathbf{n} = (n_x, n_y)$ and $\boldsymbol{\tau} = (-n_y, n_x)$, respectively.

First, the inner product of (2) and $\boldsymbol{\tau}$ is calculated:

$$
\begin{aligned}
\mathbf{G} \cdot \boldsymbol{\tau} &= [\mu(\boldsymbol{\tau} \cdot \nabla \mathbf{U} \cdot \mathbf{n} + \mathbf{n} \cdot \nabla \mathbf{U} \cdot \boldsymbol{\tau})] \\
&= [\mu(-n_y u_{\mathbf{n}} + n_x v_{\mathbf{n}} + n_x u_{\boldsymbol{\tau}} + n_y v_{\boldsymbol{\tau}})].
\end{aligned}
\tag{8}
$$

Here, $u_{\mathbf{n}} = \nabla u \cdot \mathbf{n}, u_{\boldsymbol{\tau}} = \nabla u \cdot \boldsymbol{\tau}, v_{\mathbf{n}} = \nabla v \cdot \mathbf{n}, v_{\boldsymbol{\tau}} = \nabla v \cdot \boldsymbol{\tau}$. Because divergence is invariant under rotation,

$$
n_x u_{\mathbf{n}} + n_y v_{\mathbf{n}} - n_y u_{\boldsymbol{\tau}} + n_x v_{\boldsymbol{\tau}} = \frac{\partial (\mathbf{U} \cdot \mathbf{n})}{\partial \mathbf{n}} + \frac{\partial (\mathbf{U} \cdot \boldsymbol{\tau})}{\partial \boldsymbol{\tau}} = \nabla \cdot \mathbf{U} = 0.
\tag{9}
$$

n_y is multiplied to (8), and $n_y v_{\mathbf{n}}$ is substituted for $-n_x u_{\mathbf{n}} + n_y u_{\boldsymbol{\tau}} - n_x v_{\boldsymbol{\tau}}$:

$$
n_y \mathbf{G} \cdot \boldsymbol{\tau} = \left[\mu(-u_{\mathbf{n}} + 2n_x n_y u_{\boldsymbol{\tau}} + (n_y^2 - n_x^2)v_{\boldsymbol{\tau}}) \right].
$$

Similarly, the following equation is obtained:

$$
n_x \mathbf{G} \cdot \boldsymbol{\tau} = \left[\mu(v_{\mathbf{n}} + 2n_x n_y v_{\boldsymbol{\tau}} + (n_x^2 - n_y^2)u_{\boldsymbol{\tau}}) \right].
$$

The jump conditions of the derivatives in the Cartesian directions can be decomposed into conditions in the normal and tangential directions:

$$
\begin{aligned}
[\mu u_x] &= [\mu u_{\mathbf{n}}]n_x - [\mu u_{\boldsymbol{\tau}}]n_y \\
&= [\mu u_{\mathbf{n}}]n_x - [\mu]u_{\boldsymbol{\tau}}n_y \\
[\mu u_y] &= [\mu u_{\mathbf{n}}]n_y + [\mu u_{\boldsymbol{\tau}}]n_x \\
&= [\mu u_{\mathbf{n}}]n_y + [\mu]u_{\boldsymbol{\tau}}n_x.
\end{aligned}
$$

The jump condition $[u] = [v] = 0$ is used, resulting in $[\mu u_{\boldsymbol{\tau}}] = [\mu]u_{\boldsymbol{\tau}}^- + \mu^+[u_{\boldsymbol{\tau}}] = [\mu]u_{\boldsymbol{\tau}}$. The superscripts "+" and "−" for u_τ are dropped because $[u_\tau] = 0$. When these resulting expressions are combined, the jump condition for the Cartesian component of $[\mu \nabla u]$ is produced:

$$
\begin{aligned}
[\mu u_x] &= [\mu](2n_x^2 n_y - n_y)u_{\boldsymbol{\tau}} + [\mu](n_x n_y^2 - n_x^3)v_{\boldsymbol{\tau}} - n_x n_y \mathbf{G} \cdot \boldsymbol{\tau}, \\
[\mu u_y] &= [\mu](2n_x n_y^2 + n_x)u_{\boldsymbol{\tau}} + [\mu](n_y^3 - n_x^2 n_y)v_{\boldsymbol{\tau}} - n_y^2 \mathbf{G} \cdot \boldsymbol{\tau}.
\end{aligned}
\tag{10}
$$

Similarly, the formula for $[\mu \nabla v]$ can be derived:

$$
\begin{aligned}
[\mu v_x] &= [\mu](-2n_x^2 n_y - n_y)v_{\boldsymbol{\tau}} + [\mu](n_x n_y^2 - n_x^3)u_{\boldsymbol{\tau}} + n_x^2 \mathbf{G} \cdot \boldsymbol{\tau}, \\
[\mu v_y] &= [\mu](-2n_x n_y^2 + n_x)v_{\boldsymbol{\tau}} + [\mu](n_y^3 - n_x^2 n_y)u_{\boldsymbol{\tau}} + n_x n_y \mathbf{G} \cdot \boldsymbol{\tau}.
\end{aligned}
\tag{11}
$$

Lastly, the inner product of (2) and $\mathbf{n}$ is calculated:

$$
\mathbf{G} \cdot \mathbf{n} = [2\mu(\mathbf{n} \cdot \nabla \mathbf{U} \cdot \mathbf{n}) - p].
$$

From (9),

$$
\mathbf{n} \cdot \nabla \mathbf{U} \cdot \mathbf{n} = -\boldsymbol{\tau} \cdot \nabla \mathbf{U} \cdot \boldsymbol{\tau} = -n_y u_{\boldsymbol{\tau}} + n_x v_{\boldsymbol{\tau}}.
$$

Therefore,

$$
[p] = -2[\mu](-n_y u_{\boldsymbol{\tau}} + n_x v_{\boldsymbol{\tau}}) - \mathbf{G} \cdot \mathbf{n}.
\tag{12}
$$

Through this formula, we express the jump condition $[\mu \nabla \mathbf{U}], [p]$ as a linear combination of the singular force $\mathbf{G}$ and the tangential derivatives of the velocities.

Remark The jump condition formula can be extended to three dimensions with few modifications. After constructing two tangent vectors with respect to the normal vector, one can derive two jump conditions of velocities, similar to (8), and obtain a divergence-free condition in the normal and tangent coordinates. With an appropriate linear combination of these three equations, jump conditions for the velocity gradients can be obtained. For the jump condition of pressure, we first calculate the inner product of the normal vector and (2). After replacing the normal derivatives of the velocities with the tangential derivatives of the velocities using divergence-free condition, we obtain a three-dimensional version of (12).

3.2 Numerical methods for two-phase steady-state Stokes equation

Ignoring the material derivative with density in (1), we consider the incompressible two-phase steady-state Stokes equation:

$$\mu^{\pm} \triangle \mathbf{U} - \nabla p + \mathbf{f} = 0 \text{ on } \Omega^{\pm},$$
$$\nabla \cdot \mathbf{U} = 0 \text{ on } \Omega, \tag{13}$$
$$[\sigma \mathbf{n}] = G \text{ on } \Gamma.$$

Here, we introduce numerical methods to solve (13) in a sharp manner. The idea of xGFM [7] is extended to jump conditions where $\mathbf{U}$ and p are coupled together. We construct an iterative method that corrects the jump conditions and solutions together for every iterative step. Details of the algorithms will be presented under the assumption of two dimensions, whereas extension to three dimensions will be straightforward.

3.2.1 Visiting two-phase steady-state Stokes equation with ghost fluid method

Under the assumption of two dimensions, the following are considered:

$$\mu^{\pm} \triangle u - p_x = -f_1 \text{ on } \Omega^{\pm}$$
$$\mu^{\pm} \triangle v - p_y = -f_2 \text{ on } \Omega^{\pm} \tag{14}$$
$$u_x + v_y = 0 \text{ on } \Omega^{\pm}$$

with the jump conditions

$$[\mu \nabla u] = \mathbf{c} \text{ on } \Gamma$$
$$[\mu \nabla v] = \mathbf{d} \text{ on } \Gamma \tag{15}$$
$$[p] = a \text{ on } \Gamma$$

instead of (2). Although (14) with jump conditions (15) result in over-determined partial differential equations, GFM is a suitable method for solving such a problem. When the methodologies of GFM are followed, (14) with (15) may be discretized as follows:

$$(\mu \triangle u)_{i+\frac{1}{2},j}^{\mathrm{GFM}} - \frac{p_{i+1,j} - p_{i,j}}{\Delta x} = -f_1(x_{i+\frac{1}{2}}, y_j) + c_u + a_u$$
$$(\mu \triangle v)_{i,j+\frac{1}{2}}^{\mathrm{GFM}} - \frac{p_{i,j+1} - p_{i,j}}{\Delta y} = -f_2(x_i, y_{j+\frac{1}{2}}) + d_v + a_v \tag{16}$$
$$\frac{u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}}}{\Delta y} = 0.$$

Here, $\mu \triangle u$ and p_x at $(x_{i+\frac{1}{2}}, y_j)$ are approximated as $(\mu \triangle u)_{i+\frac{1}{2},j}^{\mathrm{GFM}} - c_u$ and $\frac{p_{i+1,j} - p_{i,j}}{h} + a_u$, respectively. $(\mu \triangle u)_{i+\frac{1}{2},j}^{\mathrm{GFM}}$ denotes the discrete Laplacian that appears in GFM [23], and c_u is the correction term added to the right-hand side. If $(x_{i+\frac{1}{2}}, y_j) \in \Omega^{\pm}$ is assumed,

$$(\mu \triangle u)_{i+\frac{1}{2},j}^{\mathrm{GFM}} = \left(\hat{\mu}_R(u_{i+\frac{3}{2},j} - u_{i+\frac{1}{2},j}) + \hat{\mu}_L(u_{i-\frac{1}{2},j} - u_{i+\frac{1}{2},j}) \right) \Big/ \Delta x^2 +$$
$$\left(\hat{\mu}_T(u_{i+\frac{1}{2},j+1} - u_{i+\frac{1}{2},j}) + \hat{\mu}_B(u_{i+\frac{1}{2},j-1} - u_{i+\frac{1}{2},j}) \right) \Big/ \Delta y^2 ,$$
$$c_u = c_R + c_L + c_T + c_B,$$

where

$$\hat{\mu}_R = \begin{cases} \mu^{\pm} & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i+\frac{3}{2},j} > 0) \\ \frac{\mu^+ \mu^-}{\mu^{\pm}\theta_R + \mu^{\mp}(1-\theta_R)} & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i+\frac{3}{2},j} \leq 0) \end{cases}$$

and

$$c_R = \begin{cases} 0 & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i+\frac{3}{2},j} > 0) \\ \pm\hat{\mu}_R \frac{(1-\theta_R)[\mu u_x]_R}{\mu^{\mp}\Delta x} & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i+\frac{3}{2},j} \leq 0) \end{cases}$$

for $\theta_R = \frac{|\phi_{i+\frac{1}{2},j}|}{|\phi_{i+\frac{1}{2},j}|+|\phi_{i+\frac{3}{2},j}|}$. $\hat{\mu}_L, \hat{\mu}_T, \hat{\mu}_B$ and c_L, c_T, c_B are defined similarly. (For a more detailed explanation, see [23].) In a similar fashion, the correction term for p_x is determined as

$$a_u = a_L + a_R,$$

where

$$
a_R = \begin{cases} 0 & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i+1,j} > 0 \\ \mp\frac{[p]_R}{\Delta x} & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i+1,j} < 0 \end{cases},
$$

$$
a_L = \begin{cases} 0 & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i,j} > 0 \\ \pm\frac{[p]_L}{\Delta x} & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i,j} < 0 \end{cases}.
$$

The formulation of $(\mu \triangle v)_{i,j+\frac{1}{2}}^{\text{GFM}}, d_v, a_v$ is similar to that of $(\mu \triangle u)_{i+\frac{1}{2},j}^{\text{GFM}}, c_u, a_u$. For simplicity, (16) is rewritten as

$$
A(u, v, p) = b(a, \mathbf{c}, \mathbf{d}, \mathbf{f}). \tag{17}
$$

Notably, A is multi-linear with respect to u, v, and p, whereas b is multi-linear with respect to $a, \mathbf{c}$, and $\mathbf{d}$.

Remark Note that linear system (16) is symmetric. A correction term is not added to the divergence-free equation $u_x + v_y = 0$. Furthermore, a $O(1)$ truncation error occurs near the interface. If one uses jump conditions $[\mu u_x], [\mu v_y]$ to discretize a divergence-free equation, the truncation error near the interface becomes $O(h)$ for $h = \max(\Delta x, \Delta y)$ but breaks the symmetry of the matrix when μ^- and μ^+ are different. Because $\mu \triangle \mathbf{U} - \nabla p + \mathbf{f} = 0$ are discretized with $O(1)$ truncation error near the interface, considering the jump condition for a divergence-free equation will not increase the order of convergence dramatically, but rather, will make the linear system difficult to solve by creating a non-symmetric linear system.

3.2.2 Velocity extrapolation algorithm and tangential derivative at the interface

Here, the velocity extrapolation algorithm of [7] is revisited. The following pseudo time-dependent partial differential equation is first considered:

$$
\begin{aligned}
\frac{\partial \hat{u}}{\partial t} + \text{sign}(\phi)\mathbf{n} \cdot \nabla \hat{u} &= 0, \\
\frac{\partial \hat{v}}{\partial t} + \text{sign}(\phi)\mathbf{n} \cdot \nabla \hat{v} &= 0, \\
\hat{u} = u, \hat{v} &= v \text{ on } \Gamma.
\end{aligned} \tag{18}
$$

The steady-state solution of (18) can be viewed as an extrapolation of u, v off the interface and will be used to approximate tangential derivatives of u, v on the interface. Equation (18) is then discretized using a first-order upwind scheme. For example, if $\phi_{i+\frac{1}{2},j} > 0$ is assumed, it is discretized as

$$
\frac{\hat{u}_{i+\frac{1}{2},j}^{l+1} - \hat{u}_{i+\frac{1}{2},j}^{l}}{\Delta t_{i+\frac{1}{2},j}} + (n_x^+ D_x^- \hat{u} + n_x^- D_x^+ \hat{u}) + (n_y^+ D_y^- \hat{u} + n_y^- D_y^+ \hat{u}) = 0 \tag{19}
$$

for $n_x^+ = \max(n_x, 0), n_x^- = \min(n_x, 0)$, and similarly defined $n_y^\pm$. The boundary condition of (18) is applied on Γ using a sub-cell resolution technique:

$$
D_x^- \hat{u}_{i+\frac{1}{2},j} = \begin{cases} \frac{\hat{u}_{i+\frac{1}{2},j}^{l} - \hat{u}_{i-\frac{1}{2},j}^{l}}{\Delta x} & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i-\frac{1}{2},j} > 0 \\ \frac{\hat{u}_{i+\frac{1}{2},j}^{l} - u_\Gamma}{\theta_L \Delta x} & \text{if } \phi_{i+\frac{1}{2},j}\phi_{i-\frac{1}{2},j} < 0 \end{cases}. \tag{20}
$$

$\theta_L = \frac{|\phi_{i+\frac{1}{2},j}|}{|\phi_{i+\frac{1}{2},j}| + |\phi_{i-\frac{1}{2},j}|}$ is measured to approximate the location of the interface. The interfacial value u_Γ is obtained according to the formula in [23]:

$$
u_\Gamma = \left(\mu_2 \theta_L u_{i+\frac{1}{2},j} + \mu_1(1 - \theta_L) u_{i-\frac{1}{2},j} - \text{sign}(\phi_{i+\frac{1}{2},j}) \left([\mu u_x] (1 - \theta_L)\theta_L \Delta x \right) \right) \Big/ \hat{\mu}
$$

for

$$
\mu_1 = \begin{cases} \mu^+ \text{ if } \phi_{i+\frac{1}{2},j} > 0 \\ \mu^- \text{ if } \phi_{i+\frac{1}{2},j} < 0 \end{cases}, \quad \mu_2 = \begin{cases} \mu^+ \text{ if } \phi_{i-\frac{1}{2},j} > 0 \\ \mu^- \text{ if } \phi_{i-\frac{1}{2},j} < 0 \end{cases}, \quad \hat{\mu} = \mu_2 \theta_L + \mu_1(1 - \theta_L).
$$

Other derivatives and interface boundary conditions are computed similarly. To avoid small time-stepping in the whole domain, the local pseudo time step is set to $\Delta t_{i+\frac{1}{2},j} = CFL \times \min(\theta_R \Delta x, \theta_L \Delta x, \theta_B \Delta y, \theta_T \Delta y)$, where $\theta_L = 1$ if $\phi_{i+\frac{1}{2},j}\phi_{i-\frac{1}{2},j} > 0$. The idea of taking a grid-dependent time step and using it in reinitialization can be found in [24].

Equation (18) is solved only on the grid points near the interface, and $u^{\text{ext}}, v^{\text{ext}}$ is denoted as the steady-state solution of (18). Specifically, extrapolations were performed for the points where $|\phi_{i+\frac{1}{2},j}|, |\phi_{i,j+\frac{1}{2}}| \leq 5 \max(\Delta x, \Delta y)$

and $u^{\text{ext}}_{i+\frac{1}{2},j} = \hat{u}^{l+1}_{i+\frac{1}{2},j}$ when $|\hat{u}^{l+1}_{i+\frac{1}{2},j} - \hat{u}^{l}_{i+\frac{1}{2},j}| < \epsilon$ for every i,j with tolerance ϵ. After steady-state solutions u^{ext} and v^{ext} are obtained, the followings are defined:

$$u^{\text{ext}}_{\boldsymbol{\tau}} = \nabla u^{\text{ext}} \cdot \boldsymbol{\tau}, \quad v^{\text{ext}}_{\boldsymbol{\tau}} = \nabla v^{\text{ext}} \cdot \boldsymbol{\tau} \tag{21}$$

at grid points $(x_{i+\frac{1}{2}}, y_j)$ for

$$\nabla u^{\text{ext}}(x_{i+\frac{1}{2}}, y_j) = \begin{pmatrix} \frac{u^{\text{ext}}_{i+\frac{3}{2},j} - u^{\text{ext}}_{i-\frac{1}{2},j}}{2\Delta x} \\ \frac{u^{\text{ext}}_{i,j+1} - u^{\text{ext}}_{i,j-1}}{2\Delta y} \end{pmatrix}, \quad \nabla v^{\text{ext}}(x_{i+\frac{1}{2}}, y_j) = \begin{pmatrix} \frac{v^{\text{ext}}_{i+1,j+\frac{1}{2}} + v^{\text{ext}}_{i+1,j-\frac{1}{2}} - v^{\text{ext}}_{i,j+\frac{1}{2}} - v^{\text{ext}}_{i,j-\frac{1}{2}}}{2\Delta x} \\ \frac{v^{\text{ext}}_{i+1,j+\frac{1}{2}} - v^{\text{ext}}_{i+1,j-\frac{1}{2}} + v^{\text{ext}}_{i,j+\frac{1}{2}} - v^{\text{ext}}_{i,j-\frac{1}{2}}}{2\Delta y} \end{pmatrix}.$$

$\boldsymbol{\tau}_{i+\frac{1}{2},j} = (-n_y, n_x)_{i+\frac{1}{2},j}$ is obtained from $\mathbf{n} = (n_x, n_y)_{i+\frac{1}{2},j}$ computed with the level-set function ϕ. Because the steady-state solution $u^{\text{ext}}, v^{\text{ext}}$ is a reasonable extrapolation of u, v off the interface, $u^{\text{ext}}_{\boldsymbol{\tau}}, v^{\text{ext}}_{\boldsymbol{\tau}}$ can be viewed as an extension of $u_{\boldsymbol{\tau}}, v_{\boldsymbol{\tau}}$ at Γ to the grid points near the interface. $u^{\text{ext}}_{\boldsymbol{\tau}}, v^{\text{ext}}_{\boldsymbol{\tau}}$ at $(x_i, y_{j+\frac{1}{2}})$ can be computed with few modifications. Note that $\text{sign}(\phi)\mathbf{n}$ does not depend on $\hat{u}, \hat{v}$, whereas $\hat{u}^{l+1}_{i+\frac{1}{2},j}$ linearly depends on $\hat{u}^{l}_{i+\frac{1}{2},j}$ and u_Γ. Therefore, we may conclude that $u^{\text{ext}}_{\boldsymbol{\tau}}$ is a linear combination of $[\mu u_x], [\mu u_y]$, and u.

3.2.3 Iterative procedure

Steady-state Stokes equations (14) with jump condition (2) can be reformulated into equivalent systems of partial differential equations using (10), (11), and (12) to develop iterative methods:

$$\begin{aligned} \mu \triangle u - p_x &= -f_1 \text{ on } \Omega^{\pm} \\ \mu \triangle v - p_y &= -f_2 \text{ on } \Omega^{\pm} \\ u_x + v_y &= 0 \text{ on } \Omega^{\pm} \end{aligned} \tag{22}$$

with the jump condition

$$[\mu \nabla u] = \begin{pmatrix} [\mu](2n_x^2 n_y - n_y)u_{\boldsymbol{\tau}} & +[\mu](n_x n_y^2 - n_x^3)v_{\boldsymbol{\tau}} & -n_x n_y G \cdot \boldsymbol{\tau} \\ [\mu](2n_x n_y^2 + n_x)u_{\boldsymbol{\tau}} & +[\mu](n_y^3 - n_x^2 n_y)v_{\boldsymbol{\tau}} & -n_y^2 G \cdot \boldsymbol{\tau} \end{pmatrix} \text{ on } \Gamma$$

$$[\mu \nabla v] = \begin{pmatrix} [\mu](-2n_x^2 n_y - n_y)v_{\boldsymbol{\tau}} & +[\mu](n_x n_y^2 - n_x^3)u_{\boldsymbol{\tau}} & +n_x^2 G \cdot \boldsymbol{\tau} \\ [\mu](-2n_x n_y^2 + n_x)v_{\boldsymbol{\tau}} & +[\mu](n_y^3 - n_x^2 n_y)u_{\boldsymbol{\tau}} & +n_x n_y G \cdot \boldsymbol{\tau} \end{pmatrix} \text{ on } \Gamma \tag{23}$$

$$[p] = -2[\mu](-n_y u_{\boldsymbol{\tau}} + n_x v_{\boldsymbol{\tau}}) - G \cdot \mathbf{n} \text{ on } \Gamma.$$

Equation (22) is discretized according to 3.2.1, where jump condition (23) is discretized through the setting of $u_{\boldsymbol{\tau}} = u^{\text{ext}}_{\boldsymbol{\tau}}, v_{\boldsymbol{\tau}} = v^{\text{ext}}_{\boldsymbol{\tau}}$ obtained from 3.2.2. Although the discretized equation is linear, the solution of (22) depends on the jump condition, whereas jump condition (23) depends on the solution, which makes the linear system difficult to solve. Therefore, the solution of the Stokes equation is determined via the following iterative steps.

Given $a^k, \mathbf{c}^k, \mathbf{d}^k$, the value of $(\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{p}^{k+1})$ is solved through the discretization of the following systems according to 3.2.1 :

$$\begin{aligned} \mu \triangle \tilde{u}^{k+1} - \tilde{p}^{k+1}_x &= -f_1 \text{ on } \Omega^{\pm} \\ \mu \triangle \tilde{v}^{k+1} - \tilde{p}^{k+1}_y &= -f_2 \text{ on } \Omega^{\pm} \\ \tilde{u}^{k+1}_x + \tilde{v}^{k+1}_y &= 0 \text{ on } \Omega^{\pm} \\ [\mu \nabla \tilde{u}^{k+1}] &= \mathbf{c}^k \text{ on } \Gamma \\ [\mu \nabla \tilde{v}^{k+1}] &= \mathbf{d}^k \text{ on } \Gamma \\ [\tilde{p}^{k+1}] &= a^k \text{ on } \Gamma. \end{aligned} \tag{24}$$

Afterward, via the velocity extrapolation algorithm of 3.2.2, the steady-state solution $\tilde{u}^{\text{ext}}, \tilde{v}^{\text{ext}}$, with boundary condition given by the interface value of $\tilde{u}^{k+1}$ and $\tilde{v}^{k+1}$, is computed. With the use of the relations (10), (11), and (12), the following equations are defined:

$$\begin{aligned} \tilde{\mathbf{c}}^{k+1} &= \begin{pmatrix} [\mu](2n_x^2 n_y - n_y)\tilde{u}^{\text{ext}}_{\boldsymbol{\tau}} & +[\mu](n_x n_y^2 - n_x^3)\tilde{v}^{\text{ext}}_{\boldsymbol{\tau}} & -n_x n_y G \cdot \boldsymbol{\tau} \\ [\mu](2n_x n_y^2 + n_x)\tilde{u}^{\text{ext}}_{\boldsymbol{\tau}} & +[\mu](n_y^3 - n_x^2 n_y)\tilde{v}^{\text{ext}}_{\boldsymbol{\tau}} & -n_y^2 G \cdot \boldsymbol{\tau} \end{pmatrix} \\ \tilde{\mathbf{d}}^{k+1} &= \begin{pmatrix} [\mu](-2n_x^2 n_y - n_y)\tilde{v}^{\text{ext}}_{\boldsymbol{\tau}} & +[\mu](n_x n_y^2 - n_x^3)\tilde{u}^{\text{ext}}_{\boldsymbol{\tau}} & +n_x^2 G \cdot \boldsymbol{\tau} \\ [\mu](-2n_x n_y^2 + n_x)\tilde{v}^{\text{ext}}_{\boldsymbol{\tau}} & +[\mu](n_y^3 - n_x^2 n_y)\tilde{u}^{\text{ext}}_{\boldsymbol{\tau}} & +n_x n_y G \cdot \boldsymbol{\tau} \end{pmatrix} \\ \tilde{a}^{k+1} &= -2[\mu](-n_y \tilde{u}^{\text{ext}}_{\boldsymbol{\tau}} + n_x \tilde{v}^{\text{ext}}_{\boldsymbol{\tau}}) - G \cdot \mathbf{n}. \end{aligned} \tag{25}$$

$\tilde{\mathbf{c}}^{k+1}$ and $\tilde{\mathbf{d}}^{k+1}$ are defined at grid points $(x_{i+\frac{1}{2}}, y_j)$ and $(x_i, y_{j+\frac{1}{2}})$, respectively.

The residual vectors are then defined as follows:

$$\tilde{r}^{k+1} = A(\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{p}^{k+1}) - b(\tilde{a}^{k+1}, \tilde{\mathbf{c}}^{k+1}, \tilde{\mathbf{d}}^{k+1}, \mathbf{f}),$$
$$r^k = A(u^k, v^k, p^k) - b(a^k, \mathbf{c}^k, \mathbf{d}^k, \mathbf{f}).$$

The value of η_k that minimizes the 2-norm of

$$r^{k+1} = (1 - \eta_k)r^k + \eta_k \tilde{r}^{k+1}$$

is computed, resulting in

$$\eta_k = \frac{r^k \cdot (r^k - \tilde{r}^{k+1})}{\left\| \tilde{r}^{k+1} - r^k \right\|_2^2}. \tag{26}$$

The $(k+1)$-th iterative solution is defined to be a linear combination of the solutions with weights $1 - \eta_k$ and η_k. For example,

$$u^{k+1} = (1 - \eta_k)u^k + \eta_k \tilde{u}^{k+1}.$$

$v^{k+1}, p^{k+1}, a^{k+1}, \mathbf{c}^{k+1}$, and $\mathbf{d}^{k+1}$ are computed similarly. Because of the multi-linear property of (17), the following equation is obtained:

$$r^{k+1} = A(u^{k+1}, v^{k+1}, p^{k+1}) - b(a^{k+1}, \mathbf{c}^{k+1}, \mathbf{d}^{k+1}, \mathbf{f}). \tag{27}$$

The iterative procedure is stopped if $\left\| r^{k+1} \right\|_2$ or $\left\| (u^{k+1}, v^{k+1}, p^{k+1}) - (u^k, v^k, p^k) \right\|_\infty$ goes under the threshold.

Remark The minimum residual method (MINRES) [28] is used to solve linear system (24) for each iterative step. Because A has both positive and negative eigenvalues, incomplete Cholesky decomposition is not available. For example, given the matrix

$$M = \begin{pmatrix} -(\mu \triangle)^{\mathrm{GFM}} & 0 & 0 \\ 0 & -(\mu \triangle)^{\mathrm{GFM}} & 0 \\ 0 & 0 & \alpha I \end{pmatrix} \tag{28}$$

for positive value α, M is positive-definite, and thus we set the preconditioner of A for the MINRES method as an incomplete Cholesky decomposition of M.

3.3 Numerical methods for two-phase incompressible Navier–Stokes equation

We now introduce a new method of discretizing incompressible Navier–Stokes equations using the idea described in 3.2. As mentioned earlier, the basic framework is a semi-Lagrangian with a backward difference formula. For the Navier–Stokes equation, we assume $\mathbf{f} = -\rho \mathbf{g}$ and $\mathbf{G} = \beta \kappa \mathbf{n}$.

$$\rho_{i+\frac{1}{2},j}^{n+1} \left(\frac{u_{i+\frac{1}{2},j}^{n+1} - u_d^n}{\Delta t} \right) = (\mu \triangle u - p_x)^{n+1},$$

$$\rho_{i,j+\frac{1}{2}}^{n+1} \left(\frac{v_{i,j+\frac{1}{2}}^{n+1} - v_d^n}{\Delta t} \right) = (\mu \triangle v - p_y)^{n+1} - \rho_{i,j+\frac{1}{2}}^{n+1} g, \tag{29}$$

$$\frac{u_{i+\frac{1}{2},j}^{n+1} - u_{i-\frac{1}{2},j}^{n+1}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}}^{n+1} - v_{i,j-\frac{1}{2}}^{n+1}}{\Delta y} = 0.$$

One time-step procedure of the proposed Navier–Stokes (NS) solver is:

1. Compute the departure point and interpolate u_d^n, v_d^n.

2. Choose appropriate $\rho_{i+\frac{1}{2},j}^{n+1}, \rho_{i,j+\frac{1}{2}}^{n+1}$, and construct saddle-point system corresponding to (29) under the condition that $[\mu \nabla \mathbf{U}]^{n+1}$ is given.

3. Apply iterative procedure to solve for $\mathbf{U}^{n+1}$.

3.3.1 Discretization of convection term

The material derivative $\frac{du}{dt} = u_t + \mathbf{U} \cdot \nabla u$ is discretized using a semi-Lagrangian method. To approximate the departure point of $(x_{i+\frac{1}{2}}, y_j)$, first-order backward integration with linearized velocity is used:

$$(x_d, y_d) = (x_{i+\frac{1}{2}}, y_j) - \Delta t \left(u_{i+\frac{1}{2},j}^n, v_{i+\frac{1}{2},j}^n \right).$$
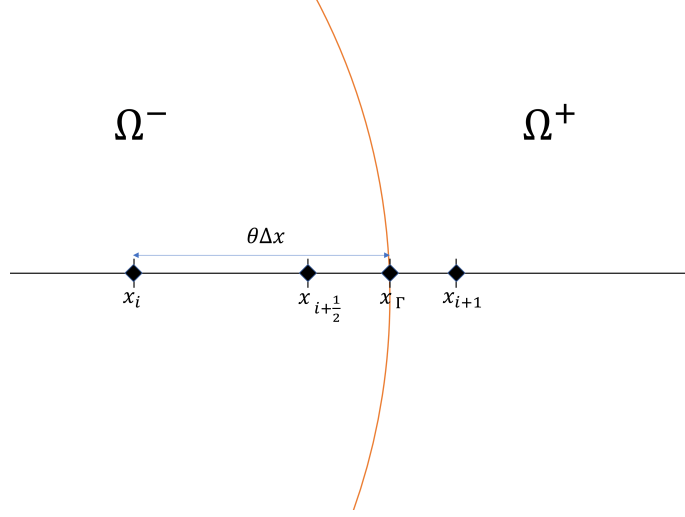
Figure 2: Visualization of the grid points near the interface when approximating p_x^-.

$v_{i+\frac{1}{2},j}^n$ is evaluated through a computation of the average v from the nearby four grid points:

$$v_{i+\frac{1}{2},j}^n = \frac{v_{i,j+\frac{1}{2}}^n + v_{i+1,j-\frac{1}{2}}^n + v_{i,j-\frac{1}{2}}^n + v_{i+1,j-\frac{1}{2}}^n}{4}.$$

u_d^n, which is an approximation of u^n at (x_d, y_d), is calculated using the quadratic interpolation defined in [25]. The approximation of v_d^n does not differ much from that of u_d^n.

3.3.2 Construction of linear system before iterative procedure

Based on the notation used in 3.2, the following values are assigned:

$$a = [p], \quad \mathbf{c} = [\mu \nabla u], \quad \mathbf{d} = [\mu \nabla v].$$

Furthermore, linear operator $\mathbf{A}$ and external force with correction term $b(a, \mathbf{c}, \mathbf{d}, f)$ are defined to be the same as those described in 3.2. Under the assumption that the jump condition $a^{n+1}, \mathbf{c}^{n+1}, \mathbf{d}^{n+1}$ is known, (29) can be discretized as

$$\begin{pmatrix} \rho I & 0 & 0 \\ 0 & \rho I & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u^{n+1} \\ v^{n+1} \\ p^{n+1} \end{pmatrix} - \begin{pmatrix} \rho u_d^n \\ \rho v_d^n \\ 0 \end{pmatrix} = \Delta t \left(\mathbf{A} \begin{pmatrix} u^{n+1} \\ v^{n+1} \\ p^{n+1} \end{pmatrix} - b(a^{n+1}, \mathbf{c}^{n+1}, \mathbf{d}^{n+1}, -\rho\mathbf{g}) \right). \tag{30}$$

The saddle-point system is obtained when this discretization is organized as follows:

$$\begin{pmatrix} \rho I - \Delta t(\mu\triangle)^{\mathrm{GFM}} & 0 & \Delta t\nabla_x^h \\ 0 & \rho I - \Delta t(\mu\triangle)^{\mathrm{GFM}} & \Delta t\nabla_y^h \\ -\Delta t\nabla_x^h & -\Delta t\nabla_y^h & 0 \end{pmatrix} \begin{pmatrix} u^{n+1} \\ v^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} \rho u_d^n \\ \rho v_d^n \\ 0 \end{pmatrix} - \Delta t b(a^{n+1}, \mathbf{c}^{n+1}, \mathbf{d}^{n+1}, -\rho\mathbf{g}). \tag{31}$$

One natural way of setting ρ is to determine its value based on whether the grid points belong to either Ω^+ or Ω^-:

$$\rho_{i+\frac{1}{2},j} = \begin{cases} \rho^- & \text{if } \phi_{i+\frac{1}{2},j}^{n+1} < 0 \\ \rho^+ & \text{if } \phi_{i+\frac{1}{2},j}^{n+1} \geq 0 \end{cases}.$$

However, our choice of ρ near the interface differs from the ρ obtained via the aforementioned method. First, the local truncation error of p_x near the interface is considered. The following assumptions are then made: $\phi_{i,j}^{n+1} < 0, \phi_{i+1,j}^{n+1} > 0$, and $\phi_{i+\frac{1}{2},j}^{n+1} < 0$. Let

$$\theta = \frac{|\phi_{i,j}^{n+1}|}{|\phi_{i,j}^{n+1}| + |\phi_{i+1,j}^{n+1}|},$$

such that $x_\Gamma = (x_i + \theta\Delta x, y_j)$ approximates the location of the interface. Figure 2 shows a visualization of these grid points and interface. p_x^- at $(x_{i+\frac{1}{2}}, y_j)$ is approximated as

$$p_x^- \approx \frac{p_{i+1,j} - [p]_\Gamma - p_{i,j}}{\Delta x}.$$

9

The value $p_R^{\pm} = p^{\pm}(x_i + \theta\Delta x, y_j)$ is then assigned. Thus,

$$\frac{p_{i+1,j} - [p]_\Gamma - p_{i,j}}{\Delta x} - p_x^- = \frac{p_R^+ + (1-\theta)p_x^+ \Delta x + -[p]_\Gamma - p_{i,j}}{\Delta x} - p_x^- + O(\Delta x)$$

$$= \frac{p_R^- - p_{i,j}}{\Delta x} + (1-\theta)p_x^+ - p_x^- + O(\Delta x)$$

$$= \theta\frac{p_R^- - p_{i,j}}{\theta\Delta x} + (1-\theta)p_x^+ - p_x^- + O(\Delta x)$$

$$= \theta p_x^- + (1-\theta)p_x^+ - p_x^- + o(\Delta x) = (1-\theta)[p_x] + O(\Delta x).$$

If the jump conditions on the Navier–Stokes equation are considered,

$$\left[\rho\frac{d\mathbf{U}}{dt}\right] = [\mu \triangle \mathbf{U} - \nabla p - \rho\mathbf{g}]$$

is determined. Afterward, when the material derivative being continuous across the interface $\left[\frac{d\mathbf{U}}{dt}\right] = 0$ is considered,

$$[\nabla p] = -[\rho]\left(\frac{d\mathbf{U}}{dt} + \mathbf{g}\right) + [\mu \triangle \mathbf{U}]$$

is obtained. In real-world simulations, $[\rho]$ dominates $[\mu\triangle\mathbf{U}]$, and thus $\rho_{i+\frac{1}{2},j}^{n+1}$ is selected to reduce the truncation error of p_x:

$$\rho_{i+\frac{1}{2},j}^{n+1} = \theta\rho^- + (1-\theta)\rho^+.$$

If the truncation error of the material derivative at $(x_{i+\frac{1}{2}}, y_j)$ is then computed,

$$\rho_{i+\frac{1}{2},j}^{n+1}\left(\frac{u_{i+\frac{1}{2},j}^{n+1} - u_d^n}{\Delta t}\right) - \rho^-\frac{du}{dt} = (\rho_{i+\frac{1}{2},j}^{n+1} - \rho^-)\frac{du}{dt} + O(\Delta x + \Delta y)$$

$$= (1-\theta)[\rho]\frac{du}{dt} + O(\Delta x + \Delta y)$$

is obtained, which cancels the local truncation error of p_x with the $[\rho]$ term. Generally, $\rho_{i+\frac{1}{2},j}^{n+1}$ is defined as

$$\rho_{i+\frac{1}{2},j}^{n+1} = \theta\rho^1 + (1-\theta)\rho^2$$

for

$$\rho^1 = \begin{cases} \rho^- \text{ if } \phi_{i,j}^{n+1} < 0 \\ \rho^+ \text{ if } \phi_{i,j}^{n+1} \geq 0 \end{cases} , \quad \rho^2 = \begin{cases} \rho^- \text{ if } \phi_{i+1,j}^{n+1} < 0 \\ \rho^+ \text{ if } \phi_{i+1,j}^{n+1} \geq 0 \end{cases}$$

and

$$\theta = \frac{|\phi_{i,j}^{n+1}|}{|\phi_{i,j}^{n+1}| + |\phi_{i+1,j}^{n+1}|}.$$

$\rho_{i,j+\frac{1}{2}}^{n+1}$ is defined similarly with few modifications.

3.3.3 Iterative method

Given that linear system (31) has been established, the iterative method from 3.2.3 can be applied. Let

$$\hat{\mathbf{A}}(u,v,p) = \begin{pmatrix} \rho I - \Delta t(\mu\triangle)^{\text{GFM}} & 0 & \Delta t\nabla_x^h \\ 0 & \rho I - \Delta t(\mu\triangle)^{\text{GFM}} & \Delta t\nabla_y^h \\ -\Delta t\nabla_x^h & -\Delta t\nabla_y^h & 0 \end{pmatrix}\begin{pmatrix} u \\ v \\ p \end{pmatrix}$$

and

$$\hat{b}(a,\mathbf{c},\mathbf{d},\rho\mathbf{g}) = \begin{pmatrix} \rho u_d^n \\ \rho v_d^n \\ 0 \end{pmatrix} - \Delta t b(a,\mathbf{c},\mathbf{d},-\rho\mathbf{g}).$$

To sum up, the velocity and pressure at time-level t^{n+1} are computed via the solution for $u^{n+1}, v^{n+1}, p^{n+1}$ of the linear system

$$\hat{\mathbf{A}}(u^{n+1}, v^{n+1}, p^{n+1}) = \hat{b}(a^{n+1}, \mathbf{c}^{n+1}, \mathbf{d}^{n+1}, -\rho\mathbf{g}), \tag{32}$$

where

$$\mathbf{c}^{n+1} = \begin{pmatrix} [\mu](2n_x^2 n_y - n_y)u_{\boldsymbol{\tau}}^{\text{ext}} & +[\mu](n_x n_y^2 - n_x^3)v_{\boldsymbol{\tau}}^{\text{ext}} \\ [\mu](2n_x n_y^2 + n_x)u_{\boldsymbol{\tau}}^{\text{ext}} & +[\mu](n_y^3 - n_x^2 n_y)v_{\boldsymbol{\tau}}^{\text{ext}} \end{pmatrix}$$
$$\mathbf{d}^{n+1} = \begin{pmatrix} [\mu](-2n_x^2 n_y - n_y)v_{\boldsymbol{\tau}}^{\text{ext}} & +[\mu](n_x n_y^2 - n_x^3)u_{\boldsymbol{\tau}}^{\text{ext}} \\ [\mu](-2n_x n_y^2 + n_x)v_{\boldsymbol{\tau}}^{\text{ext}} & +[\mu](n_y^3 - n_x^2 n_y)u_{\boldsymbol{\tau}}^{\text{ext}} \end{pmatrix} \tag{33}$$
$$a^{n+1} = -2[\mu](-n_y u_{\boldsymbol{\tau}}^{\text{ext}} + n_x v_{\boldsymbol{\tau}}^{\text{ext}}) - \beta\kappa.$$

$u_{\boldsymbol{\tau}}^{\text{ext}}, v_{\boldsymbol{\tau}}^{\text{ext}}$ are computed according to (20) with the steady-state solution of (18), where the boundary condition is given by $u_\Gamma = u^{n+1}, v_\Gamma = v^{n+1}$. Because the right-hand side of (32) also involves a linear combination of u^{n+1}, v^{n+1}, an iterative method is used to solve the given linear system. For the iterative method, the time-level subscript $n + 1$ is dropped for the sake of simplicity. k denotes the number of iterative procedures.

1. For $a^k, \mathbf{c}^k$, and $\mathbf{d}^k$, solve for $\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{p}^{k+1}$ in

$$\hat{\mathbf{A}}\left(\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{p}^{k+1}\right) = \hat{b}(a^k, \mathbf{c}^k, \mathbf{d}^k, -\rho\mathbf{g}).$$

   At the beginning of the iterative method, set $\mathbf{c}^0 = \mathbf{0}, \mathbf{d}^0 = \mathbf{0}$, and $a^0 = -\beta\kappa$.

2. Get $\tilde{u}^{\text{ext}}, \tilde{v}^{\text{ext}}$ as a steady-state solution of velocity extrapolation 3.2.2, with boundary condition $u_\Gamma = \tilde{u}^{k+1}, v_\Gamma = \tilde{v}^{k+1}$. Compute the jump condition $\tilde{a}^{k+1}, \tilde{\mathbf{c}}^{k+1}, \tilde{\mathbf{d}}^{k+1}$ using (23), where u_τ, v_τ are substituted with $\tilde{u}^{\text{ext}}, \tilde{v}^{\text{ext}}$.

3. For the two residual vectors of (32),

$$r^k = \hat{b}(a^k, \mathbf{c}^k, \mathbf{d}^k, -\rho\mathbf{g}) - \hat{\mathbf{A}}\left(u^k, v^k, p^k\right),$$
$$\tilde{r}^{k+1} = \hat{b}(\tilde{a}^{k+1}, \tilde{\mathbf{c}}^{k+1}, \tilde{\mathbf{d}}^k, -\rho\mathbf{g}) - \hat{\mathbf{A}}\left(\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{p}^{k+1}\right),$$

   and with respect to $\left(u^k, v^k, p^k\right)$ and $\left(\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{p}^{k+1}\right)$, find η_k that minimizes $\left\|(1 - \eta_k)r^k + \eta_k \tilde{r}^{k+1}\right\|_2$, which is given by (26). The $(k + 1)$-th iterative solutions $u^{k+1}, v^{k+1}, p^{k+1}$, and jump conditions $a^{k+1}, \mathbf{c}^{k+1}, \mathbf{d}^{k+1}$ are computed as described in 3.2.3, as a linear combination with weights $1 - \eta_k$ and η_k.

The aforementioned three steps are repeated until convergence occurs. Because $\hat{\mathbf{A}}$ is symmetric, we use the minimal residual (MINRES) method to solve the linear system in step 1 of the iterative method. An initial guess for the iterative method is given by the velocity and pressure at time level t^n. As a by-product of the iterative method, $u^{\text{ext}}, v^{\text{ext}}$ at time level t^{n+1} is obtained. These values are used to advect the level-set from t^{n+1} to t^{n+2}, via the definitions $u_{i,j}^{n+1} = \frac{u_{i+\frac{1}{2},j}^{\text{ext}} + u_{i-\frac{1}{2},j}^{\text{ext}}}{2}$, $v_{i,j}^{n+1} = \frac{v_{i,j+\frac{1}{2}}^{\text{ext}} + v_{i,j-\frac{1}{2}}^{\text{ext}}}{2}$.

3.3.4 Time-step restriction

In [15], time-step restrictions were computed for each convection, viscous term, gravity, and surface tension. The classical time-step restriction of [3] was not violated, but the effects of external forces and convection were summed up to produce a small time step Δt. On the other hand, our time-step restriction is based on [39], which extended the time-step restriction created by [9], which, in turn, is valid for two-phase flows involving the same densities and viscosities.

$$C_{cfl} = c_0 \frac{1}{\|\mathbf{U}\|_\infty} \Delta x$$

and

$$S_{cfl} = \frac{c_1 \mu_{\min}}{\beta} \Delta x + \sqrt{\left(c_1 \frac{\mu_{\min}}{\beta} \Delta x\right)^2 + c_2 \frac{(\rho^- + \rho^+)\Delta x^3}{4\pi\beta}}$$

are then defined as the time-step restrictions for convection and surface tension, respectively. The researchers in [39] proposed to set time-step

$$\Delta t = \min(C_{cfl}, S_{cfl})$$

with $c_1, c_2 < 1$ and $c_0 < 1$. Because our method advects the level-set with the Eulerian method, $c_0 < 0.5$ must be imposed. We use $c_0 = 0.45, c_1 = 0.9$, and $c_2 = 0.9$ in our numerical simulations.

4 Numerical Experiments

In this section, we present our numerical experiments on two-phase incompressible flows. First, we deal with analytical solutions, and then singular source and external force terms are given according to the solutions. Practical numerical examples with surface tension and gravity are then considered. All computations are performed in C++ and implemented on a personal computer with 3.40 GHz CPU and 32.0 GB memory without parallelization.

(a) $\mu^+ = 0.1, \mu^- = 0.01$

resolution	L^∞ error of $\mathbf{U}$	order	L^∞ error of p	order	L^2 error of $\mathbf{U}$	order	L^2 error of p	order
32^2	9.56.E-02		9.19.E-02		1.83.E-01		8.08.E-02	
64^2	4.67.E-02	1.03	5.42.E-02	0.76	7.04.E-02	1.37	3.12.E-02	1.37
128^2	1.68.E-02	1.48	2.41.E-02	1.17	2.03.E-02	1.79	1.02.E-02	1.61
256^2	7.46.E-03	1.17	1.59.E-02	0.59	9.34.E-03	1.12	4.55.E-03	1.17

(b) $\mu^+ = 0.01, \mu^- = 0.1$

resolution	L^∞ error of $\mathbf{U}$	order	L^∞ error of p	order	L^2 error of $\mathbf{U}$	order	L^2 error of p	order
32^2	1.35.E-01		7.61.E-02		2.78.E-01		5.48.E-02	
64^2	5.72.E-02	1.24	4.06.E-02	0.91	9.82.E-02	1.50	1.59.E-02	1.78
128^2	1.95.E-02	1.55	2.49.E-02	0.70	2.94.E-02	1.74	9.35.E-03	0.77
256^2	8.38.E-03	1.22	1.43.E-02	0.81	1.40.E-02	1.07	3.83.E-03	1.29

Table 2: Cumulative number of MINRES iterations

resolution	$\mu^+ = 0.1, \mu^- = 0.01$	$\mu^+ = 0.01, \mu^- = 0.1$
32^2	1419	1495
64^2	2638	3191
128^2	5084	6299
256^2	10038	10670

4.1 Steady-state Stokes equation

We first consider the two-phase steady-state Stokes equation $\mu \triangle \mathbf{U} - \nabla p = \mathbf{f}$ on $\Omega = [-2, 2]^2$. The interface Γ is defined by the zero level-set of $\phi = \sqrt{x^2 + y^2} - 1$, and the exact solution $\mathbf{U} = (u, v)$ and p are

$$
u = \begin{cases} \frac{y}{4} & \text{if } \phi(x,y) < 0 \\ \frac{y}{4}\left(x^2 + y^2\right) & \text{if } \phi(x,y) \geq 0 \end{cases}, \quad
v = \begin{cases} -\frac{x}{4}(1 - x^2) & \text{if } \phi(x,y) < 0 \\ -\frac{x}{4}y^2 & \text{if } \phi(x,y) \geq 0 \end{cases}, \quad
p = \begin{cases} \cos(\pi x)\cos(\pi y) + 10 & \text{if } \phi(x,y) < 0 \\ x^2 + y^2 & \text{if } \phi(x,y) \geq 0 \end{cases}.
$$

Figure 3 shows the solution profile on a 64×64 grid. The external force term $\mathbf{f} = (f_1, f_2)$ and the jump condition $\left[\mu\left(\nabla \mathbf{U} + \nabla \mathbf{U}^T\right)\mathbf{n} - p\mathbf{n}\right] = \mathbf{G} = (G_1, G_2)$ are obtained according to the exact solution:

$$
f_1 = \begin{cases} \pi \sin(\pi x)\cos(\pi y) & \text{if } \phi(x,y) < 0 \\ 2\mu^+ y - 2x & \text{if } \phi(x,y) \geq 0 \end{cases}, \quad
f_2 = \begin{cases} \frac{3\mu^- x}{2} + \pi \cos(\pi x)\sin(\pi y) & \text{if } \phi(x,y) < 0 \\ -\frac{\mu^+ x}{2} - 2y & \text{if } \phi(x,y) \geq 0 \end{cases}
$$

$$
G_1 = x\left(\cos(\pi x)\cos(\pi y) - x^2 - y^2 + 10 + \mu^+ x\,y\right) + y\left(\mu^+\left(\frac{x^2}{4} + \frac{y^2}{2}\right) - \frac{3\,\mu^-\,x^2}{4}\right),
$$

$$
G_2 = y\left(\cos(\pi x)\cos(\pi y) - x^2 - y^2 + 10 - \mu^+ x\,y\right) + x\left(\mu^+\left(\frac{x^2}{4} + \frac{y^2}{2}\right) - \frac{3\,\mu^-\,x^2}{4}\right).
$$

Numerical experiments are conducted with viscosities $\mu^+ = 0.1, \mu^- = 0.01$ or $\mu^+ = 0.01, \mu^- = 0.1$. The results of convergence for the L^∞ and L^2 norms are presented in Table 1. The estimated order of convergence for pressure in the L^∞ norms is around 0.8, but first-order convergence is observed for pressure in the L^2 norms and for velocities in both the L^∞ and L^2 norms. In Table 2, we present the cumulative numbers of MINRES iterations when preconditioners are chosen as incomplete Cholesky decompositions of M, with $\alpha = \frac{1}{\Delta x^2}$ defined as in (28). We also observe that the growth rate of the total iteration is linear to the grid size.

4.2 Analytical solution of Navier–Stokes equation with stationary interface

As a second example, a two-phase incompressible Navier–Stokes equation with stationary interface $\Gamma = \{(x, y) \in \mathbb{R}^2 \mid \sqrt{x^2 + y^2} = 1\}$ is considered on the computational domain $\Omega = [-2, 2]^2$. The exact solutions for velocity and
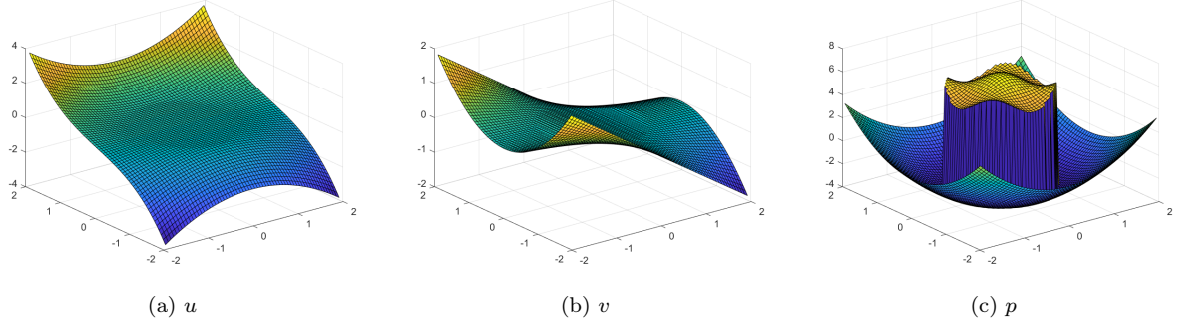
(a) u (b) v (c) p

Figure 3: Visualization of solution $\mathbf{U} = (u, v)$ and p for 4.1.

Table 3: Convergence of Example 4.2.

resolution	L^∞ error of $\mathbf{U}$	order	L^∞ error of p	order	L^2 error of $\mathbf{U}$	order	L^2 error of p	order
32^2	1.54.E-01		3.20.E-02		1.35.E-01		3.48.E-02	
64^2	8.62.E-02	0.84	1.85.E-02	0.79	7.20.E-02	0.91	1.35.E-02	1.36
128^2	3.62.E-02	1.25	9.47.E-03	0.97	2.91.E-02	1.31	6.53.E-03	1.05
256^2	1.75.E-02	1.05	4.93.E-03	0.94	1.35.E-02	1.11	3.11.E-03	1.07

pressure are determined to be

$$
u = \begin{cases} y(x^2 + y^2 - 1)\cos(t) & \text{if } \phi(x,y) < 0 \\ \left(\frac{y}{r} - y\right)\cos(t) & \text{if } \phi(x,y) \geq 0 \end{cases}, \quad
v = \begin{cases} -x(x^2 + y^2 - 1)\cos(t) & \text{if } \phi(x,y) < 0 \\ \left(-\frac{x}{r} + x\right)\cos(t) & \text{if } \phi(x,y) \geq 0 \end{cases},
$$

$$
p = \begin{cases} \cos(x)\cos(y)\cos(t) & \text{if } \phi(x,y) < 0 \\ 0 & \text{if } \phi(x,y) \geq 0 \end{cases}
$$

where $r = \sqrt{x^2 + y^2}$. The density and viscosity are chosen to be $\frac{\rho^+}{\rho^-} = \frac{\mu^+}{\mu^-} = 100$ for $\rho^+ = 1$ and $\mu^+ = 0.01$. As in 4.1, the values of $\mathbf{f}$ and $\mathbf{G}$ are determined according to the analytical solution. Furthermore, $\mathbf{U} = \mathbf{0}$ on Γ, and therefore the interface does not change over time. To check the accuracies of solution methods for the Navier–Stokes equation, we do not advect level-set in this example. Numerical simulations are performed up to $t = \pi$, and the profile of the solution is shown in Figure 4. First-order convergence for velocity and pressure both in L^∞ and L^2 norms are observed in Table 3. The results demonstrate that our method can manage the non-smoothness of solutions to two-phase Navier–Stokes equations.
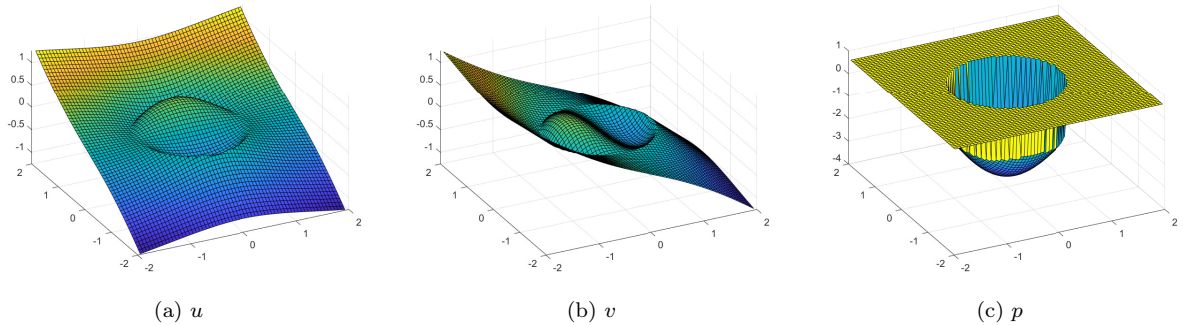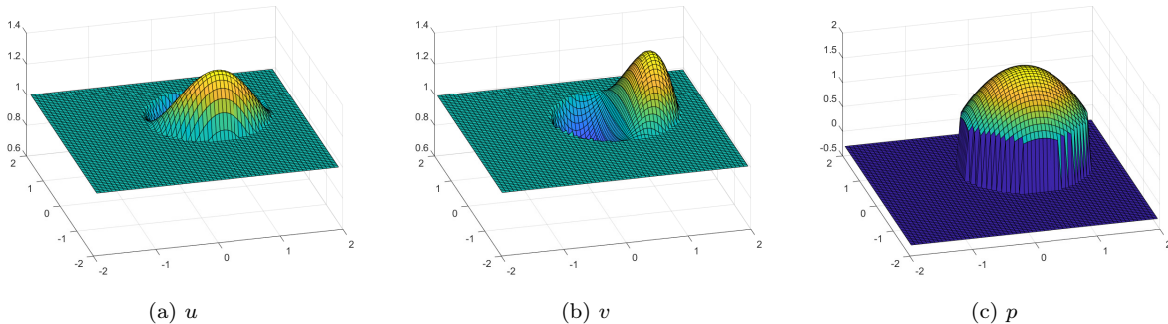

(a) u (b) v (c) p

Figure 4: Visualization of solution for Example 4.2 at $t = \pi$.

13

Table 4: Convergence of Example 4.3.

resolution	L^∞ error of $\mathbf{U}$	order	L^∞ error of p	order	L^∞ error of ϕ	order
32×32	1.22.E-01		9.81.E-02		3.58.E-02	
64×64	7.03.E-02	0.80	5.22.E-02	0.91	1.66.E-02	1.10
128×128	3.96.E-02	0.83	3.67.E-02	0.51	8.32.E-03	1.00
256×256	2.10.E-02	0.91	1.51.E-02	1.28	4.20.E-03	0.98

(a) u (b) v (c) p

Figure 5: Visualization of solution for Example 4.3 at $t = 1$.

4.3   Analytical solution of Navier–Stokes equation with moving interface

We now consider moving interface $\Gamma_t = \{(x,y) \in \mathbb{R}^2 | \sqrt{(x-t+0.5)^2 + (y-t+0.5)^2} = 1\}$ on the computational domain $\Omega = [-2,2]^2$. The exact solutions are chosen to have a constant velocity on the interface

$$u = \begin{cases} (y-t+0.5)(r^2-1)+1 & \text{if } \phi(x,y,t) < 0 \\ 1 & \text{if } \phi(x,y,t) \geq 0 \end{cases}, \quad v = \begin{cases} -(x-t+0.5)(r^2-1)+1 & \text{if } \phi(x,y,t) < 0 \\ 1 & \text{if } \phi(x,y,t) \geq 0 \end{cases},$$

$$p = \begin{cases} 2-r^2 & \text{if } \phi(x,y,t) < 0 \\ 0 & \text{if } \phi(x,y,t) \geq 0 \end{cases}$$

for $r = \sqrt{(x-t+0.5)^2 + (y-t+0.5)^2}$. The profile of the solution is shown in Figure 5. The process of checking $\mathbf{U} = (1,1)$ on Γ_t is easy, and therefore, movement of the interface Γ_t is easily found to agree with the velocity $\mathbf{U}$. Numerical simulations are conducted from $t = 0$ to $t = 1$, with material quantities $\mu^- = 0.01, \mu^+ = 0.1$ and $\rho^- = 0.1, \rho^+ = 1$. The convergence order estimates for the velocity, pressure, and interface location are observed in Table 4, where the error of the interface location is measured based on the error of the level-set function on the grid points of $|\phi_{i,j}| < 3\Delta x$. Because of the movement of the interface, the convergences of velocity and pressure are not exactly first-order. Nonetheless, we obtain first-order accuracy for the interface position.

4.4   Parasitic currents

As a fourth example, we consider a Navier–Stokes equation that accounts for surface tension in the absence of gravity. The initial interface is given as a circle, resulting in an exact solution that has zero velocity, and piece-wise constant pressure, depending on the radius of the circle and surface tension coefficient. This example is also called a parasitic current, which has been tested in [31, 35, 39]. We perform simulations up to $t = 0.5$, where the initial interface is given as $\phi = \sqrt{x^2 + y^2} - 1$ on the computational domain $\Omega = [-2,2]^2$. The following values for density, viscosity, and surface tension coefficient,

$$\rho^+ = 0.1, \quad \rho^- = 1, \quad \mu^+ = 0.01, \quad \mu^- = 0.1, \quad \beta = 50,$$

are chosen. L^∞ errors of velocity, pressure, and interface location are presented in Table 5, showing first-order convergence.

4.5   Rising bubble

Lastly, we simulate rising-bubbles problems involving strong surface tension. For each numerical simulation, we measure the rising velocity, circularity, and relative area loss of the bubble. Each of these quantities are calculated as

Table 5: Convergence of Example 4.4.

resolution	L^∞ error of $\mathbf{U}$	order	L^∞ error of p	order	L^∞ error of ϕ	order
32×32	8.77.E-02		4.72.E-01		3.67.E-03	
64×64	4.00.E-02	1.13	2.54.E-01	0.89	2.10.E-03	0.81
128×128	1.81.E-02	1.14	1.53.E-01	0.73	6.62.E-04	1.66
256×256	1.06.E-02	0.78	7.87.E-02	0.96	2.54.E-04	1.38

(a) $t = 0.02$ (b) $t = 0.035$ (c) $t = 0.05$

Figure 6: Visualization of solution for example 4.5.1.

follows:

$$\text{rising velocity} = \frac{\int_\Omega (1 - H(\phi))\mathbf{U}dx}{\pi r_0^2}, \quad \text{circularity} = \frac{2\pi r_0}{\int_\Omega \delta(\phi)dx}, \quad \text{relative area loss} = \frac{\pi r_0^2 - \int_\Omega 1 - H(\phi)dx}{\pi r_0^2}$$

r_0 is the initial radius of the bubble, and H, δ are the numerical Heaviside and delta functions, which are defined as

$$H(\phi) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1}{2} + \frac{\phi}{2\epsilon} + \frac{1}{2\pi}\sin(\frac{\pi\phi}{\epsilon}) & -\epsilon \le \phi \le \epsilon , \\ 1 & \epsilon < \phi \end{cases} \quad \delta(\phi) = \begin{cases} 0 & \phi < -\epsilon \\ \frac{1}{2\epsilon} + \frac{1}{2\epsilon}\cos(\frac{\pi\phi}{\epsilon}) & -\epsilon \le \phi \le \epsilon . \\ 0 & \epsilon < \phi \end{cases}$$

In our numerical experiments, we set $\epsilon = 1.5\Delta x$.

4.5.1   Small-air-bubble rising

First, we simulate an air bubble rising in a tank filled with water, which has been tested in [15]. Density, viscosity, surface tension coefficient, and gravity are set realistically, with the values

$$\rho^+ = 1000, \quad \rho^- = 1.226, \quad \mu^+ = 1.137 \times 10^{-3}, \quad \mu^- = 1.78 \times 10^{-5}, \quad \beta = 0.0728, \quad \mathbf{g} = (0, -9.8) .$$

The interface is initialized with level-set function $\phi(x, y, 0) = \sqrt{x^2 + y^2} - \frac{1}{300}$ on the computational domain $\Omega = [-0.01, 0.01] \times [-0.01, 0.02]$ with a no-slip boundary condition for the velocities. Calculations are performed in Cartesian grids of sizes $40 \times 60, 80 \times 120, 160 \times 240$, and 320×480. Visualizations of the air bubble at $t = 0.02, 0.035$, and 0.05 are shown in Figure 6, verifying the convergence of the interface position. Rising velocity, circularity, and relative area loss are presented in Figure 7. Relative area losses on the four different grids at $t = 0.05$ are $1.36, 0.34, 0.11$, and 0.03%, respectively. Compared to the results for GFM [15], which were $17.23, 5.76, 1.54$, and 0.0036%, the results of our air-bubble-rising experiment demonstrate a significant improvement in terms of area preservation at a low resolution.
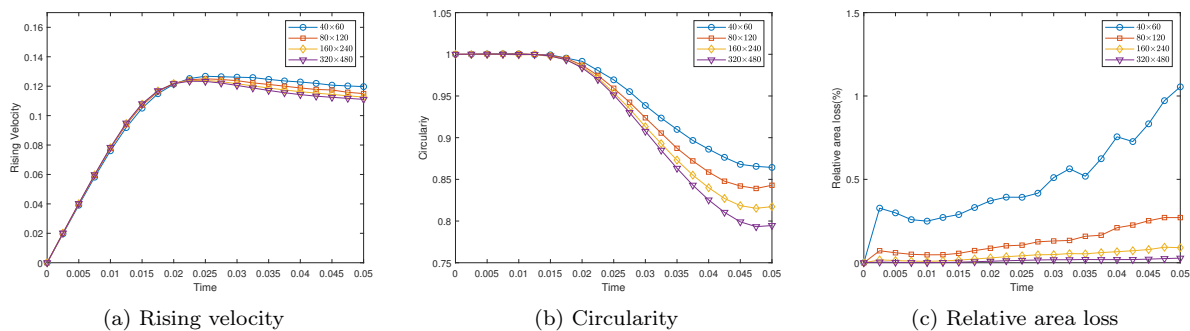
(a) Rising velocity (b) Circularity (c) Relative area loss

Figure 7: Rising velocity, circularity, and relative area loss for small air bubble in 4.5.1.

4.5.2 Two-dimensional bubble-rising benchmark test by Hysing et al. [13]

We consider case 1 of the benchmark problem proposed in [13]. Material quantities are then assigned the values

$$\rho^+ = 1000, \quad \rho^- = 100, \quad \mu^+ = 10, \quad \mu^- = 1, \quad \beta = 24.5, \quad \mathbf{g} = (0, -0.98).$$

On the computational domain $\Omega = [0, 1] \times [0, 2]$, the bubble is initialized as a circle centered at $(0.5, 1)$ with radius $r_0 = 0.25$. A no-slip boundary condition is imposed on the horizontal wall, whereas free-slip boundary condition is imposed on the vertical wall. This example is experimented on uniform Cartesian grids with sizes $40 \times 80, 80 \times 160, 160 \times 320$, and 320×640, up to $t = 3$. Visualizations of the bubble at $t = 1.5$ and 3 are shown in Figure 8. In contrast to 4.5.1, little difference exists between the shapes of the bubble as the grid is refined. Nonetheless, one can see the convergence of interface position by zooming in the results. Rising velocity, circularity, and relative area loss are shown in Figure 9. The graphs of rising velocity and circularity agree with the results in [13]. Convergence of the quantities is also verified.
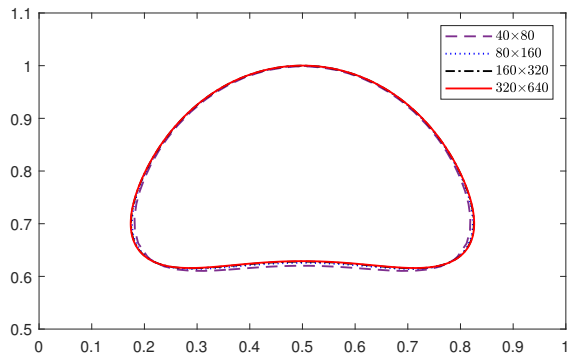
5   Conclusion

In this paper, we presented a sharp capturing method for two-phase incompressible Navier–Stokes equations. We derived jump condition formulas for pressure and velocity gradients to apply with the ghost fluid method. Together with a divergence-free condition, saddle-point systems for velocity and pressure are constructed, allowing the jump condition to be applied on the viscous term and pressure implicitly. The saddle-point system is solved via an iterative method, where each iterative step consists of solving symmetric linear systems and extrapolating interfacial velocities to nearby grid points to correct the jump conditions. The numerical experiment supports the idea that our method is first-order accurate for velocity, pressure, and interface position for analytical solution, and can be applied to practical problems.
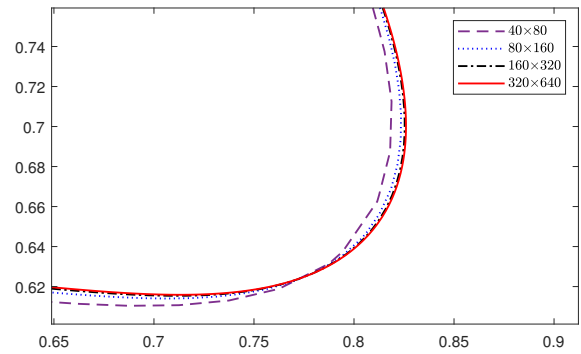
Although we proposed a preconditioner for the saddle-point problem, the construction of a more effective preconditioner for the saddle-point problem, and its detailed analysis, remain as prospects for future work. Furthermore, we will consider second-order extension of the proposed method, by applying the jump condition to the convection term and considering jump conditions for gradients of pressure.
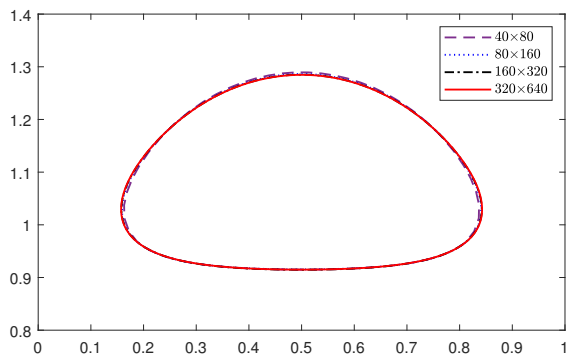
References

[1] Assêncio, D.C., Teran, J.M.: A second order virtual node algorithm for stokes flow problems with interfacial forces, discontinuous material properties and irregular domains. Journal of Computational Physics **250**, 77–105 (2013)

[2] Bedrossian, J., Von Brecht, J.H., Zhu, S., Sifakis, E., Teran, J.M.: A second order virtual node method for elliptic problems with interfaces and irregular domains. Journal of Computational Physics **229**(18), 6405–6426 (2010)

[3] Brackbill, J.U., Kothe, D.B., Zemach, C.: A continuum method for modeling surface tension. Journal of computational physics **100**(2), 335–354 (1992)

[4] Chen, X., Li, Z., Álvarez, J.R.: A direct iim approach for two-phase stokes equations with discontinuous viscosity on staggered grids. Computers & Fluids **172**, 549–563 (2018)

[5] Cho, H., Han, H., Lee, B., Ha, Y., Kang, M.: A second-order boundary condition capturing method for solving the elliptic interface problems on irregular domains. Journal of Scientific Computing **81**(1), 217–251 (2019)
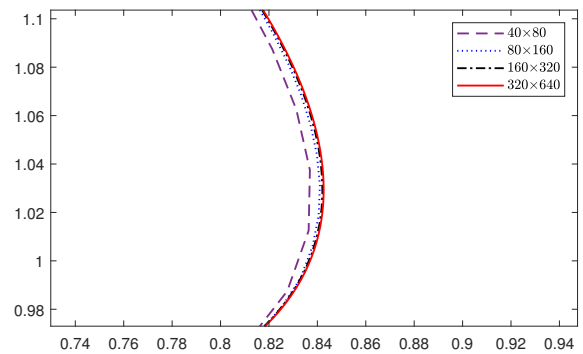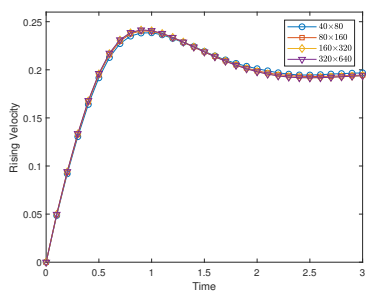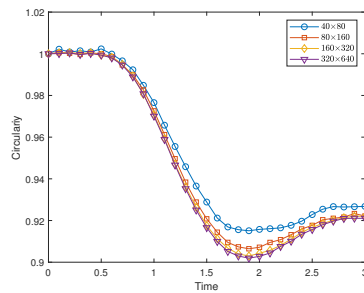
(a) $t = 1.5$

(b) $t = 1.5$(zoomed)
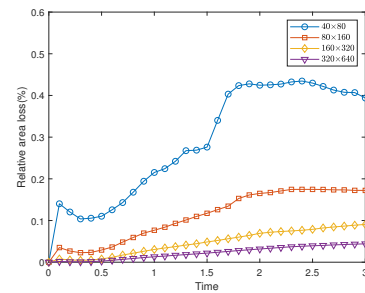
(c) $t = 3$

(d) $t = 3$(zoomed)

Figure 8: Visualization of solution for example 4.5.2.

(a) Rising velocity

(b) Circularity

(c) Relative area loss

Figure 9: Rising velocity, circularity, and relative area loss for rising bubble in 4.5.2.

[6] Chorin, A.J.: Numerical solution of the navier-stokes equations. Mathematics of computation **22**(104), 745–762 (1968)

[7] Egan, R., Gibou, F.: xgfm: Recovering convergence of fluxes in the ghost fluid method. Journal of Computational Physics p. 109351 (2020)

[8] Fedkiw, R.P., Aslam, T., Merriman, B., Osher, S.: A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). Journal of computational physics **152**(2), 457–492 (1999)

[9] Galusinski, C., Vigneaux, P.: On stability condition for bifluid flows with surface tension: Application to microfluidics. Journal of Computational Physics **227**(12), 6140–6164 (2008)

[10] Gibou, F., Fedkiw, R., Osher, S.: A review of level-set methods and some recent applications. Journal of Computational Physics **353**, 82–109 (2018)

[11] Harlow, F.H., Welch, J.E.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. The physics of fluids **8**(12), 2182–2189 (1965)

[12] Hellrung Jr, J.L., Wang, L., Sifakis, E., Teran, J.M.: A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions. Journal of Computational Physics **231**(4), 2015–2048 (2012)

[13] Hysing, S.R., Turek, S., Kuzmin, D., Parolini, N., Burman, E., Ganesan, S., Tobiska, L.: Quantitative benchmark computations of two-dimensional bubble dynamics. International Journal for Numerical Methods in Fluids **60**(11), 1259–1288 (2009)

[14] Jiang, G.S., Shu, C.W.: Efficient implementation of weighted eno schemes. Journal of computational physics **126**(1), 202–228 (1996)

[15] Kang, M., Fedkiw, R.P., Liu, X.D.: A boundary condition capturing method for multiphase incompressible flow. Journal of Scientific Computing **15**(3), 323–360 (2000)

[16] Lalanne, B., Villegas, L.R., Tanguy, S., Risso, F.: On the computation of viscous terms for incompressible two-phase flows with level set/ghost fluid method. Journal of Computational Physics **301**, 289–307 (2015)

[17] Lee, L., LeVeque, R.J.: An immersed interface method for incompressible navier–stokes equations. SIAM Journal on Scientific Computing **25**(3), 832–856 (2003)

[18] Leveque, R.J., Li, Z.: The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. SIAM Journal on Numerical Analysis **31**(4), 1019–1044 (1994)

[19] LeVeque, R.J., Li, Z.: Immersed interface methods for stokes flow with elastic boundaries or surface tension. SIAM Journal on Scientific Computing **18**(3), 709–735 (1997)

[20] Li, Z., Ito, K.: The immersed interface method: numerical solutions of PDEs involving interfaces and irregular domains, vol. 33. Siam (2006)

[21] Li, Z., Ito, K., Lai, M.C.: An augmented approach for stokes equations with a discontinuous viscosity and singular forces. Computers & Fluids **36**(3), 622–635 (2007)

[22] Li, Z., Lai, M.C.: The immersed interface method for the navier–stokes equations with singular forces. Journal of Computational Physics **171**(2), 822–842 (2001)

[23] Liu, X.D., Fedkiw, R.P., Kang, M.: A boundary condition capturing method for poisson's equation on irregular domains. Journal of computational Physics **160**(1), 151–178 (2000)

[24] Min, C.: On reinitializing level set functions. Journal of computational physics **229**(8), 2764–2772 (2010)

[25] Min, C., Gibou, F.: A second order accurate level set method on non-graded adaptive cartesian grids. Journal of Computational Physics **225**(1), 300–321 (2007)

[26] Osher, S., Fedkiw, R., Piechor, K.: Level set methods and dynamic implicit surfaces. Appl. Mech. Rev. **57**(3), B15–B15 (2004)

[27] Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. Journal of computational physics **79**(1), 12–49 (1988)

[28] Paige, C.C., Saunders, M.A.: Solution of sparse indefinite systems of linear equations. SIAM journal on numerical analysis **12**(4), 617–629 (1975)

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

[30] Saye, R.: Implicit mesh discontinuous galerkin methods and interfacial gauge methods for high-order accurate interface dynamics, with applications to surface tension dynamics, rigid body fluid–structure interaction, and free surface flow: Part i. Journal of Computational Physics **344**, 647–682 (2017)

[31] Schroeder, C., Stomakhin, A., Howes, R., Teran, J.M.: A second order virtual node algorithm for navier–stokes flow problems with interfacial forces and discontinuous material properties. Journal of Computational Physics **265**, 221–245 (2014)

[32] Shu, C.W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. Journal of computational physics **77**(2), 439–471 (1988)

[33] Sussman, M., Fatemi, E., Smereka, P., Osher, S.: An improved level set method for incompressible two-phase flows. Computers & Fluids **27**(5-6), 663–680 (1998)

[34] Sussman, M., Smereka, P., Osher, S.: A level set approach for computing solutions to incompressible two-phase flow. Journal of Computational physics **114**(1), 146–159 (1994)

[35] Sussman, M., Smith, K.M., Hussaini, M.Y., Ohta, M., Zhi-Wei, R.: A sharp interface method for incompressible two-phase flows. Journal of computational physics **221**(2), 469–505 (2007)

[36] Tan, Z., Le, D.V., Li, Z., Lim, K.M., Khoo, B.C.: An immersed interface method for solving incompressible viscous flows with piecewise constant viscosity across a moving elastic membrane. Journal of Computational Physics **227**(23), 9955–9983 (2008)

[37] Tan, Z., Le, D.V., Lim, K.M., Khoo, B.: An immersed interface method for the incompressible navier–stokes equations with discontinuous viscosity across the interface. SIAM Journal on Scientific Computing **31**(3), 1798–1819 (2009)

[38] Tan, Z., Lim, K., Khoo, B.: An implementation of mac grid-based iim-stokes solver for incompressible two-phase flows. Communications in Computational Physics **10**(5), 1333–1362 (2011)

[39] Theillard, M., Gibou, F., Saintillan, D.: Sharp numerical simulation of incompressible two-phase flows. Journal of Computational Physics **391**, 91–118 (2019)

[40] Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., Jan, Y.J.: A front-tracking method for the computations of multiphase flow. Journal of computational physics **169**(2), 708–759 (2001)

[41] Unverdi, S.O., Tryggvason, G.: A front-tracking method for viscous, incompressible, multi-fluid flows. Journal of computational physics **100**(1), 25–37 (1992)