Journal of Computational Physics 265 (2014) 221–245

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

A second order virtual node algorithm for Navier–Stokes flow problems with interfacial forces and discontinuous material properties

Craig Schroeder *, Alexey Stomakhin, Russell Howes, Joseph M. Teran

University of California, Los Angeles, Department of Mathematics, 520 Portola Plaza, Math Sciences Building 6363, Los Angeles, CA 90095, United States

ARTICLE INFO

Article history:
Received 18 June 2013
Received in revised form 20 January 2014
Accepted 27 January 2014
Available online 11 February 2014

Keywords:
Navier Stokes
Virtual node algorithms
Interface problems
Cartesian grids

ABSTRACT

We present a numerical method for the solution of the Navier–Stokes equations in three dimensions that handles interfacial discontinuities due to singular forces and discontinuous fluid properties such as viscosity and density. We show that this also allows for the enforcement of normal stress and velocity boundary conditions on irregular domains. The method improves on results in [1] (which solved the Stokes equations in two dimensions) by providing treatment of fluid inertia as well as a new discretization of jump and boundary conditions that accurately resolves null modes in both two and three dimensions. We discretize the equations using an embedded approach on a uniform MAC grid to yield discretely divergence-free velocities that are second order accurate. We maintain our interface using the level set method or, when more appropriate, the particle level set method. We show how to implement Dirichlet (known velocity), Neumann (known normal stress), and slip velocity boundary conditions as special cases of our interface representation. The method leads to a discrete, symmetric KKT system for velocities, pressures, and Lagrange multipliers. We also present a novel simplification to the standard combination of the second order semi-Lagrangian and BDF schemes for discretizing the inertial terms. Numerical results indicate second order spatial accuracy for the velocities (L^∞ and L^2) and first order for the pressure (in L^∞, second order in L^2). Our temporal discretization is also second order accurate.

1. Introduction

The simulation of multiphase incompressible flow in arbitrary domains is necessary for many applications in computational physics and engineering. Unfortunately, it is particularly difficult to attain orders of accuracy easily achievable in the case of uniform or periodic domains. Due to irregular interface and domain boundary geometries, a natural approach to the numerical approximation of the equations is the finite element method (FEM) with unstructured meshes that conform to the irregular geometry. However, meshing complex interface geometries can prove difficult and time-consuming when the interface frequently changes. We have recently developed a class of embedded methods that utilize uniform Cartesian grids and sub-cell representations of interface/boundary geometry to achieve optimal accuracy without the need for frequent

* Corresponding author.
E-mail addresses: craig@math.ucla.edu (C. Schroeder), alexey@math.ucla.edu (A. Stomakhin), rhowes@math.ucla.edu (R. Howes), jteran@math.ucla.edu (J.M. Teran).

http://dx.doi.org/10.1016/j.jcp.2014.01.051
0021-9991/

remeshing [1–4]. Our use of regular grids simplifies the implementation, permits straightforward numerical linear algebra and naturally allows for higher order accuracy in L^∞. We have used the term virtual node methods to describe these techniques since they utilize additional structured degrees that are outside the domain of interest. In the present work, we introduce a new virtual node method for approximating the two-phase Navier–Stokes equations with irregular embedded interfaces and boundaries on a uniform Cartesian Marker and Cell (MAC) grid, where velocity degrees of freedom are located at face centers and pressure degrees of freedom are located at cell centers.

As in [1], we duplicate Cartesian grid cells along the interface Γ to introduce additional virtual nodes that accurately resolve discontinuous quantities. This naturally treats discontinuities in material properties such as viscosity and density. Interface cells are cut and duplicated using a level set that allows for accurate evaluation of integrals needed for the numerical stencils. These stencils (for the viscous stress forces as well as the divergence-free and jump constraints) are constructed from a variational formulation that yields a symmetric linear system. This approach requires the introduction of a Lagrange multiplier variable to maintain continuity of the fluid velocity across the interface. Unfortunately, the introduction of this variable forced the approach in [1] to require that interface domain geometry have a constant normal on each MAC grid cell. Although it is a reasonable restriction in two dimensions, this is not possible in three dimensions and so the method was fundamentally limited to 2D. We present an improved discretization of this Lagrange multiplier term that works naturally in both two and three dimensions without the restriction of a constant normal per MAC grid cell. The necessity of this in [1] was due to the requirement that the discretization resolve null modes in the variational form of the equations exactly. Failure to do this resulted in significant degradation in performance. We show that our new discretization also captures these modes exactly.

We also consider a simplification to the combination of the second order Backward Difference Formula (BDF) and second order semi-Lagrangian schemes that are often used in second order Navier–Stokes discretizations to calculate the intermediate velocity field [5]. This simplification reduces the number of semi-Lagrangian interpolation steps required from four to two while retaining the temporal and spatial accuracy of the original method. The interface is evolved using the level set method or, when more appropriate, the particle level set method. Numerical experiments indicate second order accuracy in L^∞ and L^2 for the velocity and first order accuracy in L^∞, second in L^2, for pressure. Numerical experiments indicate a stability restriction on the *minimum* time step size (relative to the grid spacing) that may be taken by our method in the case of a Navier–Stokes discretization. We explore the nature and source of this restriction further.

2. Existing methods

In our discussion of existing approaches, we will focus only on embedded (or immersed) methods that avoid unstructured meshing when addressing boundary and interface conditions at irregular geometric boundaries. Embedded methods place an irregular domain, or a domain with an interface, into a rectangular computational domain with a Cartesian grid. A good review of such methods is given by Lew et al. in [6]. A classic embedded method is the Immersed Boundary Method (IBM) developed by Peskin [7–12] originally to simulate blood flow in the heart. The IBM uses regularized delta functions to represent singular forces acting on interfaces. This renders the method first order accurate in general for thin interfaces, implying that the physical characteristics of the flow near those interfacial boundaries are not accurately captured [13]. For interfaces with a nonzero thickness, modifications to the IBM can yield second-order accuracy [14]. The original IBM also featured poor volume conservation near the interface, motivating the development of a volume-conserving version in [15]. However, the IBM has proven very useful for many applications.

Many methods have been developed to improve on the performance of the IBM. Mittal and collaborators have shown that a discrete forcing (rather than one first applied to the continuous equations and then discretized) can be used to get second order accuracy for flows in irregular domains [16–19]. The Immersed Interface Method (IIM) [13,20–23] is a popular example that attains second order accuracy in L^∞ by modifying the numerical stencil near the interface, and by using jump conditions instead of regularized delta functions to relate the singular forces to interfacial discontinuities in pressure, velocities and their derivatives. The IIM has been used in simulating interfaces between fluids with different viscosities [24–27] and has been extended to higher-order implementations [28,29]. The IIM is considerably more difficult to implement than the IBM and most applications are in two space dimensions as a result. However, researchers have applied the IIM to three-dimensional flows [30]. Recent IIM approaches use adaptive grid techniques near the interface to maintain high resolution near the important parts of the boundary while reducing the overall degrees of freedom [31]. In general, the IIM yields non-symmetric linear systems, and therefore requires the use of solvers such as GMRES or BiCG-STAB. Some implementations of IIM [32] yield symmetric positive definite systems, however this is only possible when the viscosity is continuous across the interface. The Matched Interface and Boundary (MIB) method [29,33] adjusts the approach of the IIM with dimension-by-dimension modifications that utilize fictitious points. Enforcement of jump conditions is decoupled from the modified finite difference stencil. The work of [34] applies the framework of the MIB to interfacial flow in two dimensions.

Our approach was initially motivated by The Ghost Fluid Method (GFM). However, because that method does not in the end introduce any additional degrees of freedom into the discretization, we used the term from another of Fedkiw's methods [35] that does similarly introduce virtual degrees of freedom. Notably, the GFM always guarantees a symmetric discretization. Initially applied to the Poisson equation with interfacial jumps and variable coefficients [36], the GFM was also used to simulate multiphase incompressible flow in [37]. Unfortunately, the GFM is only capable of achieving first order

results for interface problems. Also, in [37] the GFM treats viscous terms explicitly because they cannot be decoupled in the case of discontinuous viscosity.

Some of the first embedded methods were fictitious domain methods by Hyman [38] and Saul'ev [39]. The fictitious domain approach has been used with incompressible materials in a number of works [40–48]. These approaches embed the irregular geometry in a simpler domain for which fast solvers exist (e.g. Fast Fourier Transforms). The calculations include fictitious material in the complement of the domain of interest. A forcing term (often from a Lagrange multiplier) is used to maintain boundary conditions at the irregular geometry. Although these techniques naturally allow for efficient solution procedures, they depend on a smooth solution across the embedded domain geometry for optimal accuracy, which is not typically possible.

The extended finite element method (XFEM) and related approaches in the finite element literature also make use of geometry embedded in regular elements. Although originally developed for crack-based field discontinuities in elasticity problems, these techniques are also used with embedded problems in irregular domains. Daux et al. first showed that these techniques can naturally capture embedded Neumann boundary conditions [49,50]. These approaches are equivalent to the variational cut cell method of Almgren et al. in [51]. Enforcement of Dirichlet constraints is more difficult with variational cut cell approaches [52,6] and typically involves a Lagrange multiplier or stabilization. Dolbow and Devan recently investigated the convergence of such approaches with incompressible materials and point out that much analysis in this context remains to be completed [53]. Despite the lack of thorough analysis, such XFEM approaches appear to be very accurate and have been used in many applications involving incompressible materials in irregular domains [54–58].

There are also a handful of highly accurate embedded finite difference (FDM) and finite volume methods (FVM) utilizing cut uniform grid cells which have been developed in the context of incompressible flow for irregular domains, although these methods are not applicable to interfacial flows. For example, Marella et al. [59] use collocated grids and define sub cell interface and boundary geometry in cut cells via level sets, and achieve second order accuracy in two and three dimensions. Ng et al. also use level set descriptions of the irregular domain and achieve a symmetric positive definite discretization with second order accuracy in L^∞ for flows in two [60] and three [61] dimensions.

3. Numerical method

We consider the Navier–Stokes equations over an irregular domain $\Omega = \Omega^+ \cup \Omega^-$ with boundary $\partial\Omega = \partial\Omega_d \cup \partial\Omega_n \cup \partial\Omega_s$, where Dirichlet velocity constraints are enforced at $\partial\Omega_d$, Neumann boundary conditions at $\partial\Omega_n$, and slip conditions at $\partial\Omega_s$. The two subdomains Ω^+ and Ω^- are separated by an interface Γ, which is typically a co-dimension one closed surface. The corresponding equations are

$$\rho \boldsymbol{u}_t + \rho(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = \nabla\cdot\sigma + \boldsymbol{f}, \quad \boldsymbol{x}\in\Omega\backslash\Gamma, \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \quad \boldsymbol{x}\in\Omega\backslash\Gamma, \tag{2}$$

$$[\boldsymbol{u}] = \boldsymbol{a}_i, \quad \boldsymbol{x}\in\Gamma, \tag{3}$$

$$[\sigma\cdot\boldsymbol{n}] = \hat{\boldsymbol{f}}, \quad \boldsymbol{x}\in\Gamma, \tag{4}$$

$$\boldsymbol{u} = \boldsymbol{b}, \quad \boldsymbol{x}\in\partial\Omega_d, \tag{5}$$

$$\sigma\cdot\boldsymbol{n} = \hat{\boldsymbol{g}}, \quad \boldsymbol{x}\in\partial\Omega_n, \tag{6}$$

$$\boldsymbol{n}\cdot\boldsymbol{u} = \boldsymbol{n}\cdot\boldsymbol{c}, \quad \boldsymbol{x}\in\partial\Omega_s, \tag{7}$$

$$\left(\boldsymbol{I}-\boldsymbol{n}\boldsymbol{n}^T\right)\cdot\sigma\cdot\boldsymbol{n} = \left(\boldsymbol{I}-\boldsymbol{n}\boldsymbol{n}^T\right)\hat{\boldsymbol{h}}, \quad \boldsymbol{x}\in\partial\Omega_s \tag{8}$$

where the stress is $\sigma = \mu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T) - p\boldsymbol{I}$, $\boldsymbol{a}_i$ describes velocity jumps at interfaces, $\boldsymbol{b}$ describes velocities at Dirichlet boundaries, $\boldsymbol{c}$ represents slip velocities, $\hat{\boldsymbol{f}}$ describes interface forces, $\hat{\boldsymbol{g}}$ describes Neumann boundary conditions, and $\hat{\boldsymbol{h}}$ describes tangential stresses for slip boundary conditions. Although for physical problems the velocity jump is equal to zero (representing continuity of the velocity), we include a velocity jump which may be nonzero in our formulation of the interface. This is convenient not only for testing our implementation, but for handling the other types of boundary conditions. We take the standard convention of defining $[\boldsymbol{u}] = \boldsymbol{u}^+ - \boldsymbol{u}^-$ for interface jumps and $\boldsymbol{n}$ as the normal pointing outward from Ω^-. The interface Γ and boundary pieces $\partial\Omega_d$, $\partial\Omega_n$, and $\partial\Omega_s$ are assumed to be smooth and not intersect one another. This layout is illustrated in Fig. 1. We do not consider triple junctions in this paper.

Our method extends the framework of [1] from two to three spatial dimensions and from the Stokes equations to the full Navier–Stokes equations. As with [1], our method is second order in L^∞ and L^2 for the velocities and first order in L^∞ for the pressure (second order in L^2) for embedded interfacial discontinuities and irregular boundaries in two-phase flows, though unlike the previous work, we are also second order in time. Our method produces sparse and symmetric indefinite KKT-type linear systems. Furthermore, our approach yields discretely divergence-free velocities. We do not require knowledge of jumps on the fluid variables and their derivatives but rather only of expressions for the interfacial forces.

Our numerical method uses an Eulerian representation of the fluid velocity and pressure. A level set ϕ represents the domain Ω and interface Γ at each time step. We split the Navier–Stokes equation (1) into two equations by introducing

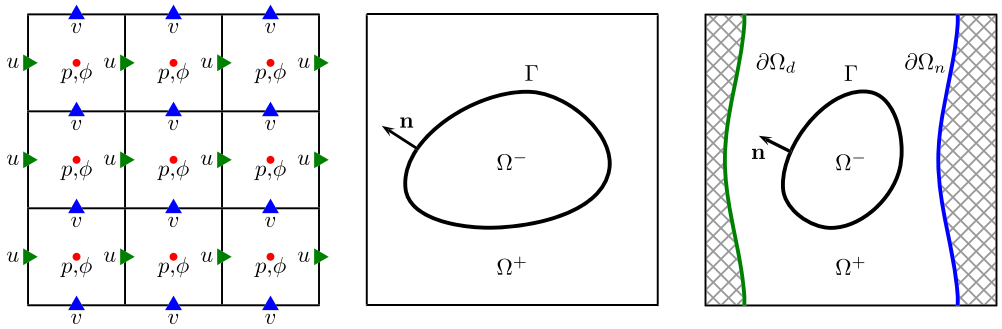


Fig. 1. (a) MAC layout in two dimensions. The red dots indicate location of pressure and level set variables, the green and blue triangles represent horizontal (u) and vertical (v) variables respectively. (b) Interface Γ separates the fluid domain $\Omega = \Omega^- \cup \Omega^+ \cup \Gamma$. In this case all sides of the computational domain have a periodic boundary condition applied. (c) Another layout which our method can handle in practice. Here an embedded Dirichlet boundary condition is applied on the left side of the domain Ω and a Neumann boundary condition on the right. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

an intermediate velocity $\boldsymbol{u}^\star$. We use a weak form of the Navier–Stokes interface problem and simultaneously solve for the pressure, velocity, and Lagrange multipliers needed to preserve the specified velocity jumps across the interface. The complete procedure for advancing one time step is:

1. Update level set $\phi^n \to \phi^{n+1}$.
2. Advect time $n-1$ and n velocity.
3. Compute intermediate fluid velocity $\boldsymbol{u}^\star$ using BDF.
4. Setup symmetric system and right hand side.
5. Solve the linear system to obtain p and $\boldsymbol{u}^{n+1}$.

We will describe each of these steps in necessary detail, following which we will discuss how to add surface tension forces, and present requirements for the stability of our method.

3.1. Spatial discretization

We use a standard MAC layout for our degrees of freedom, with velocity degrees of freedom at faces and pressures at cell centers, as shown in Fig. 1(a). We maintain our level set at cell centers.

Fluid advection and the implicit update at the end both require valid ghost data in a narrow band around the region where each fluid phase is defined. We accommodate this by storing a separate velocity array for each fluid phase for $\boldsymbol{u}^{n-1}$ and $\boldsymbol{u}^n$, which over the course of a time step we update to $\boldsymbol{u}^n$ and $\boldsymbol{u}^{n+1}$. At each MAC face, only one of the velocity arrays is considered to hold a real velocity degree of freedom, based on the sign of the level set interpolated to that face. The other array is treated as ghost data and is populated as needed using the extrapolation of [62]. Note that across interfaces the velocity field, though physically continuous, may have a jump in its derivatives. In our case, we also included support for velocity discontinuities, since this makes analytic tests far easier to construct and thus the method itself easier to test and debug.

In practice, we found using the level set directly to distinguish real and ghost velocities to be too unreliable and on a few occasions lead to the use of invalid velocity data. We avoid these problems by explicitly storing which, if any, velocity degree of freedom is valid at each MAC face, both for $\boldsymbol{u}^{n-1}$ and $\boldsymbol{u}^n$. Since we do this, there is no need to maintain a level set other than the one at the current time.

3.2. Update level set

We discretize our momentum equation at time t^{n+1}. Setting up the intermediate $\boldsymbol{u}^\star$ requires that we know which fluid region is valid at each face, which is determined by ϕ^{n+1}. Thus, we begin our time step by updating our level set $\phi^n \to \phi^{n+1}$. We use the level set method for this task in most of our examples. For examples where the level set method loses volume too quickly, we use the more expensive but more accurate particle level set method [63] instead.

In the case of the level set method, there are two steps: advection and reinitialization. We advect our level set using third order Runge–Kutta [64] in time and fifth order HJ-WENO [65,66] in space. This update requires an estimate for $\boldsymbol{u}^n$, $\boldsymbol{u}^{n+1}$, and then $\boldsymbol{u}^{n+\frac{1}{2}}$. To obtain these, we merge our per-phase velocity fields into a single velocity field by selecting the non-virtual degree of freedom at each MAC face (as determined by the sign of ϕ). We will call these merged velocities $\boldsymbol{u}_m^{n-1}$ and $\boldsymbol{u}_m^n$. Our velocity estimates for the level set advection are then $\boldsymbol{u}_m^n$, $2\boldsymbol{u}_m^n - \boldsymbol{u}_m^{n-1}$, and $\frac{3}{2}\boldsymbol{u}_m^n - \frac{1}{2}\boldsymbol{u}_m^{n-1}$. Note that our velocities do not live at the same locations as our level set, so interpolation is required to co-locate them prior to advection. For reinitialization we use also use third order Runge–Kutta and fifth order HJ-WENO.

When we use the particle level set method, we perform advection and reinitialization the same way we do for the level set method. In addition to this, the particle level set method also does particle evolution, for which we use third order Runge–Kutta and the same velocity estimates as for the level set advection step.

3.3. Discretization of inertial terms

Following [5], we use a variant on semi-Lagrangian for advection and a BDF discretization for the time derivative. This transforms (1) into

$$\rho\frac{3\boldsymbol{u}^{n+1}-4\boldsymbol{u}_d^n+\boldsymbol{u}_d^{n-1}}{2\Delta t}=\nabla\cdot\sigma+\boldsymbol{f}, \tag{9}$$

where $\boldsymbol{u}_d^n=\boldsymbol{u}^n(\boldsymbol{x}^n)$ and $\boldsymbol{u}_d^{n-1}=\boldsymbol{u}^{n-1}(\boldsymbol{x}^{n-1})$ indicate that the $\boldsymbol{u}^n$ and $\boldsymbol{u}^{n-1}$ velocities are evaluated at the departure locations obtained by tracing the position back along the characteristics of the fluid flow from the face located at $\boldsymbol{x}^{n+1}$.

Note that if we ignore advection, $\frac{3\boldsymbol{u}^{n+1}-4\boldsymbol{u}^n+\boldsymbol{u}^{n-1}}{2\Delta t}$ is a backward difference formula (BDF) for the time derivative, which is second order at time t^{n+1}. By contrast, the central difference approximation $\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^{n-1}}{2\Delta t}$ is second order accurate at time t^n and would be suitable if we chose to discretize our momentum equation at time t^n. This would naturally lead to an explicit update rule, which would not have suitable stability characteristics. Discretizing at time t^{n+1} using the BDF rule leads to an implicit method.

By introducing an intermediate velocity $\boldsymbol{u}^\star$ we split the Navier–Stokes equations into the two separate equations:

$$\rho\frac{3\boldsymbol{u}^\star-4\boldsymbol{u}_d^n+\boldsymbol{u}_d^{n-1}}{2\Delta t}=\boldsymbol{0}, \tag{10}$$

in which we apply the inertial terms to an intermediate velocity, and

$$\alpha\left(\boldsymbol{u}^{n+1}-\boldsymbol{u}^\star\right)=\nabla\cdot\sigma+\boldsymbol{f}, \tag{11}$$

where we have introduced $\alpha=\frac{3\rho}{2\Delta t}$ for convenience accommodating the first time step. We discuss the first time step later in this section.

Standard discretization To obtain second order temporal and spatial accuracy, [5] computes the advected $\boldsymbol{u}_d^n$ and $\boldsymbol{u}_d^{n-1}$ using

$$\boldsymbol{u}_d^n=\boldsymbol{u}^n\left(\boldsymbol{x}^{n+1}-\Delta t\boldsymbol{u}^{n+\frac{1}{2}}\left(\boldsymbol{x}^{n+1}-\frac{1}{2}\Delta t\boldsymbol{u}^n\left(\boldsymbol{x}^{n+1}\right)\right)\right), \tag{12}$$

where $\boldsymbol{u}^{n+\frac{1}{2}}=\frac{3}{2}\boldsymbol{u}^n-\frac{1}{2}\boldsymbol{u}^{n-1}$, and

$$\boldsymbol{u}_d^{n-1}=\boldsymbol{u}^{n-1}\left(\boldsymbol{x}^{n+1}-2\Delta t\boldsymbol{u}^n\left(\boldsymbol{x}^{n+1}-\Delta t\boldsymbol{u}^n\left(\boldsymbol{x}^{n+1}\right)\right)\right). \tag{13}$$

Here, the position $\boldsymbol{x}^{n+1}$ refers to a location where a velocity degree of freedom in $\boldsymbol{u}^{n+1}$ will be required, which in the case of our MAC discretization corresponds to the center of a face.

To see what this update is doing, consider the update for $\boldsymbol{u}_d^n$. In a semi-Lagrangian discretization, an advected velocity degree of freedom is computed at any desired location by looking upstream from that point to the find the velocity of the region of fluid that will flow to the desired location during the current time step. A simple implementation of this uses some estimate of the velocity at the current point, normally $\boldsymbol{u}^n(\boldsymbol{x}^{n+1})$. If we assume the fluid ending up at location $\boldsymbol{x}^{n+1}$ flowed with this velocity for the entire time step, it must have originated at location $\hat{\boldsymbol{x}}^n=\boldsymbol{x}^{n+1}-\Delta t\boldsymbol{u}^n(\boldsymbol{x}^{n+1})$. Thus, we would compute $\boldsymbol{u}_d^n=\boldsymbol{u}^n(\hat{\boldsymbol{x}}^n)$. This update is only a first order accurate approximation for $\boldsymbol{u}_d^n$, even if quadratic interpolation is used. The reason for this is that $\boldsymbol{u}^n(\boldsymbol{x}^{n+1})$ is not an accurate enough estimate of how the fluid flowed over the course of the time step. A better approximation can be obtained using a sort of midpoint approximation, where we estimate how the fluid was moving half-way through the time step. To do this, we need an estimate $\hat{\boldsymbol{x}}^{n+\frac{1}{2}}$ of where the fluid was half-way through the time step and what the velocity field looked like at that time, $\boldsymbol{u}^{n+\frac{1}{2}}$. The location we can approximate as before, $\hat{\boldsymbol{x}}^{n+\frac{1}{2}}=\boldsymbol{x}^{n+1}-\frac{1}{2}\Delta t\boldsymbol{u}^n(\boldsymbol{x}^{n+1})$. A suitable estimate for $\boldsymbol{u}^{n+\frac{1}{2}}$ can be obtained by extrapolating to that time from $\boldsymbol{u}^{n-1}$ and $\boldsymbol{u}^n$. This gives us a better estimate $\hat{\boldsymbol{u}}^{n+\frac{1}{2}}=\boldsymbol{u}^{n+\frac{1}{2}}(\hat{\boldsymbol{x}}^{n+\frac{1}{2}})$ for the flow over the time step of the fluid that ends up at $\boldsymbol{x}^{n+1}$. New we get a new estimate for the departure point $\hat{\boldsymbol{x}}^n=\boldsymbol{x}^{n+1}-\Delta t\hat{\boldsymbol{u}}^{n+\frac{1}{2}}$ and the corresponding improved estimate $\boldsymbol{u}_d^n=\boldsymbol{u}^n(\hat{\boldsymbol{x}}^n)$. This is equivalent to (12). The formula for (13) is obtained similarly. Note that this scheme follows the same general pattern as the classical second order Runge–Kutta method.

These formulas require three velocity evaluations each. The innermost, $\boldsymbol{u}^n(\boldsymbol{x}^{n+1})$, is evaluated at a face, so one component is obtained by a simple lookup and the remaining components are obtained by linear interpolation. Note that the grid is not moving, so the velocity samples in $\boldsymbol{u}^{n+1}$, $\boldsymbol{u}^n$, and $\boldsymbol{u}^{n-1}$ are all stored at the same face-center locations in the MAC grid. The

middle evaluation requires interpolation, which can be linear interpolation since it will be multiplied by an extra factor of Δt. The outermost velocity evaluation also requires interpolation, but this time quadratic interpolation is required to obtain second order spatial accuracy.

Simplified discretization Taylor series analysis of the advection and BDF process reveals that the standard approach to computing $\boldsymbol{u}_d^n$ and $\boldsymbol{u}_d^{n-1}$ is more expensive than necessary. In particular, it suffices to use

$$\boldsymbol{u}_d^n = \boldsymbol{u}^n\big(\boldsymbol{x}^{n+1} - \Delta t \boldsymbol{u}^n\big(\boldsymbol{x}^{n+1}\big)\big) \tag{14}$$

$$\boldsymbol{u}_d^{n-1} = \boldsymbol{u}^{n-1}\big(\boldsymbol{x}^{n+1} - 2\Delta t \boldsymbol{u}^{n-1}\big(\boldsymbol{x}^{n+1}\big)\big). \tag{15}$$

That is, the application of inertial terms simplifies down to doing a step of semi-Lagrangian advection on $\boldsymbol{u}^n$ and $\boldsymbol{u}^{n-1}$ and then computing $\boldsymbol{u}^\star$ as a linear combination of the results using (10). Normally, using semi-Lagrangian advection would only be expected to produce first order temporal accuracy, but using it in combination with BDF in this way yields second order temporal accuracy. Note that the interaction of semi-Lagrangian advection and BDF does not improve spatial accuracy, so quadratic interpolation is still required for the semi-Lagrangian advection steps. We use this simplified discretization in all of our numerical examples.

First step Since BDF is a multistep method, we need a way to take the first step. We can afford one time step with one order lower, so we simply use a backward Euler discretization, which leads to the alternate formula

$$\rho \frac{\boldsymbol{u}^\star - \boldsymbol{u}_d^n}{\Delta t} = \boldsymbol{0} \tag{16}$$

for computing $\boldsymbol{u}^\star$, where $\alpha = \frac{\rho}{\Delta t}$ is used in (11) and $\boldsymbol{u}_d^n$ is computed as in (14).

3.4. Discretization of implicit terms

We follow the derivation put forth in [1] to discretize the implicit portion of our splitting. From (11), the continuity equation, and the boundary conditions associated with the problem, we derive the weak form, which yields our discrete stencils for the velocity and fluid variables. We continue by detailing aspects of our discretization necessary to account for possible null modes of the linear system, and to admit Dirichlet, Neumann, and slip boundary conditions. We then discuss our implementation of computing the integrals necessary to obtain the discrete stencils in our discretization. In Appendix A, we assume for now that we have an interface at Γ but no other non-periodic boundaries. We will discuss other boundary conditions in Section 3.4.4.

3.4.1. Stiffness matrix

The weak form of (11), as derived in Appendix A, is given by the equations

$$\begin{aligned}\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u}\,dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}\big(\nabla\boldsymbol{w} + \nabla\boldsymbol{w}^T\big) : \big(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\big)\,dV - \int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{w}\,dV + \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\,dA \\ = \int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u}^\star\,dV + \int_{\Omega\setminus\Gamma} \boldsymbol{w}\cdot\boldsymbol{f}\,dV + \int_\Gamma \overline{\boldsymbol{w}}\cdot\hat{\boldsymbol{f}},\end{aligned} \tag{17}$$

$$\int_{\Omega\setminus\Gamma} \lambda\nabla\cdot\boldsymbol{u}\,dV = 0, \tag{18}$$

$$\int_\Gamma \boldsymbol{v}\cdot[\boldsymbol{u}]\,dA = \int_\Gamma \boldsymbol{v}\cdot\boldsymbol{a}_i\,dA. \tag{19}$$

Note that we use $[\boldsymbol{w}] = \boldsymbol{w}^+ - \boldsymbol{w}^-$ to denote the jump in $\boldsymbol{w}$ across the interface, and $\overline{\boldsymbol{w}} = \frac{1}{2}(\boldsymbol{w}^+ + \boldsymbol{w}^-)$ for the average of $\boldsymbol{w}$ across the interface. We further use $\boldsymbol{u} = \boldsymbol{u}^{n+1}$ for conciseness. Also, note that the test functions $\boldsymbol{w}$, λ, and $\boldsymbol{v}$ complement the unknowns $\boldsymbol{u}$, p, and $\boldsymbol{q}$ respectively. The pressure p and other Lagrange multiplier $\boldsymbol{q}$ enforce the continuity equation and the velocity jump condition, respectively.

We begin our discretization procedure by cutting each grid cell of the computational domain into portions belonging to Ω^+ and Ω^- with the aid of the level set function ϕ^{n+1}. Each cell that is cut will have one or more triangles (segments in 2D) which are used to calculate integrals over each cell that is cut. We refer to an individual triangle as a *surface element*. Different approaches exist for performing cutting from a level set; we explain the method we use in Section 3.4.5.

Following [1], we place $\boldsymbol{u}$ and p in space according to a standard MAC layout, with p at cell centers and $\boldsymbol{u}$ components at face centers. We create copies of these degrees of freedom near interfaces so that each phase will have virtual values

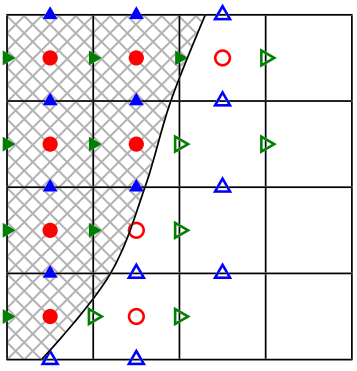


Fig. 2. A portion of a fluid grid is cut by an interface. The degrees of freedom for the hashed region are shown. Solid markers indicate real degrees of freedom, and hollow markers indicate virtual degrees of freedom. Note that only degrees of freedom with interpolating functions whose support intersects the hashed region participate in the discretization.

available as shown in Fig. 2. Although these degrees of freedom can be assigned directly, we create a copy of each degree of freedom for Ω^- and Ω^+, so that each fluid region has its own set of variables, and then discard the variables that are never referenced (those whose interpolating function support does not intersect Ω^- or Ω^+ respectively, see Fig. 2). The result is the same, but we found the resulting algorithm to be easier to implement. We follow a finite element discretization, letting

$$\begin{aligned} &\boldsymbol{u}^x(\boldsymbol{x}) = \sum_i \boldsymbol{u}_i^x N_i^x(\boldsymbol{x}), \qquad \boldsymbol{u}^y(\boldsymbol{x}) = \sum_i \boldsymbol{u}_i^y N_i^y(\boldsymbol{x}), \qquad \boldsymbol{u}^z(\boldsymbol{x}) = \sum_i \boldsymbol{u}_i^z N_i^z(\boldsymbol{x}), \qquad p(\boldsymbol{x}) = \sum_i p_i P_i(\boldsymbol{x}), \\ &\boldsymbol{q}(\boldsymbol{x}) = \sum_i q_i \boldsymbol{Q}_i(\boldsymbol{x}), \end{aligned} \tag{20}$$

as in [1], where $N_i^x(\boldsymbol{x})$, $N_i^y(\boldsymbol{x})$, and $N_i^z(\boldsymbol{x})$ define the standard piecewise trilinear basis functions associated with the velocity nodes for the respective dimension, and $P_i(\boldsymbol{x})$ is 1 in MAC cell i and 0 otherwise. We have also introduced the (vector-valued) basis $\boldsymbol{Q}_i$ and (scalar) degrees of freedom q_i for $\boldsymbol{q}(\boldsymbol{x})$. The way these are defined is critical to capturing the null mode properly, and we delay the definition of these until Section 3.4.3. We discretize the test variables the same way as their corresponding degrees of freedom as

$$\begin{aligned} &\boldsymbol{w}^x(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i^x N_i^x(\boldsymbol{x}), \qquad \boldsymbol{w}^y(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i^y N_i^y(\boldsymbol{x}), \qquad \boldsymbol{w}^z(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i^z N_i^z(\boldsymbol{x}), \qquad \lambda(\boldsymbol{x}) = \sum_i \lambda_i P_i(\boldsymbol{x}), \\ &\boldsymbol{v}(\boldsymbol{x}) = \sum_i v_i \boldsymbol{Q}_i(\boldsymbol{x}). \end{aligned} \tag{21}$$

This leaves the discretization of the forcing terms. The body force $\boldsymbol{f}$ is discretized as a vector quantity $(\boldsymbol{f}_i^x, \boldsymbol{f}_i^y, \boldsymbol{f}_i^z)$ that is constant per MAC cell as

$$\boldsymbol{f}^x(\boldsymbol{x}) = \sum_i \boldsymbol{f}_i^x P_i(\boldsymbol{x}), \qquad \boldsymbol{f}^y(\boldsymbol{x}) = \sum_i \boldsymbol{f}_i^y P_i(\boldsymbol{x}), \qquad \boldsymbol{f}^z(\boldsymbol{x}) = \sum_i \boldsymbol{f}_i^z P_i(\boldsymbol{x}). \tag{22}$$

The interface force $\hat{\boldsymbol{f}}$ is discretized with a vectorial force $(\hat{\boldsymbol{f}}_i^x, \hat{\boldsymbol{f}}_i^y, \hat{\boldsymbol{f}}_i^z)$ that is constant over each surface element i. Letting E_i be 1 on surface element i and 0 elsewhere,

$$\hat{\boldsymbol{f}}^x(\boldsymbol{x}) = \sum_i \hat{\boldsymbol{f}}_i^x E_i(\boldsymbol{x}), \qquad \hat{\boldsymbol{f}}^y(\boldsymbol{x}) = \sum_i \hat{\boldsymbol{f}}_i^y E_i(\boldsymbol{x}), \qquad \hat{\boldsymbol{f}}^z(\boldsymbol{x}) = \sum_i \hat{\boldsymbol{f}}_i^z E_i(\boldsymbol{x}). \tag{23}$$

The velocity jump is discretized in the same way as $\hat{\boldsymbol{f}}$, so that

$$\boldsymbol{a}^x(\boldsymbol{x}) = \sum_i \boldsymbol{a}_i^x E_i(\boldsymbol{x}), \qquad \boldsymbol{a}^y(\boldsymbol{x}) = \sum_i \boldsymbol{a}_i^y E_i(\boldsymbol{x}), \qquad \boldsymbol{a}^z(\boldsymbol{x}) = \sum_i \boldsymbol{a}_i^z E_i(\boldsymbol{x}). \tag{24}$$

The discretized equations can now be written in matrix form as

$$\begin{pmatrix} \boldsymbol{M}^x + \boldsymbol{A}^{xx} + \boldsymbol{B}^x & \boldsymbol{A}^{xy} & \boldsymbol{A}^{xz} & -\boldsymbol{G}^x & \boldsymbol{H}^x \\ \boldsymbol{A}^{yx} & \boldsymbol{M}^y + \boldsymbol{A}^{yy} + \boldsymbol{B}^y & \boldsymbol{A}^{yz} & -\boldsymbol{G}^y & \boldsymbol{H}^y \\ \boldsymbol{A}^{zx} & \boldsymbol{A}^{zy} & \boldsymbol{M}^z + \boldsymbol{A}^{zz} + \boldsymbol{B}^z & -\boldsymbol{G}^z & \boldsymbol{H}^z \\ -(\boldsymbol{G}^x)^T & -(\boldsymbol{G}^y)^T & -(\boldsymbol{G}^z)^T & \boldsymbol{0} & \boldsymbol{0} \\ (\boldsymbol{H}^x)^T & (\boldsymbol{H}^y)^T & (\boldsymbol{H}^z)^T & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^x \\ \boldsymbol{u}^y \\ \boldsymbol{u}^z \\ \boldsymbol{p} \\ \boldsymbol{q} \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}^x(\boldsymbol{u}^\star)^x + \boldsymbol{J}^x \boldsymbol{f}^x + \boldsymbol{K}^x \hat{\boldsymbol{f}}^x \\ \boldsymbol{M}^y(\boldsymbol{u}^\star)^y + \boldsymbol{J}^y \boldsymbol{f}^y + \boldsymbol{K}^y \hat{\boldsymbol{f}}^y \\ \boldsymbol{M}^z(\boldsymbol{u}^\star)^z + \boldsymbol{J}^z \boldsymbol{f}^z + \boldsymbol{K}^z \hat{\boldsymbol{f}}^z \\ \boldsymbol{0} \\ \boldsymbol{L}^x \boldsymbol{a}^x + \boldsymbol{L}^y \boldsymbol{a}^y + \boldsymbol{L}^z \boldsymbol{a}^z \end{pmatrix}, \tag{25}$$

where we have defined the matrix blocks ($r, s \in \{x, y, z\}$)

$$\left(\boldsymbol{M}^r\right)_{ij} = \alpha \int_{\Omega\setminus\Gamma} N_i^r N_j^r \, dV, \qquad \left(\boldsymbol{A}^{rs}\right)_{ij} = \int_{\Omega\setminus\Gamma} \mu N_{i,s}^r N_{j,r}^s \, dV, \tag{26}$$

$$\left(\boldsymbol{B}^r\right)_{ij} = \sum_{s\in\{x,y,z\}} \int_{\Omega\setminus\Gamma} \mu N_{i,s}^r N_{j,s}^r \, dV, \qquad \left(\boldsymbol{G}^r\right)_{ij} = \int_{\Omega\setminus\Gamma} N_{i,r}^r P_j \, dV, \tag{27}$$

$$\left(\boldsymbol{H}^r\right)_{ij} = \int_{\Gamma} \Theta_i N_i^r \boldsymbol{Q}_j^r \, dA, \qquad \left(\boldsymbol{J}^r\right)_{ij} = \int_{\Omega\setminus\Gamma} N_i^r P_j \, dV, \tag{28}$$

$$\left(\boldsymbol{K}^r\right)_{ij} = \int_{\Gamma} \Phi_i N_i^r E_j \, dA, \qquad \left(\boldsymbol{L}^r\right)_{ij} = \int_{\Gamma} \boldsymbol{Q}_i^r E_j \, dA. \tag{29}$$

In the $\boldsymbol{H}^r$ matrices, the value $\Theta_i = 1$ if degree of freedom i corresponds to the Ω^+ phase and $\Theta_i = -1$ if degree of freedom i corresponds to the Ω^- phase. For an interface, $\Phi_i = \frac{1}{2}$ in the $\boldsymbol{K}^r$ matrices; this will change for other types of boundary conditions that we mention in Section 3.4.4.

3.4.2. Null modes

The discretization of the Stokes equations in [1] allowed for nullspace modes corresponding to the null modes of the corresponding continuous weak formulation. In the periodic Stokes case, there is a constant velocity mode per dimension and a constant pressure mode. In problems with an interface, the pressure mode will also have nonzero $\boldsymbol{q}$ entries. The primary limitation restricting the discretization in [1] to two dimensions is the inability to capture the pressure mode discretely in 3D. We present a modification to the discretization of $\boldsymbol{q}$ that resolves this limitation and captures null modes discretely in either two or three dimensions.

First, we must identify the null modes for our weak formulation of Navier–Stokes with an interface and a periodic boundary, but no other boundary conditions. (We will discuss the effect of other boundary conditions on null modes in Section 3.4.4.) A null mode $(\boldsymbol{u}, p, \boldsymbol{q})$ must satisfy homogeneous versions of (17), (18), and (19) for any $(\boldsymbol{w}, \lambda, \boldsymbol{v})$:

$$\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u} \, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}\left(\nabla\boldsymbol{w} + \nabla\boldsymbol{w}^T\right) : \left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) dV - \int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{w} \, dV + \int_{\Gamma} [\boldsymbol{w}]\cdot\boldsymbol{q} \, dA = 0 \tag{30}$$

$$\int_{\Omega\setminus\Gamma} \lambda\nabla\cdot\boldsymbol{u} \, dV = 0 \tag{31}$$

$$\int_{\Gamma} \boldsymbol{v}\cdot[\boldsymbol{u}] \, dA = 0. \tag{32}$$

Letting $\boldsymbol{w} = \boldsymbol{u}$, $\lambda = p$, and $\boldsymbol{v} = \boldsymbol{q}$,

$$\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{u}\cdot\boldsymbol{u} \, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) : \left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) dV - \int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{u} \, dV + \int_{\Gamma} [\boldsymbol{u}]\cdot\boldsymbol{q} \, dA = 0 \tag{33}$$

$$\int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{u} \, dV = 0 \tag{34}$$

$$\int_{\Gamma} \boldsymbol{q}\cdot[\boldsymbol{u}] \, dA = 0. \tag{35}$$

Combining these yields

$$\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{u}\cdot\boldsymbol{u} \, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) : \left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) dV = 0. \tag{36}$$

Both terms are clearly nonnegative. Since $\alpha > 0$, the first term will be positive unless $\boldsymbol{u} = \boldsymbol{0}$. Thus, any null mode necessarily has $\boldsymbol{u} = \boldsymbol{0}$. Note that our weak Navier–Stokes formulation has no translational null modes, unlike the periodic Stokes problem. This reduces the homogeneous system to

$$\mathbf{0} = -\int_{\Omega\setminus\Gamma} \nabla\cdot\boldsymbol{w}p\,dV + \int_{\Gamma}[\boldsymbol{w}]\cdot\boldsymbol{q}\,dA \tag{37}$$

$$= -\int_{\Omega\setminus\Gamma} \nabla\cdot(p\boldsymbol{w})\,dV + \int_{\Omega\setminus\Gamma}\boldsymbol{w}\cdot\nabla p\,dV + \int_{\Gamma}[\boldsymbol{w}]\cdot\boldsymbol{q}\,dA \tag{38}$$

$$= \int_{\Omega\setminus\Gamma}\boldsymbol{w}\cdot\nabla p\,dV + \int_{\Gamma}[p\boldsymbol{w}]\cdot\boldsymbol{n}\,dA + \int_{\Gamma}[\boldsymbol{w}]\cdot\boldsymbol{q}\,dA. \tag{39}$$

It remains to determine conditions on p and $\boldsymbol{q}$ for a null mode. If we choose $f(\boldsymbol{x})$ to be a smooth scalar function that is positive over some set $U\subset\Omega\setminus\Gamma$ and zero elsewhere, then $\boldsymbol{w} = f\nabla p$ would produce

$$\mathbf{0} = \int_{\Omega\setminus\Gamma} f\|\nabla p\|^2\,dV, \tag{40}$$

from which $\nabla p = \mathbf{0}$ in U, and necessarily, $\nabla p = \mathbf{0}$ in $\Omega\setminus\Gamma$. Thus, p is piecewise constant, and

$$\mathbf{0} = \int_{\Gamma}[p\boldsymbol{w}]\cdot\boldsymbol{n}\,dA + \int_{\Gamma}[\boldsymbol{w}]\cdot\boldsymbol{q}\,dA. \tag{41}$$

If f is positive over some $U\subset\Omega$ but zero elsewhere, where $U\cap\Gamma\neq\emptyset$, then $\boldsymbol{w} = f\nabla\phi$, where ϕ is the level set, produces

$$\mathbf{0} = \int_{\Gamma}[p]f\nabla\phi\cdot\boldsymbol{n}\,dA = \int_{\Gamma}[p]f\,dA. \tag{42}$$

From this it follows that $[p] = 0$ in U and thus $[p] = 0$ over Γ. Finally,

$$\mathbf{0} = \int_{\Gamma}[\boldsymbol{w}]\cdot(p\boldsymbol{n}+\boldsymbol{q})\,dA. \tag{43}$$

By defining $\boldsymbol{w}$ to be f times an arbitrary piecewise constant vector, we are forced to conclude that $\boldsymbol{q} = -p\boldsymbol{n}$, where p is the constant pressure. Thus, the only possible nullspace is $\boldsymbol{u} = \mathbf{0}$, $p = c$, and $\boldsymbol{q} = -c\boldsymbol{n}$, where c is a constant.

3.4.3. Discretizing the interface stress jump

In Section 3.4.1, we introduced our discretization for all quantities except $\boldsymbol{q}$, whose description we left at (20). We will take up this topic here.

If we return to the equation for the constant pressure null mode $\boldsymbol{u} = \mathbf{0}$, $p = c$, $\boldsymbol{q} = -c\boldsymbol{n}$, we quickly run into a problem in 3D. In 2D, we can cut the MAC grid cells in a manner yielding one surface element per cut cell (ignoring under-resolved cases where there may be a second element). Such a procedure was employed in [1], and for each cut cell the normal of that element was chosen as $\boldsymbol{n}$. In 3D, it is generally impossible to maintain one surface element per cut cell, so for many cells we do not have an obvious candidate $\boldsymbol{n}$.

If we substitute the null mode into (30), we get

$$-\int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{w}\,dV + \int_{\Gamma}[\boldsymbol{w}]\cdot\boldsymbol{q}\,dA = 0. \tag{44}$$

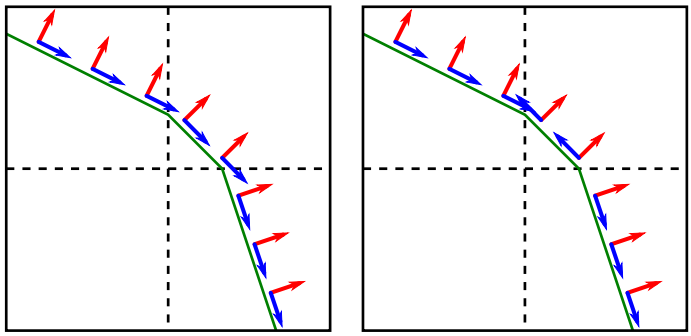
Applying the divergence theorem and using $p = c$ yields

$$\int_{\Gamma}[\boldsymbol{w}]\cdot(c\boldsymbol{n}+\boldsymbol{q})\,dA = 0. \tag{45}$$

In [1], $\boldsymbol{n}$ was constant per cut MAC cell, so discretizing $\boldsymbol{q}$ as constant per surface element (of which they had one per cut MAC cell) allowed them to discretely capture the pressure nullspace. If $\boldsymbol{w}$ were defined as piecewise constant per MAC cell, then discretizing $\boldsymbol{q}$ as constant per cell would lead to $\boldsymbol{q}$ being defined over each MAC cell C_i as

$$\boldsymbol{q}|_{C_i} = -c\Big(\int_{\Gamma\cap C_i} dA\Big)^{-1}\int_{\Gamma\cap C_i}\boldsymbol{n}\,dA \tag{46}$$

as the discrete nullspace. This is not the case, however, since $\boldsymbol{w}$ is discretized with the non-constant bases $N^x(\boldsymbol{x})$, $N^y(\boldsymbol{x})$, and $N^z(\boldsymbol{x})$.



(a) $\boldsymbol{Q}_n(\boldsymbol{x})$ and $\boldsymbol{Q}_t(\boldsymbol{x})$ for one MAC cell. (b) $\boldsymbol{Q}_t(\boldsymbol{x})$ assigned inconsistently.

Fig. 3. MAC cell with doubly-fine subcells and interface elements. $\boldsymbol{Q}_n(\boldsymbol{x})$ and $\boldsymbol{Q}_t(\boldsymbol{x})$ are constant per interface element (and doubly-fine subcell) but not per MAC cell. In 2D, cutting on a doubly-fine grid may produce multiple segments, which must be oriented consistently.

Our solution to this problem is to define a normal component q_n and two tangential components q_{t0} and q_{t1} for each cut MAC cell. Then, the basis $\boldsymbol{Q}_n(\boldsymbol{x})$ for the normal component q_n is the local normal direction $\boldsymbol{Q}_n(\boldsymbol{x}) = \boldsymbol{n}(\boldsymbol{x})$, which will be different for every surface element in the MAC cell. Now, the nullspace will be captured discretely as $q_n = -c$ and $q_{t0} = q_{t1} = 0$. The tangential bases $\boldsymbol{Q}_{t0}(\boldsymbol{x})$ and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ should be orthogonal tangential directions local to each element.

Unlike $\boldsymbol{Q}_n(\boldsymbol{x})$, which will automatically be consistent across elements in a MAC cell, the directions $\boldsymbol{Q}_{t0}(\boldsymbol{x})$ and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$, if not chosen carefully, could vary wildly in 3D. (In 2D, the tangential component can also be chosen consistently, though in 2D only one element is required anyway.) To see why such inconsistency may be problematic, consider a MAC cell cut by two coplanar surface elements e_0, e_1 of equal area. Let $\boldsymbol{Q}_{t0}(\boldsymbol{x}) = -\boldsymbol{Q}_{t0}(\boldsymbol{y})$ and $\boldsymbol{Q}_{t1}(\boldsymbol{x}) = -\boldsymbol{Q}_{t1}(\boldsymbol{y})$, where $\boldsymbol{x} \in e_0$ and $\boldsymbol{y} \in e_1$. Such a configuration would be incapable of applying a tangential tractive component in the MAC cell, since the tangential contribution from q_{t0} and q_{t1} to one element would cancel out their contributions to the other element.

To prevent tangential inconsistencies, we define a reference orientation for the MAC cell. The normal direction for this orientation is the weighted normal,

$$\bar{\boldsymbol{n}}' = \int_\Gamma \boldsymbol{n}\, dA, \qquad \bar{\boldsymbol{n}} = \frac{\bar{\boldsymbol{n}}'}{\|\bar{\boldsymbol{n}}'\|}, \tag{47}$$

where $A > 0$ is the area and $\bar{\boldsymbol{n}}$ is the unit direction. The first reference tangential direction $\bar{\boldsymbol{t}}_0$ is chosen as an arbitrary vector orthogonal to $\bar{\boldsymbol{n}}$, and $\bar{\boldsymbol{t}}_1 = \bar{\boldsymbol{t}}_0 \times \bar{\boldsymbol{n}}$, which we write as $\bar{\boldsymbol{R}} = (\bar{\boldsymbol{t}}_1 \quad \bar{\boldsymbol{t}}_0 \quad \bar{\boldsymbol{n}})$. To construct the local orientation for an element e, we begin by mapping the element's normal $\boldsymbol{n}_e$ into the reference frame as $\hat{\boldsymbol{n}}_e = \bar{\boldsymbol{R}}^T \boldsymbol{n}_e$. Note that if the adjacent elements are similar in orientation, then $\boldsymbol{n}_e \approx \bar{\boldsymbol{n}}$ and $\hat{\boldsymbol{n}}_e \approx \boldsymbol{k}$, where $\boldsymbol{i}$, $\boldsymbol{j}$, and $\boldsymbol{k}$ are the axial unit vectors. Similarly, we should have $\hat{\boldsymbol{t}}_{0e} \approx \boldsymbol{j}$ and $\hat{\boldsymbol{t}}_{1e} \approx \boldsymbol{i}$. This suggests choosing

$$\hat{\boldsymbol{t}}'_{0e} = \left(\boldsymbol{I} - \hat{\boldsymbol{n}}_e \hat{\boldsymbol{n}}_e^T\right)\boldsymbol{j}, \qquad \hat{\boldsymbol{t}}_{0e} = \frac{\hat{\boldsymbol{t}}'_{0e}}{\|\hat{\boldsymbol{t}}'_{0e}\|}, \qquad \hat{\boldsymbol{t}}_{1e} = \hat{\boldsymbol{t}}_{0e} \times \hat{\boldsymbol{n}}_e, \qquad \boldsymbol{t}_{0e} = \bar{\boldsymbol{R}}\hat{\boldsymbol{t}}_{1e}, \qquad \boldsymbol{t}_{1e} = \bar{\boldsymbol{R}}\hat{\boldsymbol{t}}_{0e}. \tag{48}$$

We found this local definition of the tangential directions to work well in practice. We can now define the bases for $\boldsymbol{q}$ locally as

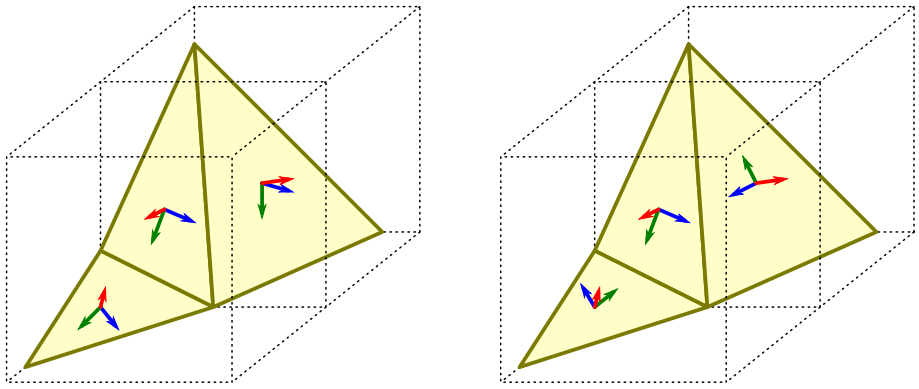
$$\boldsymbol{Q}_n(\boldsymbol{x}) = \boldsymbol{n}_e, \qquad \boldsymbol{Q}_{t0}(\boldsymbol{x}) = \boldsymbol{t}_{0e}, \qquad \boldsymbol{Q}_{t1}(\boldsymbol{x}) = \boldsymbol{t}_{1e}, \tag{49}$$

where e is the element at location $\boldsymbol{x}$. In 2D, the tangential direction is simply chosen by rotating the normal direction clockwise one-quarter turn. Note that unlike the bases for $\boldsymbol{u}$ or $\boldsymbol{p}$, the bases for $\boldsymbol{q}$ are vector quantities. For simplicity of exposition, we index the q_i degrees of freedom uniformly, ignoring the distinction between normal and tangential degrees of freedom. These consistency concerns are illustrated in Figs. 3 and 4.

3.4.4. Boundary conditions

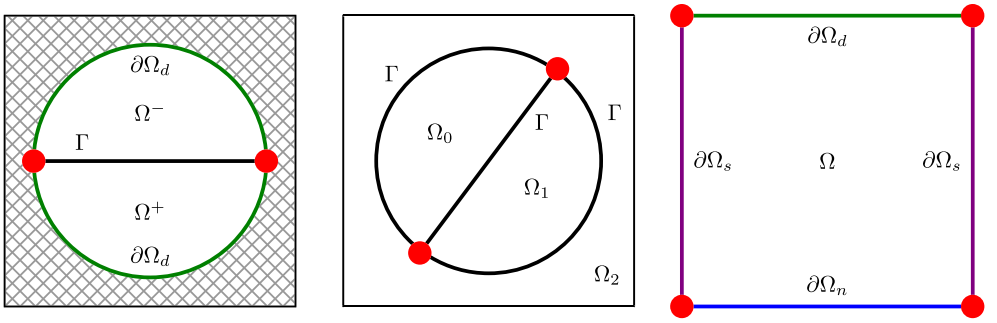
Up to this point, we have described how to discretize the interface Γ splitting the domain Ω, but have not treated boundary conditions on Ω, excepting periodic boundary conditions which can be handled in the obvious way. An advantage of our discretization of the embedded interface is the relative ease with which we can modify it to implement Dirichlet velocity boundary conditions (5), Neumann boundary conditions (6), and slipping boundary conditions (7) and (8).

We represent our boundary conditions by treating the regions beyond Dirichlet, Neumann, and slip boundaries as special fluid phases. These boundary condition phases have level sets associated with them representing the location of the boundary. The boundary condition phases are not real fluid. There are no velocity or pressure degrees of freedom associated with these phases. This representation allows us to reuse our interface routines with little extra work. The problem of handling the different boundary conditions becomes a problem of handling an interface between a fluid region and boundary



(a) $\boldsymbol{Q}_n(\boldsymbol{x})$, $\boldsymbol{Q}_{t0}(\boldsymbol{x})$, and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ for part of a MAC cell. (b) $\boldsymbol{Q}_{t0}(\boldsymbol{x})$, and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ oriented inconsistently.

Fig. 4. These illustrations show the local coordinate directions $\boldsymbol{Q}_n(\boldsymbol{x})$ in red, $\boldsymbol{Q}_{t0}(\boldsymbol{x})$ in green, and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ in blue. $\boldsymbol{Q}_n(\boldsymbol{x})$, $\boldsymbol{Q}_{t0}(\boldsymbol{x})$, and $\boldsymbol{Q}_{t1}(\boldsymbol{x})$ are constant per interface element and should be consistent within a MAC cell (left). Consistency in the normal direction is automatic, but if care is not taken, the tangential directions will be inconsistent. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Interface intersects boundary (b) Three interfaces meet (c) Different boundary conditions meet

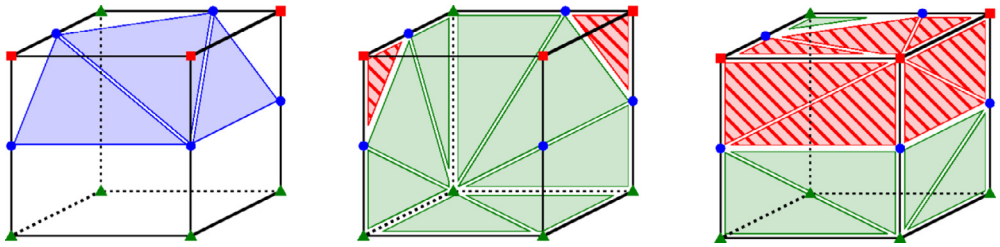
Fig. 5. Triple junctions are formed when interfaces or boundary conditions of different types meet at a point. The filled circles represent the location of triple junction in these examples.

condition region. Note that if two boundary condition regions are adjacent, there will be an interface between them, which we can ignore. There will also be a triple junction at the point where the fluid region meets the two boundary condition regions. Some common triple junction configurations are shown in Fig. 5. We leave the problem of handling triple junctions for future work.

Dirichlet boundary conditions are implemented by treating the region beyond the boundary as having an identically zero velocity. Nonzero Dirichlet velocity boundary conditions ($\boldsymbol{u} = \boldsymbol{b}$ at $\boldsymbol{x} \in \partial\Omega_d$) are treated as velocity jumps at the interface ($[\boldsymbol{u}] = \boldsymbol{b}$ at $\boldsymbol{x} \in \Gamma_d$). The stress in the region beyond the boundary condition can be taken to be continuous with the stress in the fluid, so ($[\sigma \cdot \boldsymbol{n}] = \boldsymbol{0}$ at $\boldsymbol{x} \in \Gamma_d$). There will be $\boldsymbol{q}$ degrees of freedom for Dirichlet boundary conditions just as there are for regular interfaces. Practically speaking this amounts to omitting the velocity degrees of freedom (as well as the associated rows and columns of the system) corresponding to the Dirichlet fluid phase.

In the Neumann case, we wish to enforce a desired normal stress. We treat the region beyond the boundary as having identically zero stress. In (29), we divide the interface stress jump evenly to both sides of the interface ($\Phi_i = \frac{1}{2}$). In the Neumann case, we must put the entire contribution on the side corresponding to the fluid ($\Phi_i = 1$) since the other rows will be discarded. The interface stress now corresponds to the region beyond the boundary, so $\boldsymbol{q} = \overline{\sigma} \cdot \boldsymbol{n} = \boldsymbol{0}$. Eliminating $\boldsymbol{q}$ in this way corresponds to not having any particular velocity jump to enforce ($\boldsymbol{a}$ will never be used). Practically speaking, Neumann boundary conditions are implemented by omitting $\boldsymbol{q}$ degrees of freedom corresponding to the Neumann boundary condition and omitting the corresponding entries of the system. Note that the Dirichlet and Neumann treatments above are equivalent to the standard finite element treatments of these boundary conditions.

Our treatment of the slip boundary condition takes advantage of our division of $\boldsymbol{q}$ degrees of freedom into normal and tangential components. Slip is treated like Dirichlet in the normal direction and Neumann in the tangential directions. The tangential $\boldsymbol{q}$ degrees of freedom, as well as corresponding matrix entries, are omitted. Note that the equation corresponding to the normal component of $\boldsymbol{q}$ enforces the velocity jump condition in the normal direction ($[\boldsymbol{u}] = \boldsymbol{c}$), so omitting degrees of freedom in this way suffices to encode Dirichlet in the normal direction. The tangential portion requires a slight



(a) Interface elements (b) Boundary elements (back) (c) Boundary elements (front)

Fig. 6. We use a modified marching cubes table that emits both the usual interface elements (left) as well as a triangulation of the portion of boundary of the cell in each region (center and right). These extra triangles greatly simplify our integration process.

modification, since $(\boldsymbol{I} - \boldsymbol{n}\boldsymbol{n}^T)\hat{\boldsymbol{h}}$ must be used as the interface stress. As in the Neumann case, $\Phi_i = 1$ is used in (29). We demonstrate all three types of boundary conditions in our numerical examples.

Since our implementations of Dirichlet and slip boundary conditions do not eliminate the normal components of $\boldsymbol{q}$ degrees of freedom, a Dirichlet or slip boundary condition will not preclude the presence of a null mode. However, a Neumann boundary condition will prevent the existence of a null mode since its $\boldsymbol{q}$ degrees of freedom are removed.

3.4.5. Practical implementation

The primary difficulty in implementing the proposed method is computing the necessary integrals. Our pressure basis functions are piecewise constant over MAC cells, but our velocity basis functions are piecewise trilinear over cells whose corners contain the respective velocity degrees of freedom. Since we will be integrating products of these bases and their derivatives, we perform our integration over the cells of a doubly-fine grid. Over each doubly-fine cell, these products are all polynomials. The polynomials being integrated may be different, even discontinuous, across the boundaries between adjacent doubly-fine cells (even across those contained in the same MAC cell). This is a consequence of the staggering of the variables.

We represent our regions using level sets (both for the interface and boundary conditions) stored at MAC cell centers. Since we wish to integrate over doubly-fine cells, we interpolate our level set to populate a doubly-fine node-centered level set. This representation allows us to compute our interface geometry using marching cubes in 3D (marching squares in 2D) over the doubly-fine grid. The boundary integrals amount to integrating a polynomial over these triangles. Note that all of the bases, restricted to one interface element, are polynomials.

The volumetric integrals at first seem rather difficult, particularly in light of the rather complicated regions that occur with marching cubes. If approached in the right way, however, they are quite manageable. We begin by converting the volume integral into a surface integral using the divergence theorem as in [1]. This reduces the problem into one of integrating polynomials of one degree higher over the triangles on the boundary of the cut marching cubes volumes. We augment our marching cubes table (marching squares table for 2D) to emit these triangles (segments in 2D) on the surface of the cube in addition to the triangles on the interface itself. See Fig. 6 for an illustration. This enhancement of marching cubes is straightforward in practice, as most of the work involved is required to implement marching cubes in the first place. It also greatly simplifies the integration.

The highest degree polynomials we must integrate over cut volumes are of degree six in 3D (degree four in 2D), which occur for $\boldsymbol{M}^x$, $\boldsymbol{M}^y$, and $\boldsymbol{M}^z$. These become degree seven polynomials in 3D (degree five in 2D) once the divergence theorem is applied. The highest degree polynomials we integrate for boundary integrals is three in 3D (two in 2D). We perform all of these integrations using quadrature rules of high enough order (listed in [3]) to get the integral exactly.

Although solving the linear system (25) is by far the slowest step of our method, we still perform a few simple optimizations when computing the integrals. The first is to precompute the stencils for uncut doubly-fine cells (there will be eight such integrals required, since each octant of a MAC cell may contribute differently to the final stencil). Additional integrations are only required for integrating in cut cells or computing boundary integrals. Most cells are not cut and can simply use a copy of one of these precomputed stencils. For the cells that are cut, we will be computing many integrals over the same geometry, so we begin by integrating the monomials individually using quadrature rules. With these, integrating the actual basis polynomials reduces to a simple dot product.

We can also take advantage of the way in which the volume integral was converted into a surface integral using divergence theorem to save even more work. That is we can convert a volume integral over $f(x, y, z)$ into a surface integral using

$$\int_{\Omega} f\, dV = \int_{\partial\Omega} g\, \boldsymbol{n}_x\, dA = \int_{\partial\Omega} h\, \boldsymbol{n}_y\, dA = \int_{\partial\Omega} k\, \boldsymbol{n}_z\, dA \tag{50}$$

where we have integrated the polynomial $f(x, y, z)$ to obtain

$$g = \int f\,dx, \qquad h = \int f\,dy, \qquad k = \int f\,dz. \tag{51}$$

If we choose the x direction as our preferred direction, then $\boldsymbol{n}_x = 1$ along two faces of the cube, and $\boldsymbol{n}_x = 0$ along the other four. This means we can discard the boundary elements along four of the faces of the cube.

Finally, we only need to compute volume integrals on one side of an interface, since the integrals for the other can be obtained by subtracting from the integral over the whole cube, which we have precomputed.

3.5. Solving the system

We solve our system using preconditioned MINRES using the same Jacobi-style preconditioner as in [1]. We project out our nullspace (when we have one) inside the MINRES solver in addition to projecting the right hand side for compatibility, since we have found this to improve the convergence behavior of the solver. This simple preconditioner we employ significantly improves the conditioning of our systems, but in practice the systems remain very slow to solve. We leave the problem of finding a more effective preconditioner for future work.

3.6. Surface tension

There are many popular options for introducing surface tension into a fluid discretization that are available to us. Since our discretization has provisions built in for incorporating an interface force $\hat{\boldsymbol{f}}$, we take this approach. We begin by computing cellwise normals $\boldsymbol{n}_i$ and curvature values κ_i at MAC cell centers according to

$$\boldsymbol{n}_i = \frac{\nabla\phi^{n+1}}{\|\nabla\phi^{n+1}\|}, \qquad \boldsymbol{H}_i = \text{Hessian}\left(\phi^{n+1}\right), \qquad \kappa_i = \frac{\boldsymbol{n}_i^T \boldsymbol{H}_i \boldsymbol{n}_i - \text{tr}(\boldsymbol{H}_i)}{\|\nabla\phi^{n+1}\|}, \tag{52}$$

where all derivatives are computed using central differencing. Using these, we can compute estimates for $\hat{\boldsymbol{n}}$ and for the curvature $\hat{\kappa}$ wherever necessary, by cubic interpolation of $\boldsymbol{n}_i$ and κ_i. Finally, we approximate our surface tension as the interface force

$$\hat{\boldsymbol{f}} = -\beta\hat{\kappa}\hat{\boldsymbol{n}} \tag{53}$$

where β is the surface tension coefficient. Note that $\boldsymbol{n}_i$ and κ_i are only required near the interface, and reinitialization must be performed in a wide enough band for the combined central differencing stencil and cubic interpolation stencil.

3.7. Stability

3.7.1. System stability

The final step of our scheme ($\boldsymbol{u}^\star \to \boldsymbol{u}^{n+1}$) applies viscosity and enforces incompressibility. This operation is linear (or affine if there are forcing terms such as inhomogeneous boundary conditions or surface tension). This system can be expressed as

$$\begin{pmatrix} \boldsymbol{M}+\boldsymbol{S} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}\boldsymbol{u}^\star \\ \boldsymbol{0} \end{pmatrix}, \tag{54}$$

where $\boldsymbol{M}$ contains the inertial blocks, $\boldsymbol{S}$ contains the viscous blocks, and $\boldsymbol{G}$ contains the pressure and interface stress blocks. The vector $\boldsymbol{\lambda}$ contains the $\boldsymbol{p}$ and $\boldsymbol{q}$ degrees of freedom. Non-homogeneous terms on the right hand side are omitted. Note that the $\boldsymbol{\lambda}$ degrees of freedom are not state, in that these values can be discarded at the end of the time step. The only state variables present in this system are velocities.

The matrix $\boldsymbol{S}$ made up of the viscous blocks is symmetric positive semi-definite, since from our discretization $\boldsymbol{w}'\boldsymbol{S}\boldsymbol{u}$ is equal to the inner product $\int_\Omega \frac{\mu}{2}\nabla(\boldsymbol{w}+\boldsymbol{w}^T)\cdot\nabla(\boldsymbol{u}+\boldsymbol{u}^T)\,dV$ of the piecewise trilinear functions corresponding to the vectors $\boldsymbol{u}, \boldsymbol{w}$. We will substitute $\boldsymbol{S} = \boldsymbol{C}^T\boldsymbol{C}$ for the purposes of this analysis, and rewrite our matrix equation as

$$\begin{pmatrix} \boldsymbol{M}+\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}\boldsymbol{u}^\star \\ \boldsymbol{0} \end{pmatrix}. \tag{55}$$

Following [67] and letting $\boldsymbol{w} = \boldsymbol{C}\boldsymbol{u}^{n+1}$, we can transform this system to

$$\begin{pmatrix} \boldsymbol{C}\boldsymbol{M}^{-1} & \boldsymbol{0} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1} & -\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}+\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}\boldsymbol{M}^{-1} & \boldsymbol{0} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1} & -\boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{M}\boldsymbol{u}^\star \\ \boldsymbol{0} \end{pmatrix}, \tag{56}$$

$$\begin{pmatrix} \boldsymbol{C}+\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{G} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{C}^T\boldsymbol{C} & \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{n+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}\boldsymbol{u}^\star \\ \boldsymbol{G}^T\boldsymbol{u}^\star \end{pmatrix}, \tag{57}$$

$$\begin{pmatrix} \boldsymbol{I}+\boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{C}^T & \boldsymbol{C}\boldsymbol{M}^{-1}\boldsymbol{G} \\ \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{C}^T & \boldsymbol{G}^T\boldsymbol{M}^{-1}\boldsymbol{G} \end{pmatrix} \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}\boldsymbol{u}^\star \\ \boldsymbol{G}^T\boldsymbol{u}^\star \end{pmatrix}, \tag{58}$$

$$\left(\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} C \\ G^T \end{pmatrix} M^{-1} \begin{pmatrix} C \\ G^T \end{pmatrix}^T \right) \begin{pmatrix} w \\ \lambda \end{pmatrix} = \begin{pmatrix} C \\ G^T \end{pmatrix} u^\star, \tag{59}$$

$$\left(P + KM^{-1}K^T\right) z = Ku^\star, \tag{60}$$

where

$$P = \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}, \qquad K = \begin{pmatrix} C \\ G^T \end{pmatrix}, \qquad z = \begin{pmatrix} w \\ \lambda \end{pmatrix}. \tag{61}$$

Since both P and $KM^{-1}K^T$ are symmetric positive semi-definite, $P + KM^{-1}K^T$ may have the nullspace component z if and only if both PP and $KM^{-1}K^T$ do individually. $z^T Pz = 0$ implies $u^{n+1} = 0$, which reduces $z^T KM^{-1}K^T z = \lambda^T G^T M^{-1} G\lambda$. Since M is symmetric positive definite, we must have $G\lambda = 0$. That is, G has a nullspace. We often do have such a nullspace. Though this complicates the analysis, we note that we can ignore this nullspace since we will never get a component in it on the right hand side. In this way, we can solve this system for z.

The next step is to recover u^{n+1} from z, which we do using the momentum equation

$$\left(M + C^T C\right) u^{n+1} + G\lambda = Mu^\star, \tag{62}$$

$$u^{n+1} + M^{-1}C^T w + M^{-1}G\lambda = u^\star, \tag{63}$$

$$u^{n+1} + M^{-1}K^T z = u^\star, \tag{64}$$

$$u^{n+1} = u^\star - M^{-1}K^T z. \tag{65}$$

Finally, the change in kinetic energy due to the update $u^\star \to u^{n+1}$ is

$$\Delta KE = \frac{1}{2}\left(u^{n+1}\right)^T Mu^{n+1} - \frac{1}{2}u^{\star T} Mu^\star \tag{66}$$

$$= \frac{1}{2}\left(u^\star - M^{-1}K^T z\right)^T M\left(u^\star - M^{-1}K^T z\right) - \frac{1}{2}u^{\star T} Mu^\star \tag{67}$$

$$= \frac{1}{2} z^T KM^{-1}K^T z - z^T Ku^\star \tag{68}$$

$$= \frac{1}{2} z^T KM^{-1}K^T z - z^T\left(P + KM^{-1}K^T\right) z \tag{69}$$

$$= -\frac{1}{2} z^T KM^{-1}K^T z - z^T Pz \tag{70}$$

$$\leqslant 0. \tag{71}$$

Thus we see that this step does not introduce energy into the system.

3.7.2. Time step restriction

Empirically, our method appears to have a stability restriction on the value of the dimensionless quantity $\frac{\Delta t\mu}{\rho \Delta x^2}$ (see Sections 4.4 and 4.9). This limits the minimum choice for Δt. Since the criterion depends on refinement as $\frac{\Delta t}{\Delta x^2}$, convergence is possible as long as Δt is refined no faster than Δx^2. In particular, $\Delta t = k\Delta x$ and $\Delta t = k\Delta x^2$ are both suitable refinement strategies.

It is worth discussing the apparent source of this instability in more detail. In the absence of an interface, no instability is observed. When an interface (or boundary condition) is present and instability is observed, it starts near the interface. This in particular suggests that the modified velocity advection scheme proposed is stable. Indeed, the instability is also observed with the original advection scheme or no advection at all. Similarly, instability is observed with BDF or backward Euler.

An unusual characteristic of this stability restriction is that Δt must not be chosen *too small*. To see what may be causing this, consider a time step in the limit $\Delta t \to 0$. In this case, advection has no effect, and the viscosity terms vanish. Using backward Euler eliminates the complications of BDF. The only part of the time integration remaining that has an appreciable effect is setting up the right hand side and solving the system, which we showed will not increase energy. The source of the energy increase is the velocity extrapolation used to fill the ghost cells of $u^\star$ needed for the right hand side. If sufficient viscosity is present, this added energy is dissipated as it is introduced, and the scheme remains stable. Examining the role of Δt, μ, and ρ in this system, we can rescale the system so that the only reference to these quantities is through the expression $\frac{\Delta t\mu}{\rho}$. This is consistent with the empirical stability criterion suggested by our numerical examples. Noting that the viscosity blocks are scaled by $\frac{1}{\Delta x^2}$ relative to the inertial blocks leads to the full criterion $\frac{\Delta t\mu}{\rho \Delta x^2}$. This value describes the efficiency with which viscosity is able to damp out energy in our scheme.

To see why extrapolation is able to lead to instability, consider a set of uniformly spaced sample points $u_1, u_2, \ldots$. The value u_0 is to be computed by extrapolation. If $u_k = 1$ for $k \geqslant 2$ are set to a constant value but $u_1 = 1 + \varepsilon$, so that a small

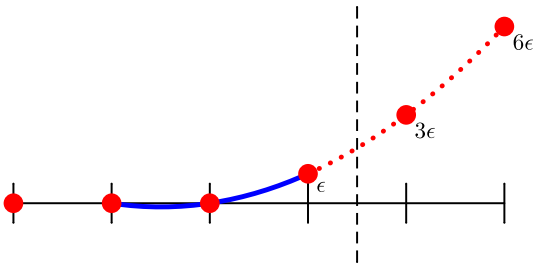
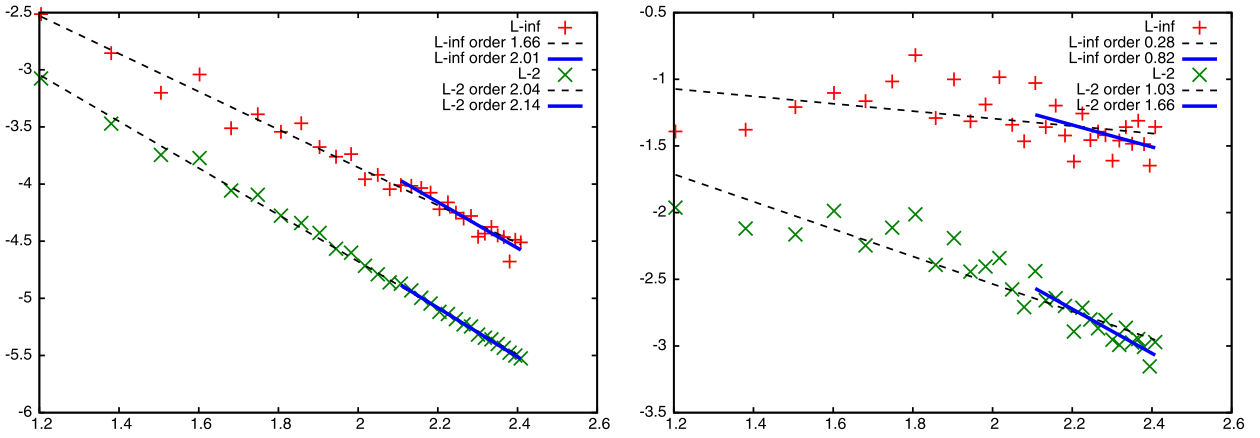


Fig. 7. Extrapolation amplifies errors. In this case, the ideal solution (black horizontal line) is perturbed by ε at the interface, which is then amplified by quadratic extrapolation to 3ε and 6ε in the ghost region.



(a) Velocity error (b) Pressure error

Fig. 8. Errors in the Taylor–Green vortex, Section 4.1, for velocity and pressure (shown log base 10) in L^∞ and L^2, plotted against resolutions from 16 to 256 by increments of 8 (shown log base 10). The pressure does not start to display convergence until the resolution is high, so separate regressions are provided for the highest resolutions 128 to 256 to eliminate bias from resolutions below the convergence regime. The estimated orders for velocity when throwing out the lowest resolutions are 2.01 in L^∞ and 2.14 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.82 in L^∞, 1.66 in L^2.

error has been made near the interface, then we will compute $u_0 = 1 + \varepsilon$ with constant extrapolation, $u_0 = 1 + 2\varepsilon$ with linear extrapolation, and $u_0 = 1 + 3\varepsilon$ with quadratic extrapolation (see Fig. 7). Thus, we see that extrapolation has magnified the error by a factor greater than one. Solving the system pulls some of this energy from the ghost region inside, where it is magnified further by extrapolation in the next time step. The above example has a growth factor of 3, though in practice a value near 1.25 is observed for unstable simulations. Instabilities always exhibit this slow and steady exponential march to infinity. Using lower order extrapolation decreases the growth rate, but even for constant extrapolation the factor is still slightly larger than one. This supports the idea that extrapolation is providing the amplification required for instability.

4. Numerical examples

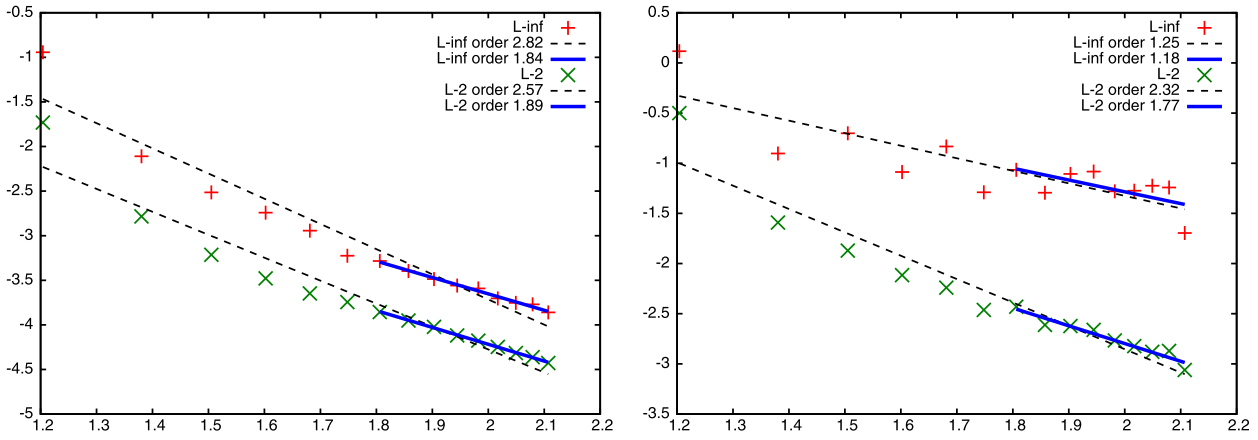
Our method supports a range of boundary conditions and forces. Through a mixture of analytic and more practical tests, we demonstrate second order accuracy for $\boldsymbol{u}$ in L^∞ and L^2, second order accuracy for p in L^2, and first order accuracy for p in L^∞. We also investigate the stability characteristics of our method.

4.1. Taylor–Green vortex

The Taylor–Green vortex is a popular analytic accuracy test for single-phase Navier–Stokes. We use a (dimensionless) domain $[0, \pi] \times [0, \pi]$ in which we confine fluid to the region $\sin(x)\sin(y) \geqslant k$, where $k = 0.2$. The fluid has $\rho = 1$, $\mu = 1$, and the final time is $T = 0.2$. The analytic solution is

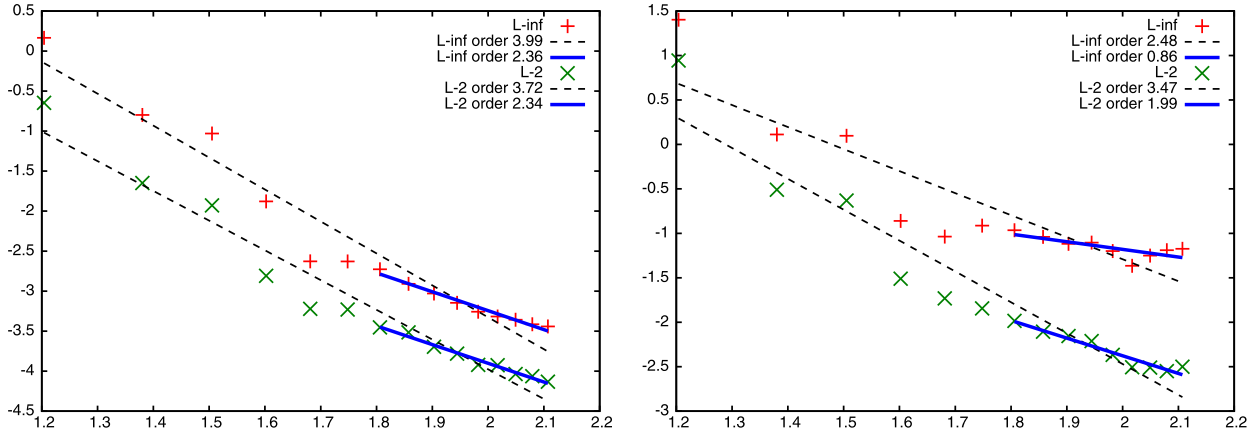
$$u = \sin(x)\cos(y), \qquad v = -\cos(x)\sin(y), \qquad p = \frac{1}{4}\rho\big(\cos(2x) + \cos(2y)\big)e^{-4\nu t}, \tag{72}$$

where $\nu = \frac{\mu}{\rho}$. The velocity field is initialized with the analytic velocity. Velocity and pressure errors along with convergence order estimates are shown in Fig. 8.



(a) Velocity error (b) Pressure error

Fig. 9. Translating Taylor–Green vortex errors, Section 4.2, for velocity and pressure (shown log base 10) plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2, with separate regressions provided for the highest resolutions 64 to 128. The estimated orders for velocity when throwing out the lowest resolutions are 1.84 in L^∞ and 1.89 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 1.18 in L^∞, 1.77 in L^2.



(a) Velocity error (b) Pressure error

Fig. 10. Analytic test I errors in velocity and pressure (shown log base 10), plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. Separate regressions are provided for the highest resolutions 64 to 128 to eliminate bias from earlier resolutions. The estimated orders for velocity when throwing out the lowest resolutions are 2.34 in L^∞ and 2.36 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.86 in L^∞, 1.99 in L^2.

4.2. Translating Taylor–Green vortex

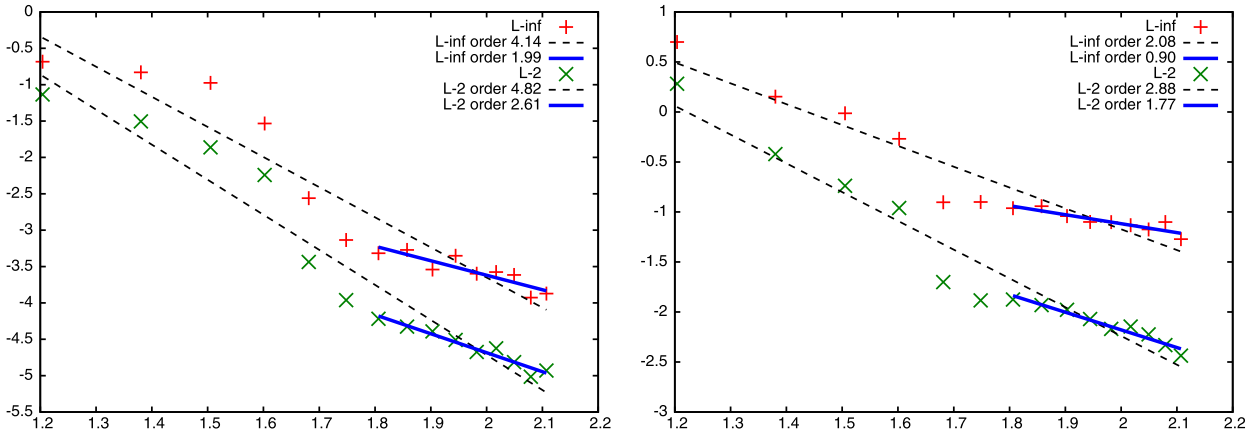
We test our method on a problem where two fluids are separated by an interface in the periodic domain $[0, 2\pi] \times [0, 2\pi]$. The interface is initially set to be the circle centered at $(\frac{11\pi}{10}, 0)$ with radius $\frac{3\pi}{5}$. Each fluid has $\rho = 1$ and $\mu = 2$, and the analytic solution for both fluids is given by a translating Taylor–Green vortex:

$$u = \sin(x - 0.2t)\cos(y - 0.5t), \qquad v = -\cos(x - 0.2t)\sin(y - 0.5t),$$
$$p = \frac{1}{4}\rho\big(\cos(2x - 0.2t) + \cos(2y - 0.5t)\big)e^{-4\nu t}, \tag{73}$$

where $\nu = \frac{\mu}{\rho}$. As before, the velocity field is initialized with the analytic velocity. Velocity and pressure errors, and estimates of the convergence order, are shown in Fig. 9.

4.3. Analytic test I

In this analytic test we evolve two fluids, separated by an interface, in the periodic domain $[-\pi, \pi] \times [-\pi, \pi]$. The interface in this example is the circle $x^2 + y^2 = (0.8\pi)^2$. We set an inner boundary at the circle $(x - 0.2\pi)^2 + y^2 = (0.2\pi)^2$,



(a) Velocity error (b) Pressure error

Fig. 11. Analytic test II errors in velocity and pressure (shown log base 10), plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. Separate regressions are provided for the highest resolutions 64 to 128 to eliminate bias from earlier resolutions. The estimated orders for velocity when throwing out the lowest resolutions are 1.99 in L^∞ and 2.61 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.90 in L^∞, 1.77 in L^2.

on which we apply a Neumann boundary condition. The fluid bounded by the inner and outer circles has $\rho^- = 1$, $\mu^- = 1$, and the outer fluid has $\rho^+ = 2$, $\mu^+ = 3$. The fluids are initialized with the analytic velocity and evolved to final time $T = 0.1$. The analytic solution is given by

$$u = \begin{cases} 0.2 - x & \boldsymbol{x} \in \Omega^-, \\ \sin(x)\cos(y) & \text{otherwise}, \end{cases} \qquad v = \begin{cases} y & \boldsymbol{x} \in \Omega^-, \\ -\cos(x)\sin(y) & \text{otherwise}, \end{cases}$$
$$p = \begin{cases} 0 & \boldsymbol{x} \in \Omega^-, \\ \frac{1}{4}\rho^+(\cos(2x) + \cos(2y))e^{-4\nu t} & \text{otherwise}, \end{cases} \tag{74}$$

where $\nu = \frac{\mu}{\rho}$. Velocity and pressure errors along with convergence order estimates are shown in Fig. 10.

4.4. Analytic test II

We embed the circle $x^2 + y^2 = (0.8\pi)^2$ into the domain $[-\pi, \pi] \times [-\pi, \pi]$. A circle $(x - 0.2\pi)^2 + y^2 = (0.2\pi)^2$ separates the larger circle into an inner domain Ω^- and an outer domain Ω^+, and a slip boundary condition is enforced along the boundary of the outer circle. The inner fluid has $\rho^- = 1$, $\mu^- = 1$ and the outer fluid has $\rho^+ = 2$, $\mu^+ = 3$. The velocity field is initialized with the analytic velocity and evolved to the final time $T = 0.1$. The analytic solution is given by

$$u = \begin{cases} 0.2 - x & \boldsymbol{x} \in \Omega^-, \\ -y & \text{otherwise}, \end{cases} \qquad v = \begin{cases} y & \boldsymbol{x} \in \Omega^-, \\ x & \text{otherwise}, \end{cases} \qquad p = \begin{cases} 0 & \boldsymbol{x} \in \Omega^-, \\ 0.5\rho^+(x^2 + y^2) & \text{otherwise}. \end{cases} \tag{75}$$

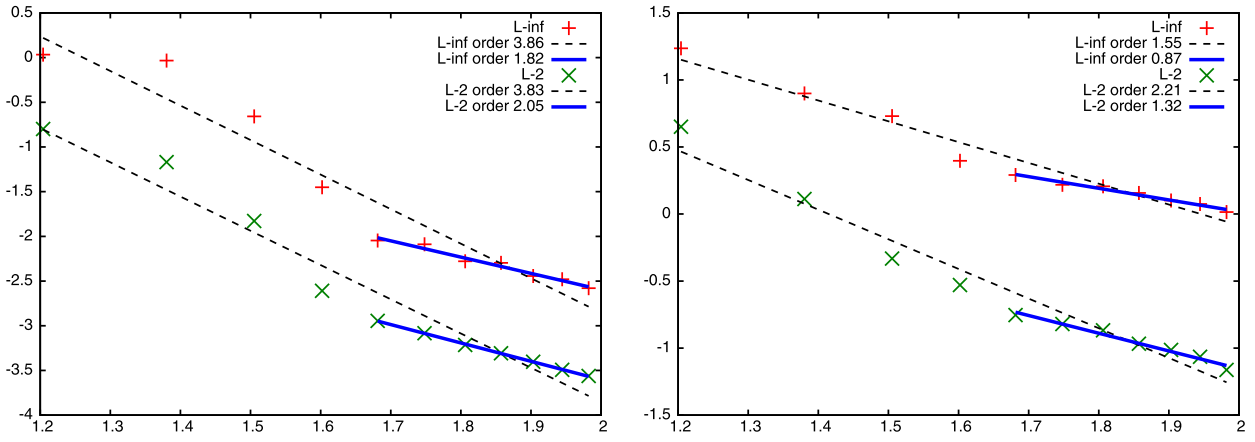
Velocity and pressure errors along with convergence order estimates are shown in Fig. 11.

4.5. Analytic test II-3D

We examine a three-dimensional analogue of our test from the previous section: The sphere $x^2 + y^2 + z^2 = (0.8\pi)^2$ is embedded into the dimensionless domain $[-\pi, \pi] \times [-\pi, \pi] \times [-\pi, \pi]$. A shell $(x - 0.2\pi)^2 + y^2 + z^2 = (0.2\pi)^2$ separates the larger sphere into an inner domain Ω^- and an outer domain Ω^+. As before, the slip boundary condition is enforced along the boundary of the outer circle. The inner fluid has $\rho^- = 1$, $\mu^- = 1$ and the outer fluid has $\rho^+ = 2$, $\mu^+ = 3$. As in previous examples, the velocity field is initialized with the analytic velocity and evolved to the final time $T = 0.1$. The analytic solution is given by

$$u = \begin{cases} 0.2 - x & \boldsymbol{x} \in \Omega^-, \\ 2z - 3y & \text{otherwise}, \end{cases} \qquad v = \begin{cases} y & \boldsymbol{x} \in \Omega^-, \\ 3x - z & \text{otherwise}, \end{cases} \qquad w = \begin{cases} -2z & \boldsymbol{x} \in \Omega^-, \\ y - 2x & \text{otherwise}, \end{cases}$$
$$p = \begin{cases} 0 & \boldsymbol{x} \in \Omega^-, \\ 0.5\rho^+(x^2 + y^2) & \text{otherwise}. \end{cases}$$

Velocity and pressure errors along with convergence order estimates are shown in Fig. 12.



(a) Velocity error (b) Pressure error

Fig. 12. Analytic test II-3D errors in velocity and pressure (shown log base 10), plotted against resolutions from 16 to 96 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. Separate regressions are provided for the highest resolutions 48 to 96 to eliminate bias from earlier resolutions. The estimated orders for velocity when throwing out the lowest resolutions are 1.82 in L^∞ and 2.05 in L^2. For pressure, the estimated orders when throwing out the lowest resolutions are 0.87 in L^∞, 1.32 in L^2.

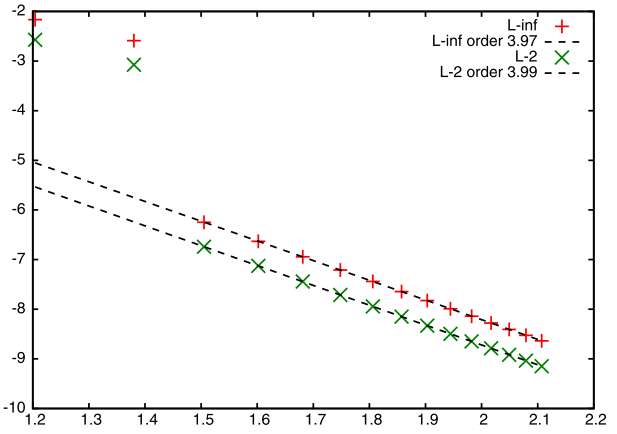


Fig. 13. Couette flow errors in velocity (shown log base 10), plotted against resolutions from 16 to 128 by increments of 8 (shown log base 10). Regression lines and the corresponding orders shown for L^∞ and L^2. The analytic solution (piecewise linear velocity, constant zero pressure) would often be resolved exactly by a second order method. In our case, we do observe fourth order velocity convergence on this simple test. Note that the first two resolutions are too small to resolve the setup and have been omitted from the regression. The pressure errors are below 10^{-9} in L^∞ and L^2 for all resolutions and are limited by the convergence tolerance of our MINRES solver.

4.6. Two-phase Couette flow

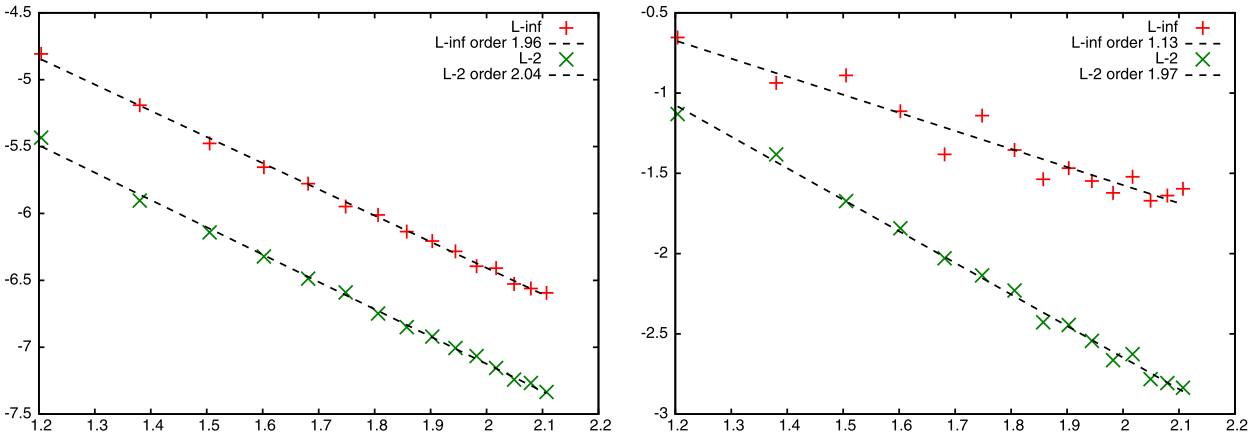
We run a two-phase Couette flow test, where two phases are separated by a stationary interface. The phases have different density and viscosity. The domain is $[0,1]\times[0,1]$. The fluid is confined by vertical no-slip walls at $x_0 = 0.2$ (where $\boldsymbol{u}(x_0, y) = (0,1)$) and $x_2 = 0.8$ (where $\boldsymbol{u}(x_2, y) = (0,-1)$). Periodic boundary conditions are enforced at the top and bottom of the domain. The interface is vertical at $x_1 = 0.5$, with phase 0 ($\rho^- = 1$, $\mu^- = 1$) occupying $0.2 < x < 0.5$ and phase 1 ($\rho^+ = 2$, $\mu^+ = 3$) occupying $0.5 < x < 0.8$. The analytic solution is $u = 0$, $p = 0$, and

$$v_1 = \frac{v_0\mu^-(x_2 - x_1) + v_2\mu^+(x_1 - x_0)}{\mu^-(x_2 - x_1) + \mu^+(x_1 - x_0)}, \qquad v = \begin{cases} v_0 + \frac{x - x_0}{x_1 - x_0}(v_1 - v_0) & x \leqslant x_1, \\ v_1 + \frac{x - x_1}{x_2 - x_1}(v_2 - v_1) & x > x_1. \end{cases} \tag{76}$$

The initial velocity is the analytic solution. This test demonstrates that the method correctly (and sharply) handles discontinuities in viscosity. Convergence results are summarized in Fig. 13.

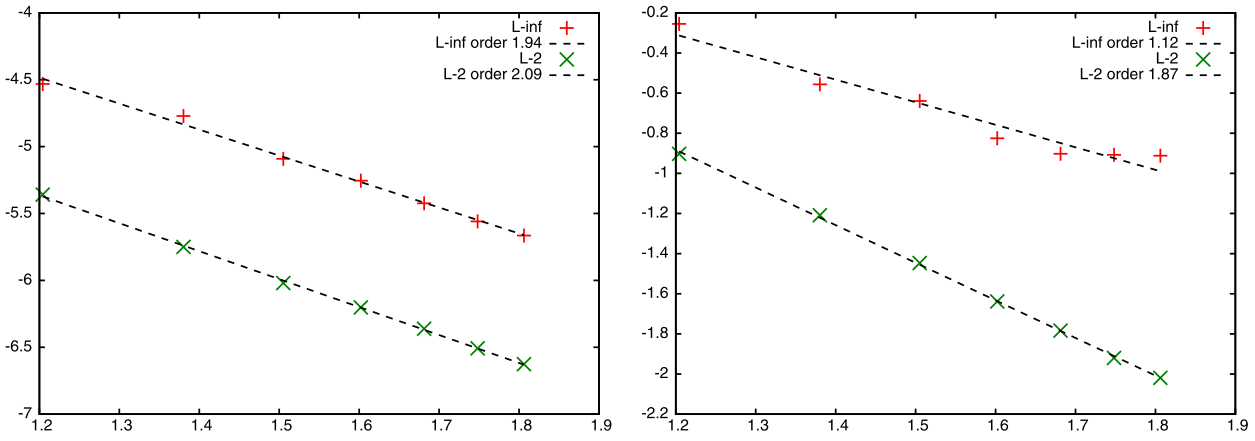
4.7. Parasitic currents

In this test, we check for convergence of parasitic currents in the case of a stationary circle with surface tension. The fluid domain is $[0\,\text{m}, 0.01\ \text{m}]\times[0\ \text{m}, 0.01\ \text{m}]$, with periodic boundary conditions and an initially circular interface (center



(a) Velocity error (b) Pressure error

Fig. 14. A stationary circle is run with surface tension to test convergence of parasitic currents. Errors in velocity and pressure (y-axis, shown log base 10) are plotted against resolutions from 16 to 256 by increments of 8 (x-axis, shown log base 10). We estimate the velocity to be order 1.96 in L^∞ and 2.04 in L^2. For pressure, we obtained 1.13 in L^∞ and 1.97 in L^2.



(a) Velocity error (b) Pressure error

Fig. 15. A stationary sphere is run with surface tension to test convergence of parasitic currents. Errors in velocity and pressure (y-axis, shown log base 10) are plotted against resolutions from 16 to 64 by increments of 8 (x-axis, shown log base 10). We estimate the velocity to be order 1.94 in L^∞ and 2.09 in L^2. For pressure, we obtained 1.12 in L^∞ and 1.87 in L^2.

(0.005 m, 0.005 m), radius 0.003 m). We simulate glycerin inside the circle ($\rho^- = 1261\ \mathrm{kg\,m^{-2}}$, $\mu^- = 1.4746\ \mathrm{kg\,s^{-1}}$) and a generic light fluid outside ($\rho^+ = 1\ \mathrm{kg\,m^{-2}}$, $\mu^+ = 1\ \mathrm{kg\,s^{-1}}$, similar in density to air but more viscous). The interface is evolved with the level set method. Convergence results are shown in Fig. 14.

4.8. Parasitic currents – 3D

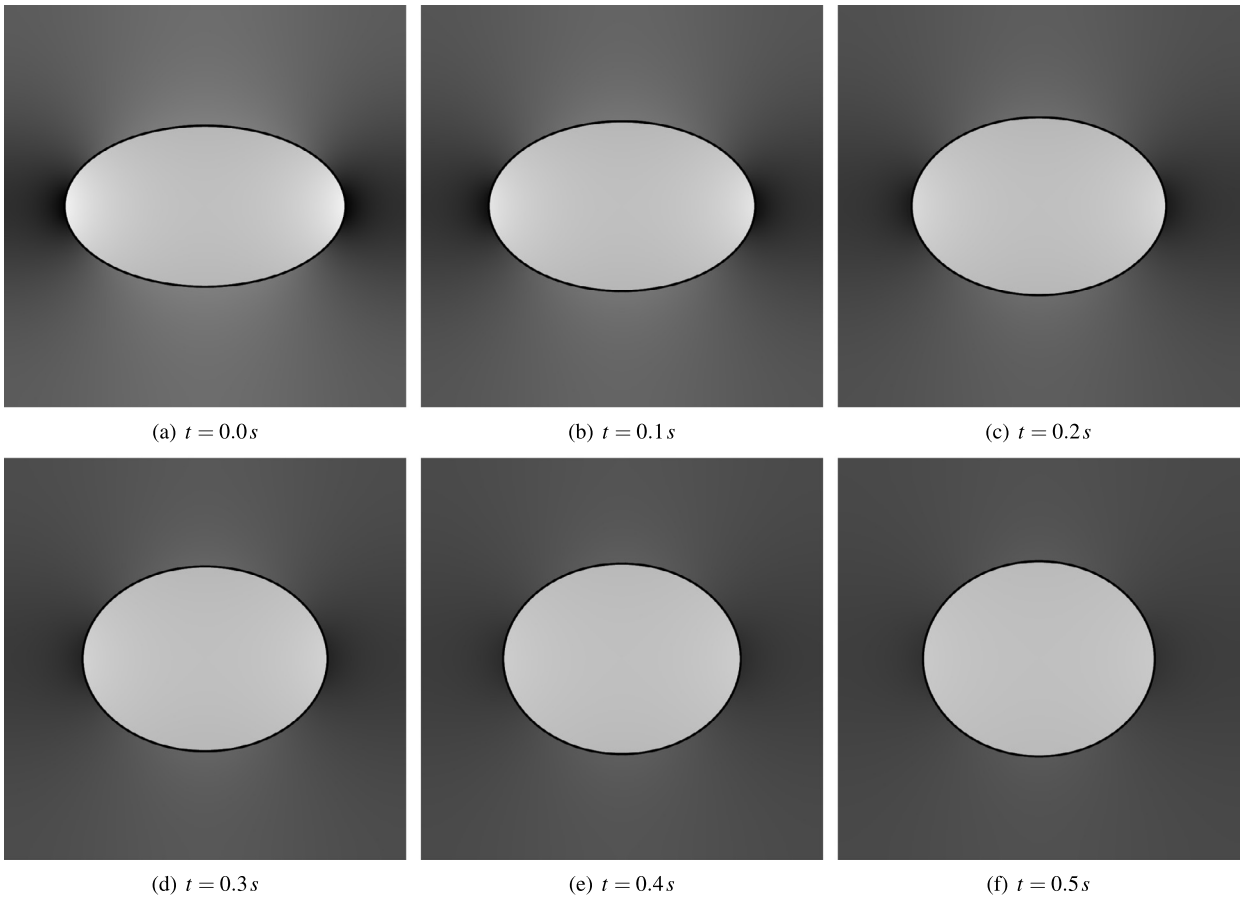
This test is a 3D analogue of Section 4.7. The fluid domain is [0 m, 0.01 m] × [0 m, 0.01 m] × [0 m, 0.01 m], with periodic boundary conditions and an initially spherical interface (center (0.005 m, 0.005 m, 0.005 m), radius 0.003 m). Glycerin is inside ($\rho^- = 1261\ \mathrm{kg\,m^{-3}}$, $\mu^- = 1.4746\ \mathrm{kg\,m^{-1}\,s^{-1}}$), and a light fluid is outside ($\rho^+ = 1\ \mathrm{kg\,m^{-3}}$, $\mu^+ = 1\ \mathrm{kg\,m^{-1}\,s^{-1}}$). The interface is evolved with the level set method, and the results are shown in Fig. 15.

4.9. Relaxing ellipse

The tests up to this point have been analytic tests. Here we run a relaxing ellipse test similar to the one performed in [1]. Two fluids are separated by an interface in the initial shape of an ellipse. The fluid domain is [−1 m, 1 m] × [−1 m, 1 m], with periodic boundary conditions. The ellipse is centered in the domain with major axis 1.4 m and minor axis 0.8 m. The inside fluid has parameters $\rho^- = 0.01\ \mathrm{kg\,m^{-2}}$ and $\mu^- = 1\ \mathrm{kg\,s^{-1}}$. The outside fluid has parameters $\rho^+ = 0.02\ \mathrm{kg\,m^{-2}}$ and

Table 1
Order of convergence for 2D relaxing ellipse. Note that pressure is too noisy in L^∞ to give a meaningful convergence estimate. On the other hand, the pressure error is transitioning to second in L^2. (The transition to second is not merely noise. Noisy L^∞ and convergence orders consistent with second order in L^2 were also observed when this simulation was run with different parameters.) This suggests that, while the pressure may be noisy, it is converging.

Resolutions compared			Order ($\boldsymbol{u}$)		Order (p)	
			L^∞	L^2	L^∞	L^2
8	16	32	1.428	1.567	0.990	1.078
16	32	64	2.724	2.948	0.326	1.084
24	48	96	4.178	3.568	0.105	1.317
32	64	128	2.521	2.682	0.216	1.684
48	96	192	2.185	2.074	−0.358	1.638
64	128	256	2.453	2.112	0.457	2.334



(a) $t = 0.0\,s$ (b) $t = 0.1\,s$ (c) $t = 0.2\,s$

(d) $t = 0.3\,s$ (e) $t = 0.4\,s$ (f) $t = 0.5\,s$

Fig. 16. Pressure and interface configuration for the relaxing ellipse of Section 4.9. Dark regions have lower pressure and lighter regions have higher pressure.

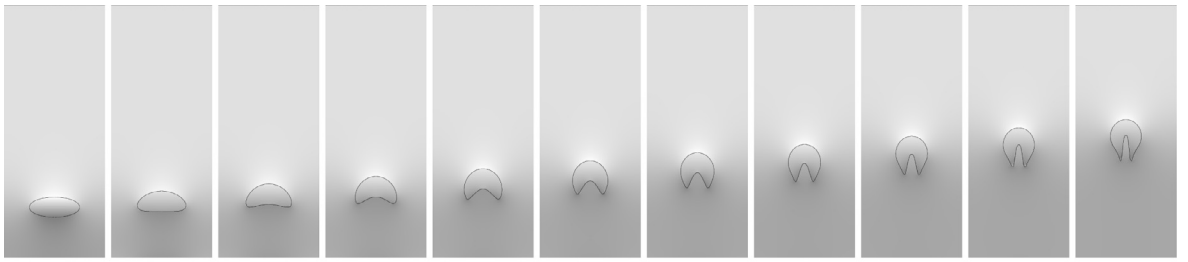
$\mu^+ = 3\ \mathrm{kg\,s^{-1}}$. The surface tension coefficient is $10\ \mathrm{kg\,m\,s^2}$. The simulations were run with time step $\Delta t = (0.01\ \mathrm{m^{-1}\,s})\Delta x$ until time $T = 0.05$ s. Convergence orders are shown in Table 1. Snapshots from the simulation are shown in Fig. 16.

4.10. Relaxing ellipsoid – 3D

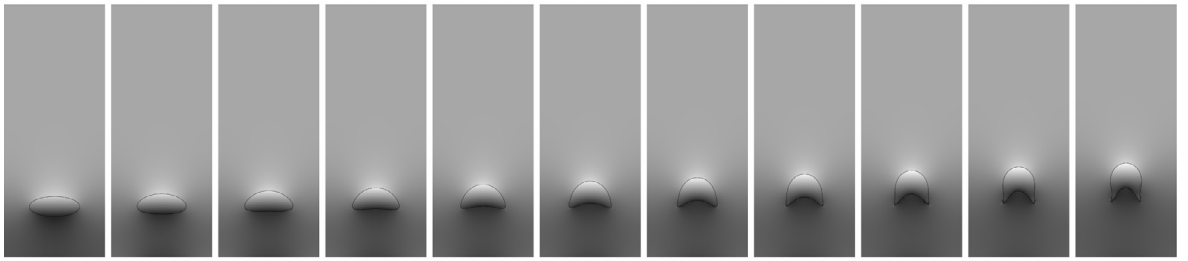
This relaxing ellipsoid test is a 3D analogue of Section 4.9. Two fluids are separated by an interface in the initial shape of an ellipsoid. The fluid domain is $[-1\ \mathrm{m}, 1\ \mathrm{m}] \times [-1\ \mathrm{m}, 1\ \mathrm{m}] \times [-1\ \mathrm{m}, 1\ \mathrm{m}]$, with periodic boundary conditions. The ellipsoid is centered in the domain with major and minor axes 1.4 m, 0.8 m, and 0.8 m. The inside fluid has parameters $\rho^- = 0.01\ \mathrm{kg\,m^{-3}}$ and $\mu^- = 1\ \mathrm{kg\,m^{-1}\,s^{-1}}$. The outside fluid has parameters $\rho^+ = 0.02\ \mathrm{kg\,m^{-3}}$ and $\mu^+ = 3\ \mathrm{kg\,m^{-1}\,s^{-1}}$. The surface tension coefficient is $10\ \mathrm{kg\,s^2}$. The simulations were run with time step $\Delta t = (0.01\ \mathrm{m^{-1}\,s})\Delta x$ until time $T = 0.015$ s. Convergence orders are shown in Table 2.

Table 2
Order of convergence for 3D relaxing ellipse.

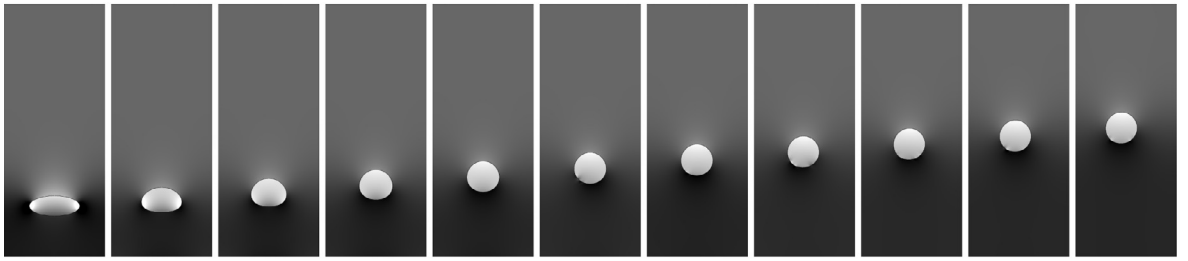
Resolutions compared			Order ($\boldsymbol{u}$)		Order (p)	
			L^∞	L^2	L^∞	L^2
8	16	32	2.779	3.012	1.856	2.188
16	32	64	3.521	3.572	0.773	1.646



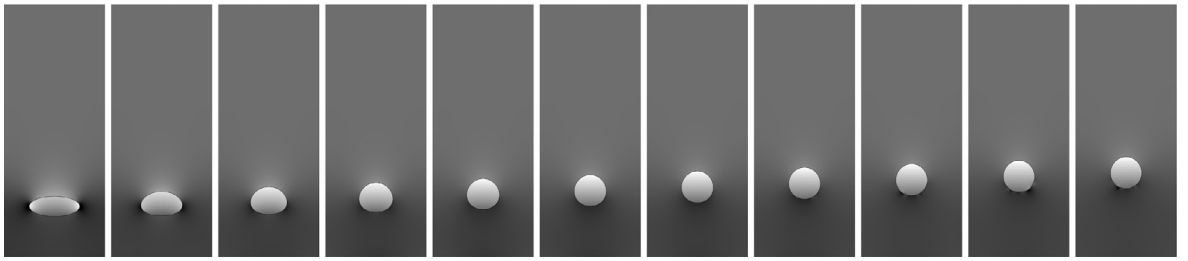
(a) $\mu^- = 3$, $\mu^+ = 1$, no surface tension



(b) $\mu^- = 1$, $\mu^+ = 3$, no surface tension



(c) $\mu^- = 3$, $\mu^+ = 1$, surface tension

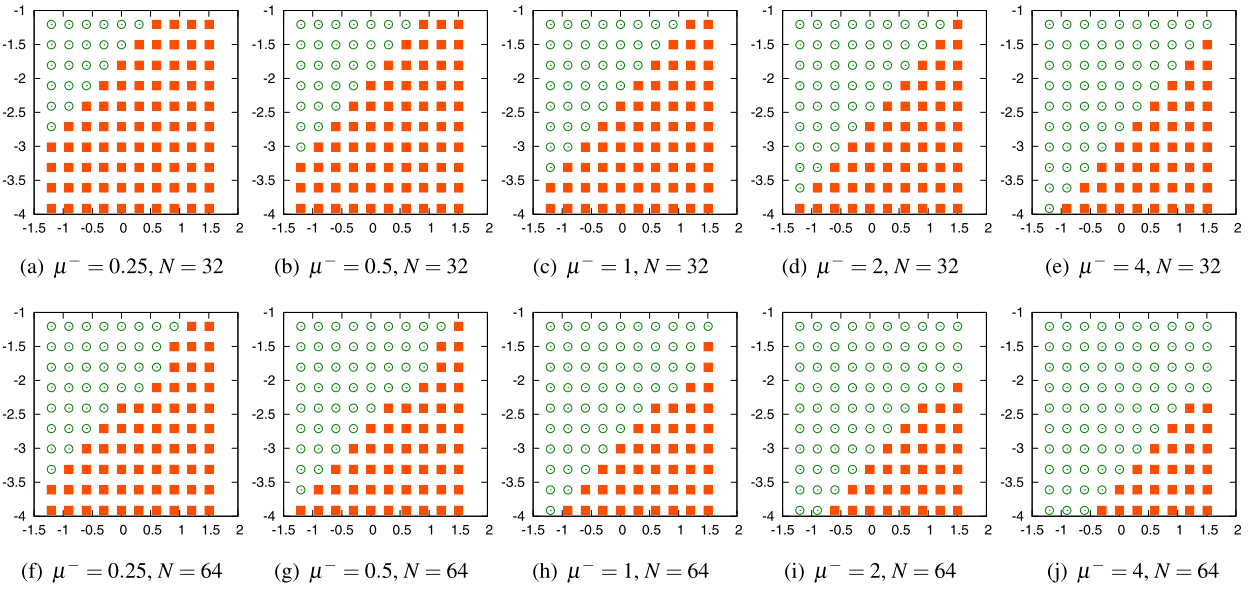


(d) $\mu^- = 1$, $\mu^+ = 3$, surface tension

Fig. 17. Pressure and interface configurations for the four rising bubble simulations described in Section 4.11 at $t = 0.0, 1.0, 2.0, \ldots, 9.0, 10.0\,s$. For each simulation, dark regions correspond to lower pressure and lighter regions have higher pressure.

4.11. Rising bubbles

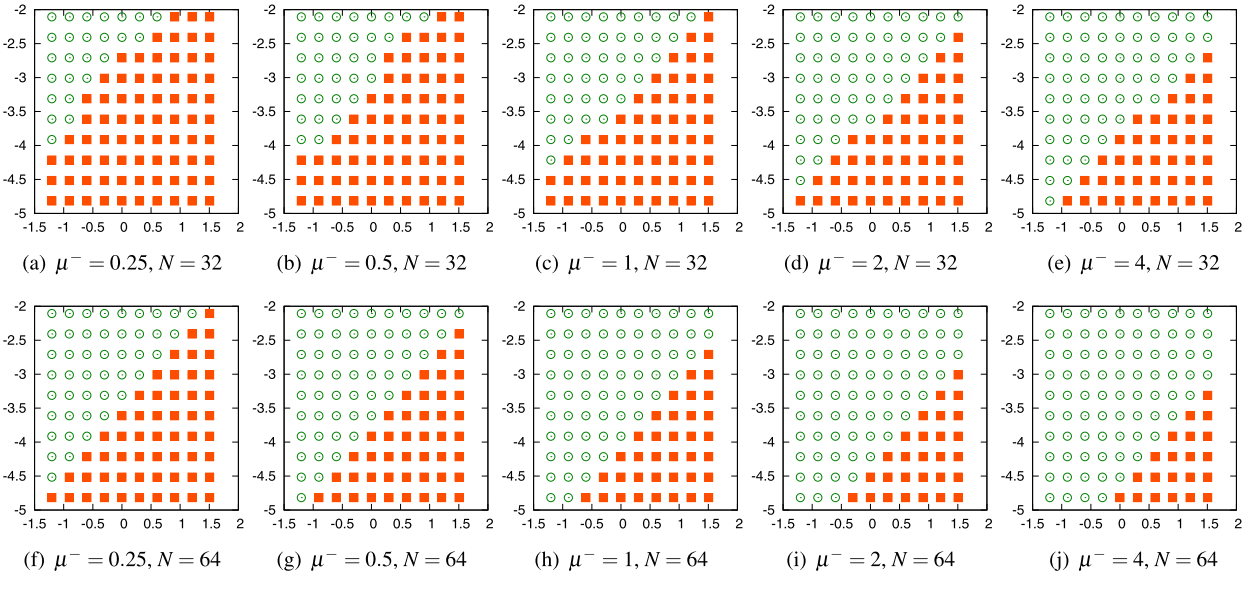
We used our algorithm to simulate a rising fluid bubble surrounded by fluid of differing viscosity and density. We assumed a uniform gravitational acceleration equal to $g = 9.8$, with the fluid densities being $\rho^- = 1$ inside the interface and $\rho^+ = 2$ outside the interface for all simulations. The interface is an ellipse of major radius $a = 0.5$ and minor radius $b = 0.2$ in a domain $[-1, 1] \times [0, 5]$, and we center it at $(0, 1)$. The top and bottom have zero Dirichlet boundary conditions on the top and bottom, and the sides are periodic. We simulate examples where the inner viscosity $\mu^- = 3$ is greater than



(a) $\mu^- = 0.25, N = 32$ (b) $\mu^- = 0.5, N = 32$ (c) $\mu^- = 1, N = 32$ (d) $\mu^- = 2, N = 32$ (e) $\mu^- = 4, N = 32$

(f) $\mu^- = 0.25, N = 64$ (g) $\mu^- = 0.5, N = 64$ (h) $\mu^- = 1, N = 64$ (i) $\mu^- = 2, N = 64$ (j) $\mu^- = 4, N = 64$

Fig. 18. These plots show the stability of our method on one of our analytic tests. Each point on the (x, y) grid corresponds to a simulation with $\rho^- = 10^x$, $\rho^+ = 2$, $\mu^+ = 2$, $\Delta x = 2\pi/N$, and $\Delta t = 10^y$. Circles represent stable simulations, and squares represent simulations that were unstable.



(a) $\mu^- = 0.25, N = 32$ (b) $\mu^- = 0.5, N = 32$ (c) $\mu^- = 1, N = 32$ (d) $\mu^- = 2, N = 32$ (e) $\mu^- = 4, N = 32$

(f) $\mu^- = 0.25, N = 64$ (g) $\mu^- = 0.5, N = 64$ (h) $\mu^- = 1, N = 64$ (i) $\mu^- = 2, N = 64$ (j) $\mu^- = 4, N = 64$

Fig. 19. These plots show the stability of our method on the relaxing ellipse with varying parameters. Each point on the (x, y) grid corresponds to a simulation with $\rho^- = 10^x$, $\rho^+ = 2\rho^-$, $\mu^+ = 3\mu^-$, $\Delta x = 2/N$, and $\Delta t = 10^y$. Circles represent stable simulations, and squares represent simulations that were unstable.

the outer viscosity $\mu^+ = 1$, and examples where the inner viscosity $\mu^- = 1$ is less than the outer viscosity $\mu^+ = 3$. These values are similar to those used in the rising bubble example in [1]. For each of these viscosity value pairs, we simulate a rising bubble without (Figs. 17(a) and 17(b)) and with (Figs. 17(c) and 17(d)) surface tension. In the simulations of Figs. 17(c) and 17(d), we use a surface tension force coefficient equal to that used in the other surface tension examples. These results are qualitatively similar to the rising bubble test in [1], which used the same parameters but periodic boundary conditions on the top and bottom.

4.12. Stability tests

To examine stability, we consider two different examples: the analytic example in Section 4.4 and the relaxing ellipse described in Section 4.9. For both examples, we run simulations for a variety of parameters that affect our stability. In each case, we choose five values of μ^-, ten values of ρ^-, two values of $\Delta x = 2/N$, and ten values of Δt, each sampled

by powers of two. Each of these 1000 simulations is classified as stable or unstable. A simulation is classified as stable if it completes without producing velocities larger than 10 (Section 4.4) or 2 (Section 4.9). In practice, the classification was quite unambiguous most of the time (most simulations that are unstable simply explode). The rather low cutoff is much smaller than what would normally be considered 'blowup', and errs on the side of classifying simulations as unstable which do not blow up but still have large uncharacteristic variations in velocity.

For both examples, we can demonstrate the stability characteristics of our method as a function of μ, ρ, Δx, and Δt. For the analytic test in Section 4.4 we vary Δt, ρ, and μ over a range of values, as described in Fig. 18. We also use two different values of Δx. For this simulation, the transition between stable and unstable occurs at $\frac{\Delta t\mu}{\rho\Delta x^2} \approx 0.2$. When this threshold is reached, instabilities begin to develop at the interface between the two phases. Note that this stability criterion places a *lower bound* on Δt. Results are shown in Fig. 18.

For the relaxing ellipse stability test, we use the setup in Section 4.9, sampling ranges of ρ, μ, Δt, and Δx as before. For this simulation, the transition between stable and unstable occurs at $\frac{\Delta t\mu}{\rho\Delta x^2} \approx 0.1$. Instabilities, when they occur, develop at the interface. Results are shown in Fig. 19.

5. Conclusion

In this work we presented a second order accurate method for the Navier–Stokes equations which can incorporate immersed interfaces, discontinuous fluid properties, and various boundary conditions. We considered examples in which both the fluid viscosities and densities are discontinuous across the interface, examples implementing each type of boundary condition listed in (1)–(8), and examples showing many combinations of these boundary conditions interacting. We demonstrated the ability of our method to handle interface forces by considering examples with surface tension. Our method yields a symmetric indefinite linear system of equations.

We discussed the two primary limitations of our method. The first limitation of our method, its additional stability restriction, effectively restricts its use to problems involving low or moderate Reynolds numbers (Re up to about 20 in practice). The method presented is not suitable for high Reynolds number flows. The second limitation is the KKT system, for which we currently lack an effective preconditioner. We leave the problem of preconditioning for future work.

Acknowledgements

All authors were partially supported by NSF (DMS-0502315, DMS-0652427, CCF-0830554), DOE (09-LR-04-116741-BERA), ONR (N000140310071, N000141010730, N000141210834) and Intel STCVisual Computing Grant (20112360).

Appendix A. Continuous weak form of implicit equation

Here we derive the continuous weak form of (11) that we discretized in Section 3.4.1. For the purposes of this derivation, we assume that we have an interface at Γ but no other non-periodic boundaries.

We begin by taking a dot product of both sides of (11) by a test function $\boldsymbol{w}$, then integrating both sides over $\Omega\backslash\Gamma$ to get

$$\int_{\Omega\backslash\Gamma} \alpha \boldsymbol{w}\cdot\left(\boldsymbol{u}-\boldsymbol{u}^{\star}\right)dV = \int_{\Omega\backslash\Gamma} \boldsymbol{w}\cdot(\nabla\cdot\sigma+\boldsymbol{f})\,dV, \tag{77}$$

where we have used $\boldsymbol{u}=\boldsymbol{u}^{n+1}$ for conciseness. Integration by parts yields

$$\int_{\Omega\backslash\Gamma} \boldsymbol{w}\cdot(\nabla\cdot\sigma)\,dV = \int_{\Omega\backslash\Gamma} \nabla\cdot(\boldsymbol{w}\cdot\sigma)-\nabla\boldsymbol{w}:\sigma\,dV = -\int_{\Gamma}[\boldsymbol{w}\cdot\sigma]\cdot\boldsymbol{n}\,dA - \int_{\Omega\backslash\Gamma}\nabla\boldsymbol{w}:\sigma\,dV, \tag{78}$$

where $\boldsymbol{n}$ is the outward normal from Ω^- and $[\boldsymbol{w}]=\boldsymbol{w}^+-\boldsymbol{w}^-$ denotes the jump in $\boldsymbol{w}$ across the interface. Then,

$$\int_{\Omega\backslash\Gamma}\alpha\boldsymbol{w}\cdot\boldsymbol{u}\,dV + \int_{\Omega\backslash\Gamma}\nabla\boldsymbol{w}:\sigma\,dV + \int_{\Gamma}[\boldsymbol{w}\cdot\sigma]\cdot\boldsymbol{n}\,dA = \int_{\Omega\backslash\Gamma}\alpha\boldsymbol{w}\cdot\boldsymbol{u}^{\star}\,dV + \int_{\Omega\backslash\Gamma}\boldsymbol{w}\cdot\boldsymbol{f}\,dV. \tag{79}$$

Utilizing symmetry,

$$\int_{\Omega\backslash\Gamma}\nabla\boldsymbol{w}:\sigma\,dV = \int_{\Omega\backslash\Gamma}\nabla\boldsymbol{w}:\left(\mu\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^{T}\right)-p\boldsymbol{I}\right)dV \tag{80}$$

$$= \int_{\Omega\backslash\Gamma}\mu\nabla\boldsymbol{w}:\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^{T}\right)dV - \int_{\Omega\backslash\Gamma}\nabla\cdot\boldsymbol{w}p\,dV \tag{81}$$

$$= \int_{\Omega\setminus\Gamma} \frac{\mu}{2}\left(\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T\right) : \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) dV - \int_{\Omega\setminus\Gamma} \nabla \cdot \boldsymbol{w} p \, dV. \tag{82}$$

Using the identity $[\boldsymbol{w}\cdot\sigma] = [\boldsymbol{w}]\cdot\overline{\sigma} + \overline{\boldsymbol{w}}\cdot[\sigma]$, where $\overline{\sigma} = \frac{1}{2}(\sigma^+ + \sigma^-)$ and $\overline{\boldsymbol{w}} = \frac{1}{2}(\boldsymbol{w}^+ + \boldsymbol{w}^-)$,

$$\int_\Gamma [\boldsymbol{w}\cdot\sigma]\cdot\boldsymbol{n}\, dA = \int_\Gamma [\boldsymbol{w}]\cdot\overline{\sigma}\cdot\boldsymbol{n}\, dA + \int_\Gamma \overline{\boldsymbol{w}}\cdot[\sigma]\cdot\boldsymbol{n}\, dA = \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA + \int_\Gamma \overline{\boldsymbol{w}}\cdot\hat{\boldsymbol{f}}\, dA, \tag{83}$$

where $\hat{\boldsymbol{f}} = [\sigma]\cdot\boldsymbol{n}$ is known but $\boldsymbol{q} = \overline{\sigma}\cdot\boldsymbol{n}$ must be treated as a degree of freedom since its value will not in general be known. Combining these with (79) yields

$$\int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u}\, dV + \int_{\Omega\setminus\Gamma} \frac{\mu}{2}\left(\nabla \boldsymbol{w} + \nabla \boldsymbol{w}^T\right) : \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right) dV - \int_{\Omega\setminus\Gamma} p\nabla\cdot\boldsymbol{w}\, dV + \int_\Gamma [\boldsymbol{w}]\cdot\boldsymbol{q}\, dA$$
$$= \int_{\Omega\setminus\Gamma} \alpha \boldsymbol{w}\cdot\boldsymbol{u}^\star\, dV + \int_{\Omega\setminus\Gamma} \boldsymbol{w}\cdot\boldsymbol{f}\, dV + \int_\Gamma \overline{\boldsymbol{w}}\cdot\hat{\boldsymbol{f}}. \tag{84}$$

Introducing test functions λ and $\boldsymbol{v}$, the weak forms for (2) and (3) are

$$\int_{\Omega\setminus\Gamma} \lambda\nabla\cdot\boldsymbol{u}\, dV = 0 \tag{85}$$

$$\int_\Gamma \boldsymbol{v}\cdot[\boldsymbol{u}]\, dA = \int_\Gamma \boldsymbol{v}\cdot\boldsymbol{a}_i\, dA. \tag{86}$$

Eqs. (84)–(86) constitute our weak form of the Navier–Stokes problem.

References

[1] D.C. Assêncio, J.M. Teran, A second order virtual node algorithm for Stokes flow problems with interfacial forces, discontinuous material parameters and irregular domains, J. Comput. Phys. 250 (1) (2013) 77–105.
[2] J. Bedrossian, J.H. von Brecht, S. Zhu, E. Sifakis, J.M. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains, J. Comput. Phys. 229 (2010) 6405–6426.
[3] J. Hellrung, L. Wang, E. Sifakis, J.M. Teran, A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions, J. Comput. Phys. 231 (4) (2012) 2015–2048.
[4] Y. Zhu, Y. Wang, J. Hellrung, A. Cantarero, E. Sifakis, J. Teran, A second-order virtual node algorithm for nearly incompressible linear elasticity in irregular domains, J. Comput. Phys. 231 (21) (2012) 7092–7117.
[5] D. Xiu, G.E. Karniadakis, A semi-Lagrangian high-order method for Navier–Stokes equations, J. Comput. Phys. 172 (2) (2001) 658–684.
[6] A. Lew, G. Buscaglia, A discontinuous-Galerkin-based immersed boundary method, Int. J. Numer. Methods Eng. 76 (4) (2008) 427–454.
[7] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (1972) 252–271.
[8] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[9] D.M. McQueen, C.S. Peskin, A three-dimensional computational method for blood flow in the heart II. Contractile fibers, J. Comput. Phys. 82 (1989) 289–297.
[10] C.S. Peskin, D.M. McQueen, A three-dimensional computational method for blood flow in the heart I. Immersed elastic fibers in a viscous incompressible fluid, J. Comput. Phys. 81 (2) (1989) 372–405.
[11] C.S. Peskin, D.M. McQueen, Modeling prosthetic heart valves for numerical analysis of blood flow in the heart, J. Comput. Phys. 37 (1) (1980) 113–132.
[12] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[13] Z. Li, M.-C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2) (2001) 822–842.
[14] B.E. Griffith, R.D. Hornung, D.M. McQueen, C.S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, J. Comput. Phys. 223 (1) (2007) 10–49.
[15] C.S. Peskin, B.F. Printz, Improved volume conservation in the computation of flows with immersed elastic boundaries, J. Comput. Phys. 105 (1993) 33–46.
[16] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comput. Phys. 156 (1999) 209–240.
[17] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, J. Comput. Phys. 174 (2001) 345–380.
[18] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, J. Comput. Phys. 227 (10) (2008) 4825–4852.
[19] J. Seo, R. Mittal, A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations, J. Comput. Phys. 230 (19) (2011) 7347–7363.
[20] S. Xu, Z.J. Wang, An immersed interface method for simulating the interaction of a fluid with moving boundaries, J. Comput. Phys. 216 (2006) 454–493.
[21] D.V. Le, B.C. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, J. Comput. Phys. 220 (2006) 109–138.
[22] L. Lee, R.J. LeVeque, An immersed interface method for incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (2003) 832–856.
[23] K. Ito, M.-C. Lai, Z. Li, A well-conditioned augmented system for solving Navier–Stokes equations in irregular domains, J. Comput. Phys. 228 (7) (2009) 2616–2628.

[24] Z. Li, K. Ito, M.-C. Lai, An augmented approach for Stokes equations with a discontinuous viscosity and singular forces, Comput. Fluids 36 (3) (2007) 622–635.
[25] Z. Tan, D.V. Le, K.M. Lim, B.C. Khoo, An immersed interface method for the incompressible Navier–Stokes equations with discontinuous viscosity across the interface, SIAM J. Sci. Comput. 31 (2009) 1798–1819.
[26] Z. Tan, D.V. Le, Z. Li, K.M. Lim, B.C. Khoo, An immersed interface method for solving incompressible viscous flows with piecewise constant viscosity across a moving elastic membrane, J. Comput. Phys. 227 (2008) 9955–9983.
[27] Z. Li, M.-C. Lai, New finite difference methods based on IIM for inextensible interfaces in incompressible flows, East Asian J. Appl. Math. 1 (2011) 155–171.
[28] X. Zhong, A new high-order immersed interface method for solving elliptic equations with imbedded interface of discontinuity, J. Comput. Phys. 225 (1) (2007) 1066–1099.
[29] Y. Zhou, S. Zhao, M. Feig, High order matched interface and boundary (MIB) schemes for elliptic equations with discontinuous coefficients and singular sources, J. Comput. Phys. 213 (2006) 1–30.
[30] S. Xu, Z.J. Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Eng. 197 (25–28) (2008) 2068–2086.
[31] Z. Li, P. Song, Adaptive mesh refinement techniques for the immersed interface method applied to flow problems, Comput. Struct. 122 (2013) 249–258.
[32] Z. Tan, K. Limb, B. Khoo, A Jacobian-free-based IIM for incompressible flows involving moving interfaces with Dirichlet boundary conditions, Int. J. Numer. Methods Eng. 83 (2010) 508–536.
[33] K. Xia, M. Zhan, G. Wei, MIB method for elliptic equations with multi-material interfaces, J. Comput. Phys. 230 (2011) 4588–4615.
[34] Y. Zhou, J. Liu, L. Harry, A matched interface and boundary method for solving multi-flow Navier–Stokes equations with applications to geodynamics, J. Comput. Phys. 231 (2012) 223–242.
[35] N. Molino, Z. Bao, R. Fedkiw, A virtual node algorithm for changing mesh topology during simulation, ACM Trans. Graph. 23 (2004) 385–392.
[36] X.-D. Liu, R.P. Fedkiw, M. Kang, A boundary condition capturing method for Poisson's equation on irregular domains, J. Comput. Phys. 160 (2000) 151–178.
[37] M. Kang, R.P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, J. Sci. Comput. 15 (2000) 323–360.
[38] M. Hyman, Non-iterative numerical solution of boundary-value problems, Appl. Sci. Res., Sect. B 2 (1) (1952) 325–351.
[39] V. Saul'ev, On solving boundary value problems on high-performance computers by fictitious domain methods, Sib. Math. J. 4 (1963) 912–925.
[40] R. Glowinski, T. Pan, T. Hesla, D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, Int. J. Multiph. Flow 25 (5) (1999) 755–794.
[41] R. Glowinski, T.-W. Pan, J. Periaux, A fictitious domain method for external incompressible viscous flow modeled by Navier–Stokes equations, Comput. Methods Appl. Mech. Eng. 112 (1–4) (1994) 133–148.
[42] R. Glowinski, T. Pan, T. Hesla, D. Joseph, J. Periaux, A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flows, J. Comput. Phys. 169 (2001) 363–426.
[43] L. Parussini, V. Pediroda, Fictitious domain approach with hp-finite element approximation for incompressible fluid flow, J. Comput. Phys. 228 (2009) 3891–3910.
[44] L. Parussini, Fictitious domain approach via Lagrange multipliers with least squares spectral element method, J. Sci. Comput. 37 (2008) 316–335.
[45] F. Bertrand, P. Tanguy, F. Thibault, A three-dimensional fictitious domain method for incompressible fluid flow problems, Int. J. Numer. Methods Fluids 25 (6) (1997) 719–736.
[46] J.M. Teran, C.S. Peskin, Tether force constraints in Stokes flow by the immersed boundary method on a periodic domain, SIAM J. Sci. Comput. 31 (5) (2009) 3404–3416.
[47] G. Biros, L. Ying, D. Zorin, A fast solver for the Stokes equations with distributed forces in complex geometries, J. Comput. Phys. 193 (1) (2004) 317–348.
[48] V. Rutka, A staggered grid-based explicit jump immersed interface method for two-dimensional Stokes flows, Int. J. Numer. Methods Fluids 57 (10) (2008) 1527–1543.
[49] C. Daux, N. Moës, J. Dolbow, N. Sukumar, T. Belytschko, Arbitrary branched and intersecting cracks with the extended finite element method, Int. J. Numer. Methods Eng. 48 (12) (2000) 1741–1760.
[50] N. Sukumar, D. Chopp, N. Moës, T. Belytschko, Modeling holes and inclusions by level sets in the extended finite-element method, Comput. Methods Appl. Mech. Eng. 190 (46–47) (2001) 6183–6200.
[51] A. Almgren, J. Bell, P. Colella, T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, SIAM J. Sci. Comput. 18 (5) (1997) 1289–1309.
[52] N. Moës, E. Béchet, M. Tourbier, Imposing Dirichlet boundary conditions in the extended finite element method, Int. J. Numer. Methods Eng. 67 (12) (2006) 1641–1669.
[53] J. Dolbow, A. Devan, Enrichment of enhanced assumed strain approximations for representing strong discontinuities: addressing volumetric incompressibility and the discontinuous patch test, Int. J. Numer. Methods Eng. 59 (1) (2004) 47–67.
[54] G. Wagner, N. Moës, W. Liu, T. Belytschko, The extended finite element method for rigid particles in Stokes flow, Int. J. Numer. Methods Eng. 51 (3) (2001) 293–313.
[55] R. Becker, E. Burman, P. Hansbo, A Nitsche extended finite element method for incompressible elasticity with discontinuous modulus of elasticity, Comput. Methods Appl. Mech. Eng. 198 (41–44) (2009) 3352–3360.
[56] A. Coppola-Owen, R. Codina, Improving Eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions, Int. J. Numer. Methods Fluids 49 (12) (2005) 1287–1304.
[57] J. Chessa, T. Belytschko, An extended finite element method for two-phase fluids, J. Appl. Math. 70 (1) (2003) 10–17.
[58] A. Gerstenberger, W. Wall, An extended finite element method/Lagrange multiplier based approach for fluid–structure interaction, Comput. Methods Appl. Mech. Eng. 197 (19–20) (2008) 1699–1714.
[59] S. Marella, S. Krishnan, H. Liu, H. Udaykumar, Sharp interface Cartesian grid method I: an easily implemented technique for 3D moving boundary computations, J. Comput. Phys. 210 (1) (2005) 1–31.
[60] Y. Ng, C. Min, F. Gibou, An efficient fluid–solid coupling algorithm for single-phase flows, J. Comput. Phys. 228 (23) (2009) 8807–8829.
[61] F. Gibou, C. Min, Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions, J. Comput. Phys. 231 (8) (2012) 3246–3263.
[62] T.D. Aslam, A partial differential equation approach to multidimensional extrapolation, J. Comput. Phys. 193 (1) (2004) 349–355.
[63] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, J. Comput. Phys. 183 (1) (2002) 83–116.
[64] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471.
[65] X.-D. Liu, S. Osher, T. Chan, Weighted essentially non-oscillatory schemes, J. Comput. Phys. 115 (1) (1994) 200–212.
[66] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[67] A. Robinson-Mosher, C. Schroeder, R. Fedkiw, A symmetric positive definite formulation for monolithic fluid structure interaction, J. Comput. Phys. 230 (4) (2011) 1547–1566.