

A Second Order Virtual Node Algorithm for Stokes Flow Problems with Interfacial Forces and Irregular Domains

Diego C. Assêncio^{*a}, Joseph M. Teran^b

^a University of California, Los Angeles, Department of Physics and Astronomy
430 Portola, Physics and Astronomy Building
Los Angeles, CA 90095

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

Abstract

We present a numerical method for the solution of the Stokes equations that handles both interfacial discontinuities and geometrically irregular flow domains. The method is efficient, easy to implement, second order accurate and yields discretely divergence free velocities. We discretize the equations using an embedded approach on a uniform MAC-grid employing virtual nodes at interfaces and boundaries. Interfaces and boundaries are represented with a hybrid Lagrangian/level set method. We rewrite the Stokes equations as three Poisson equations and use the techniques developed in Bedrossian et al. (2010) [1] to impose jump and boundary conditions. We also use a final Poisson equation to enforce a discrete divergence-free condition. All four linear systems involved are symmetric positive definite with three of the four having the standard 5-point Laplace stencil everywhere. Numerical results indicate second order accuracy in L^∞ for both velocities and pressure.

Key words: Stokes flow, creeping flow, Virtual Node Algorithms, interface problems, flow in irregular domains, Cartesian grids

1. Introduction

We consider the Stokes equations for two-phase, highly viscous incompressible flow in irregular domains:

$$\nabla \cdot \sigma = \mu \Delta \mathbf{u} - \nabla p = -\mathbf{f}, \quad \mathbf{x} \in \Omega \setminus \Gamma \quad (1)$$

$$\nabla \cdot \mathbf{u} = 0, \quad \mathbf{x} \in \Omega \setminus \Gamma \quad (2)$$

$$\mathbf{u}(\mathbf{x}) = \mathbf{a}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega \quad (3)$$

$$[\mathbf{u}](\mathbf{x}) = \mathbf{0}, \quad \mathbf{x} \in \Gamma \quad (4)$$

$$[\sigma \cdot \mathbf{n}](\mathbf{x}) = \mathbf{f}^i, \quad \mathbf{x} \in \Gamma \quad (5)$$

Here p is the pressure, $\mathbf{u} = (u, v)$ is the fluid velocity, $\sigma = \mu \left(\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial \mathbf{u}^T}{\partial \mathbf{x}} \right) - p \mathbf{I}$ is the fluid stress, Γ is the interface between the two phases, \mathbf{f} is the body force density and \mathbf{f}^i is the force per unit length supported on the interface between the two phases (e.g. surface tension). The interface Γ is generally a codimension one closed curve that divides the domain into an interior region Ω^- and an exterior region Ω^+ such that $\Omega = \Omega^- \cup \Omega^+ \cup \Gamma$ (see figure 1(a)). We let \mathbf{n} denote the outward unit normal to Ω^- at a point $\mathbf{x} \in \Gamma$ and define $[\nu](\mathbf{x}) := \nu^+(\mathbf{x}) - \nu^-(\mathbf{x}) = \lim_{\epsilon \rightarrow 0^+} \nu(\mathbf{x} + \epsilon \mathbf{n}) - \lim_{\epsilon \rightarrow 0^+} \nu(\mathbf{x} - \epsilon \mathbf{n})$

^{*}Correspondence to Diego C. Assêncio,

Email addresses: dassencio@physics.ucla.edu (Diego C. Assêncio), jteran@math.ucla.edu (Joseph M. Teran)

as the “jump” of the quantity v across the interface Γ . Unless otherwise stated, we assume the curves Γ and $\partial\Omega$ are smooth.

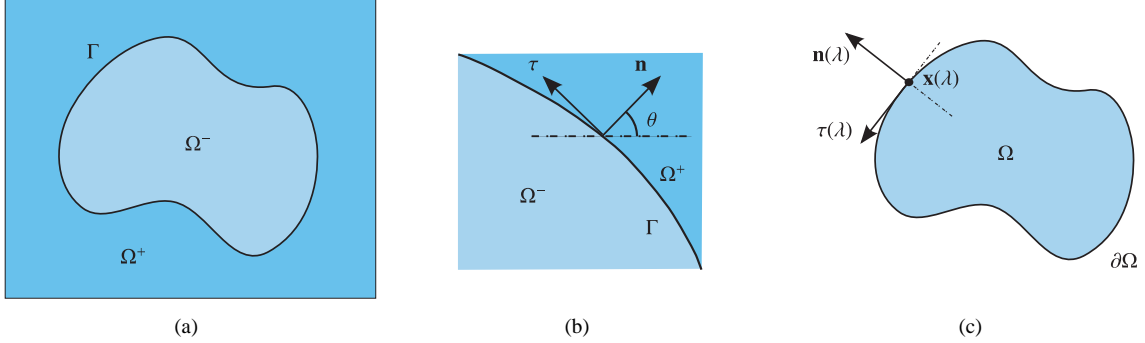


Figure 1: (a) Interface Γ separating the two fluid domains Ω^+ (exterior) and Ω^- (interior): $\Omega = \Omega^+ \cup \Omega^- \cup \Gamma$. (b) The unit vectors \mathbf{n} and $\boldsymbol{\tau}$. (c) Irregular domain Ω . The vectors $\mathbf{n}(\lambda)$ and $\boldsymbol{\tau}(\lambda)$ are the outward normal and the positively oriented tangent vectors of $\partial\Omega$ at the point $\mathbf{x}(\lambda)$.

With these concerns in mind, we introduce a second order virtual node method for approximating the two-phase Stokes equations with irregular embedded interfaces and boundaries on a uniform Cartesian MAC-grid. We use regular grids because it simplifies the implementation, permits straightforward numerical linear algebra and achieves higher order accuracy in L^∞ . Our approach uses duplicated Cartesian grid cells along the interface to introduce additional “virtual” nodes that accurately account for the lack of regularity. We formulate the Stokes problem as three Poisson equations with jump conditions to allow us to leverage our previous work in [1]. This can only be done for the case of continuous viscosity in the two phases. The staggering of variables on the MAC-grid requires separate Poisson discretizations for each variable. We also solve a final Poisson equation over the pressure grid to enforce a discrete divergence free condition yielding a total of four Poisson solves per Stokes solve. The interface is represented by Lagrangian particles for straightforward interface advection, however the techniques in [1] require a level set representation of interfaces and boundaries. We provide a method for transforming the Lagrangian interface into a level set defined over a “doubly-fine” grid that contains each of the staggered grids. This ensures that the discrete interface conditions are enforced consistently for the staggered variables. In all but the final Poisson equation, our approach yields the standard 5-point difference stencil away from embedded boundaries (notably, we have the standard 5-point stencil *across* the interface between the phases). Numerical experiments indicate second order accuracy in L^∞ for both velocities and pressure.

2. Existing Methods

Our method is second order in L^∞ for both embedded boundary conditions on irregular flow domains and for embedded interfacial discontinuities in two-phase flows. This is achieved with relatively simplistic linear algebra demands. Specifically, all linear systems are symmetric positive definite and have discrete stencils equal to that of the standard 5-point Laplacian discretization almost everywhere. Furthermore, our approach yields discretely divergence free velocities. Although many researchers have developed embedded methods for the Stokes equations with interfacial discontinuities and irregular domains, our approach is the first to support this feature set. 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 techniques use a computational domain that simply encompasses rather than geometrically adheres to the irregular domain. A good review of embedded methods is given by Lew et al in [2]. They point out that these techniques originated with the papers of Harlow and Welch [3] and Charles Peskin [4]. Peskin developed the immersed boundary method (IBM) to simulate blood flow in the heart [5, 6, 7, 8], but it has also been applied to many other problems. A summary of the development of the immersed boundary method and its applications can be found in [9].

Despite its vast popularity and considerable ease of implementation, the IBM suffers from its use of regularized delta functions to represent singular forces acting on interfaces. This renders the method first order accurate and implies that the physical characteristics of the flow near the interfacial boundaries are not accurately captured. Singular forces acting on the interface impose discontinuities in the pressure, the velocities and their derivatives which the IBM may fail to accurately resolve [10]. However, for sufficiently smooth problems in which the interfaces are thick instead of infinitesimally thin the IBM can achieve second order accuracy [11, 12]. Adaptive versions of the IBM were developed in [13] to enhance convergence over coarse grids but the results were still only first order accurate. Another deficiency of the IBM is poor conservation of volume near the interface. The seriousness of this problem, especially for the simulation of blood flow in the heart, motivated the development of a better volume conserving version of the IBM in [14].

Many methods were designed to improve the order of accuracy of the original IBM. The Immersed Interface Method (IIM) is perhaps the most popular example of this. The IIM was first developed for elliptic equations with interfacial discontinuities [15] and later applied to Stokes flow [16]. The IIM achieves second (and higher) order accuracy by capturing interfacial discontinuities in the pressure, the velocities and their derivatives in a sharp manner. The IIM has been used in many fluid flow problems including interface and rigid boundary problems [10, 17, 18, 19, 20, 21], Hele-Shaw flow [22] and also problems in which the viscosity is discontinuous across the interfaces [23]. Arbitrarily high orders of accuracy have been achieved [24, 25]. The method 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 [26]. For more about the IIM, we refer the reader to a review of its applications in [27].

A limitation of the IIM is the lack of symmetry in discretizations arising from problems with discontinuous coefficients. This imposes an obstacle on the overall speed of these methods since fast linear solvers such as conjugate gradients and Fast Fourier Transforms cannot be used. However, it should be noted that the IIM can yield symmetric matrices for continuous viscosity Stokes flow. One method that is capable of always guaranteeing a symmetric discretization is the Ghost Fluid Method (GFM). Initially applied to the Poisson equation with interfacial jumps and variable coefficients [28], the GFM was also used to simulate multiphase incompressible flow [29]. Unfortunately, the GFM is only capable of achieving first order results for interface problems.

Some of the first embedded methods were fictitious domain methods by Hyman [30] and Saul'ev [31]. The fictitious domain approach has been used with incompressible materials in a number of works [32, 33, 34, 35, 36, 37, 38, 39, 40]. These approaches embed the irregular geometry in a more simplistic 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 [41, 42]. These approaches are equivalent to the variational cut cell method of Almgren et al. in [43]. Enforcement of Dirichlet constraints is more difficult with variational cut cell approaches [44, 2] 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 [45]. 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 [46, 47, 48, 49, 50].

There are also many finite difference (FDM) and finite volume methods (FVM) that utilize cut uniform grid cells. Many of these methods have been developed in the context of incompressible flow. For example, Almgren et al. use cut uniform bilinear cells to solve the Poisson equation for pressures in incompressible flow calculations [43]. Marelle et al. use collocated grids and define sub cell interface and boundary geometry in cut cells via level sets [51]. Ng et al. also use level set descriptions of the irregular domain and achieve second order accuracy in L^∞ for incompressible flows [52]. The approach of Batty and Bridson is similar, but not as accurate [53]. Cut cell FDM and FVM have also been developed for incompressible and nearly incompressible elastic materials. Bijelonja et al. use cut cell FVM to enforce incompressibility more accurately than is typically seen with FEM [54]. Beirão da Veiga et al. use polygonal

FVM cells to avoid remeshing with irregular domains [55]. Barton et al. [56] and Hill et al. [57] use cut cells with Eulerian elastic/plastic flows.

3. Reformulation of the Stokes Equations: Poisson

Our method is designed to leverage the advances in a recent paper by Bedrossian et al [1]. We facilitate this by formulating our discretization in terms of three Poisson equations. We will first cover the derivation of these Poisson equations from the two-phase Stokes equations in irregular domains (1). Solutions \mathbf{u} and p may have discontinuities across the interface Γ [16]. We therefore first consider each subdomain Ω^+ and Ω^- separately to avoid complications associated with these discontinuities. Taking the divergence of both sides of equation (1) and noting that $\nabla \cdot \mathbf{u} = 0$ from incompressibility, we get the following Poisson equation:

$$\nabla \cdot (\mu \Delta \mathbf{u} - \nabla p + \mathbf{f}) = 0 \implies \Delta p = \nabla \cdot \mathbf{f}, \quad \mathbf{x} \in \Omega \setminus \Gamma \quad (6)$$

Again, this divergence is rigorously defined for all points $\mathbf{x} \in \Omega$ on either side of the interface ($\mathbf{x} \in \Omega \setminus \Gamma = \Omega^- \cup \Omega^+$) but not for points \mathbf{x} on the interface. After solving this Poisson equation for the pressure p , we can solve another two Poisson equations for the velocity components u and v using the solution for p :

$$\Delta p = \nabla \cdot \mathbf{f}, \quad \mathbf{x} \in \Omega \setminus \Gamma \quad (7)$$

$$\mu \Delta u = p_x - f_1, \quad \mathbf{x} \in \Omega \setminus \Gamma \quad (8)$$

$$\mu \Delta v = p_y - f_2, \quad \mathbf{x} \in \Omega \setminus \Gamma. \quad (9)$$

Here, f_1 and f_2 are the horizontal and vertical components of the body force density field \mathbf{f} . These equations hold on the interior of the domain and away from the interface. We must therefore derive boundary and interface conditions in terms of the body forces \mathbf{f} and the interfacial forces \mathbf{f}^i to solve these equations in practice.

3.1. Interface Conditions

We assume that all fluids have the same viscosity μ . In this case, the discontinuities in u , v and p are decoupled from each other [23] and the three Poisson equations above can truly be solved separately. Let $\mathbf{x}(\lambda, t)$ be an arbitrary parametrization of the curve Γ . We can express the jump conditions in u , v and p in terms of the force per unit length of the parametrization parameter λ . We will use $\mathbf{F}(\lambda, t)$ to denote this parameterization dependent force density. Recall that the jump in a quantity w at a point \mathbf{x} of the interface is expressed as:

$$[w](\mathbf{x}) := w^+(\mathbf{x}) - w^-(\mathbf{x}) = \lim_{\epsilon \rightarrow 0^+} w(\mathbf{x} + \epsilon \mathbf{n}) - \lim_{\epsilon \rightarrow 0^+} w(\mathbf{x} - \epsilon \mathbf{n}) \quad (10)$$

where \mathbf{n} is the outward unit normal of Ω^- at the point \mathbf{x} . If we let τ denote the positively oriented unit tangent to the interface Γ at the point \mathbf{x} , then

$$\begin{aligned} \mathbf{n} &= (\cos \theta, \sin \theta) \\ \tau &= (-\sin \theta, \cos \theta), \end{aligned} \quad (11)$$

where θ is the positively oriented angle between \mathbf{n} and the x axis (see Figure 1(b)). The normal and tangential components of the force density $\mathbf{F}(\lambda, t)$ are then:

$$\begin{aligned} F_n(\lambda, t) &:= \mathbf{F}(\lambda, t) \cdot \mathbf{n}(\lambda, t) \\ F_\tau(\lambda, t) &:= \mathbf{F}(\lambda, t) \cdot \tau(\lambda, t). \end{aligned} \quad (12)$$

We can use these conventions to express the jump conditions on u , v , p [10, 20, 58]:

$$[p](\mathbf{x}(\lambda, t)) = F_n(\lambda, t) \left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\|^{-1} \quad (13)$$

$$[u](\mathbf{x}(\lambda, t)) = 0 \quad (14)$$

$$[v](\mathbf{x}(\lambda, t)) = 0 \quad (15)$$

$$\left[\frac{\partial p}{\partial n} \right](\mathbf{x}(\lambda, t)) = \frac{\partial}{\partial \lambda} \left(F_\tau(\lambda, t) \left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\|^{-1} \right) \left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\|^{-1} \quad (16)$$

$$\mu \left[\frac{\partial u}{\partial n} \right](\mathbf{x}(\lambda, t)) = F_\tau(\lambda, t) \sin \theta \left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\|^{-1} \quad (17)$$

$$\mu \left[\frac{\partial v}{\partial n} \right](\mathbf{x}(\lambda, t)) = -F_\tau(\lambda, t) \cos \theta \left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\|^{-1}, \quad (18)$$

Despite the explicit appearance of the parametrization function $\mathbf{x}(\lambda, t)$, the jumps defined above are independent of the chosen parametrization in the sense that, if $\tilde{\mathbf{x}}(\tilde{\lambda}, t)$ also parametrizes Γ , then all the jumps above are the same at each point $\tilde{\mathbf{x}}(\tilde{\lambda}, t) = \mathbf{x}(\lambda, t)$.

3.2. Irregular Domain Boundary Conditions

We must also know appropriate boundary conditions for u , v and p to solve Poisson equations (7), (8) and (9) in practice. We will assume that Dirichlet velocity conditions are known on $\partial\Omega$. Thus for all $\mathbf{x} \in \partial\Omega$ we will have $u(\mathbf{x}) = U(\mathbf{x})$ and $v(\mathbf{x}) = V(\mathbf{x})$ for some functions $U(\mathbf{x})$ and $V(\mathbf{x})$ defined only over $\partial\Omega$. Let $\mathbf{x}(\lambda)$ be a parameterization of $\partial\Omega$, let $\mathbf{n}(\lambda)$ be the unit outward normal vector to $\partial\Omega$ at the point $\mathbf{x}(\lambda)$ and let $\boldsymbol{\tau}(\lambda)$ be the unit tangential vector to $\partial\Omega$ at that same point (see Figure 1(c)). The Neumann boundary conditions for the pressure are then:

$$\nabla p \cdot \mathbf{n} = \mathbf{f} \cdot \mathbf{n} - \frac{\partial \omega}{\partial \tau} \quad \text{for } \mathbf{x} \in \partial\Omega, \quad (19)$$

where $\omega = v_y - u_x$ is the fluid vorticity and $\partial\omega/\partial\tau = \nabla\omega \cdot \boldsymbol{\tau}$ is the tangential component of the gradient of ω at $\mathbf{x} \in \partial\Omega$.

4. Description of Numerical Method

The method couples a discrete Lagrangian representation of the interface (Γ_h) with a background Eulerian representation of fluid velocities and pressures. The Lagrangian interface moves with the local fluid velocity and the temporal discretization is explicit. The fluid variables are stored on a staggered MAC grid [59] to facilitate enforcement of a discrete divergence-free condition. We use the Virtual Node Algorithm (VNA) [1] to solve the Poisson interface problems (7), (8) and (9) over the respective sub-grids in the MAC grid. The interface is represented with Lagrangian particles and interfacial forces are naturally defined at the particle locations. However, the VNA was designed for a level set representation of the interface geometry and jump conditions. We provide a procedure for defining a level set representation of the interface geometry and boundary conditions from the Lagrangian particles. The overall procedure for advancing one time step is:

1. Compute the level set and transfer jump conditions from the Lagrangian interface Γ_h ; details in Section 4.1
2. Use VNA to solve $\Delta p = \nabla \cdot \mathbf{f}$ with interface conditions (13,16) and Neumann boundary conditions (19); details in Section 4.2
3. Use VNA to solve $\mu \Delta u = -p_x + f_1$ with interface conditions (14,17); details in Sections 4.2 and 4.3
4. Use VNA to solve $\mu \Delta v = -p_y + f_2$ with interface conditions (15,18); details in Sections 4.2 and 4.3
5. Project the velocity field $\mathbf{u} = (u, v)$ to enforce the divergence-free condition; details in Section 4.4
6. Interpolate velocities from the MAC grid to the Lagrangian interface and forward Euler update particle positions.

In the last step, the x and y components of the velocity on a given interface node \mathbf{x}_i are interpolated from the u and v velocities respectively determined in steps 4 and 5. This interpolation is done using the conventions presented in [1] for defining quantities on the interface.

4.1. Computation of the Level Set from Γ_h

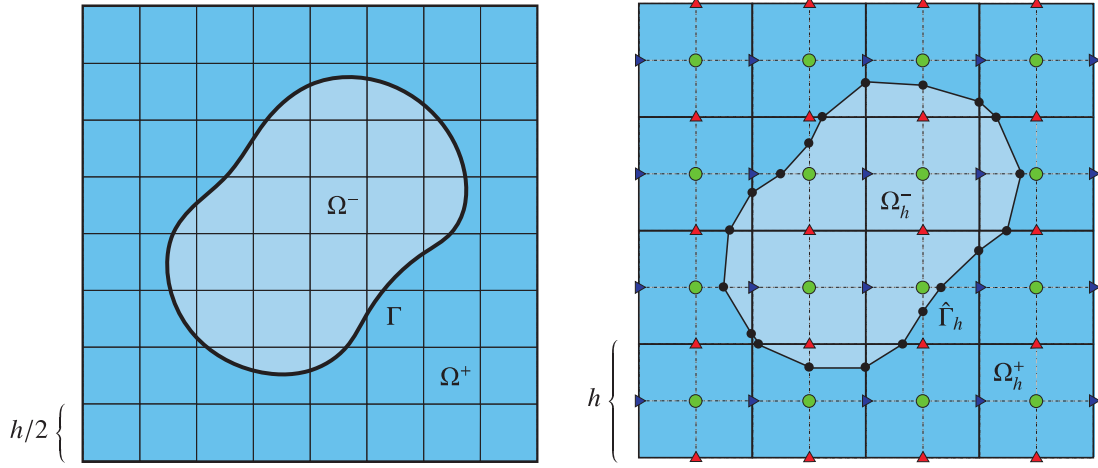


Figure 2: The level set is defined on a doubly-fine grid with $2N$ cells per direction (on the left). The values of u are stored on the blue nodes (triangles pointing to the right), the values of v are stored on the red nodes (triangles pointing upwards) and the values of p are stored on the green nodes (circles). The duplication of nodes and cells required by the method of [1] is done for the u , v and p grids separately. The doubly-fine level set grid ensures that these duplications are done in a consistent manner.

A Lagrangian representation of the interface is convenient for computing interfacial forces and also for discretizing the motion of the interface. We represent the discrete interface Γ_h with a sequence of points \mathbf{x}_i where $i \in \{0, 1, 2, \dots, M-1\}$. The points are connected by segments and form a closed curve (we also support multiple closed curves) dividing the domain into the discrete interior and exterior domains Ω_h^- and Ω_h^+ respectively (see Figures 2 and 3). In order to use the VNA, we need to define a level set over the MAC grid that corresponds with Γ_h . Let N denote the number of MAC grid cells per dimension and let h be the length of a cell edge (we assume x and y edges have the same length, see Figure 2). The level set is defined on the nodes of a doubly-fine grid with $2N \times 2N$ cells per direction (see Figure 2). We use the doubly-fine grid because it naturally defines a discrete level set on the u , v and p subgrids. An alternative is to compute a level set with N cells per dimension on each of the respective subgrids, however such a choice would complicate the accurate computation of p_x and p_y needed for the u and v Poisson solves (see Section 4.3 for a more detailed discussion of these issues). The level set is computed as a signed distance function on the nodes of cells intersecting Γ_h . Nodes that are not incident on a cell intersecting Γ_h are set to a positive or negative constant depending on whether the node is inside Ω_h^+ or Ω_h^- respectively. The computation of the narrow band level set is negligible compared to the time taken to solve the linear systems with the VNA.

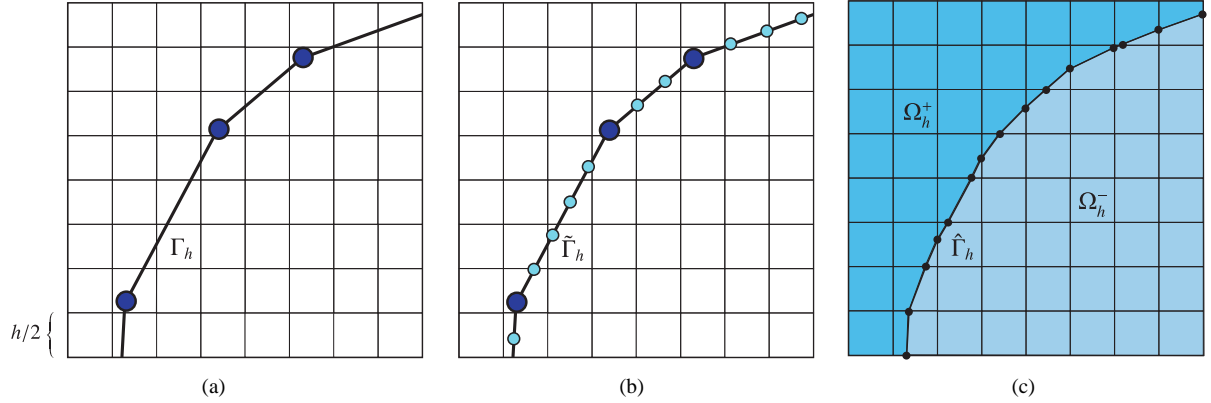


Figure 3: Figure (a) shows the original discrete Lagrangian interface Γ_h with nodes \mathbf{x}_i (blue circles). Figure (b) shows the subdivided $\tilde{\Gamma}_h$. The added nodes ($\tilde{\mathbf{x}}_i$) are indicated with smaller light blue circles. Nodes are added until all segments of $\tilde{\Gamma}_h$ are shorter than the cell width $h/2$ of the doubly-fine grid. Figure (c) shows the Lagrangian approximation $\hat{\Gamma}_h$ of the discrete interface Γ_h which is generated by the level set computed from Γ_h . The black circles denote the positions at which $\hat{\Gamma}_h$ crosses the edges of each cut cell. The domains Ω_h^+ and Ω_h^- are defined from the level set and $\hat{\Gamma}_h$. The Lagrangian curve $\hat{\Gamma}_h$ is used in the VNA [1] for quadrature purposes and interface conditions must be transferred from Γ_h (where they are naturally defined) to $\hat{\Gamma}_h$.

In order to simplify the process of determining which interface segments in Γ_h intersect which grid cell edges, we first ensure that each segment on the interface is smaller than the cell width $h/2$ of the doubly-fine grid. This can be done by adding nodes $\tilde{\mathbf{x}}_i$ to subdivide its segments (see Figure 3). This is only for the computation of nodal level set values; the computation of the interfacial forces uses the original set of segments Γ_h . We will denote this subdivided discrete interface as $\tilde{\Gamma}_h$. This subdivision simplifies the determination of which doubly-fine grid cells intersect which segments in Γ_h . We also perturb the nodes $\tilde{\mathbf{x}}_i$ of $\tilde{\Gamma}_h$ to prevent them from falling directly on the edges of any cell (see figure 4). Specifically, for a given $\tilde{\mathbf{x}}_i$, we determine which cell in the doubly-fine grid contains it. We then clamp this node toward the cell center in a dimension by dimension fashion until it is at least as far as some tolerance αh away from the edges of the cell (typically we use $\alpha = 1e^{-6}$). Once the nodes are perturbed, we can unambiguously determine which segments in $\tilde{\Gamma}_h$ intersect which edges in the doubly-fine grid. Cell edges cut an even number of times are considered to be uncut. All possible cases are illustrated in Figure 5. In order to avoid ambiguities, cases (4) and (7) require special treatment. We treat case (4) as a subcase of case (3) and also case (7) as a subcase of case (6). For example, consider Figure 6. In case (4) the node A is within a small tolerance of the segment PQ , P is in the lower-left cell and Q is in the upper-right cell. Here, we automatically consider edges AC and AB as intersecting edge PQ . Otherwise if node A is sufficiently far from edge PQ we explicitly compute intersections between edge PQ and edges AC , AB , AE and AD to determine if we are in case (5) or case (3).

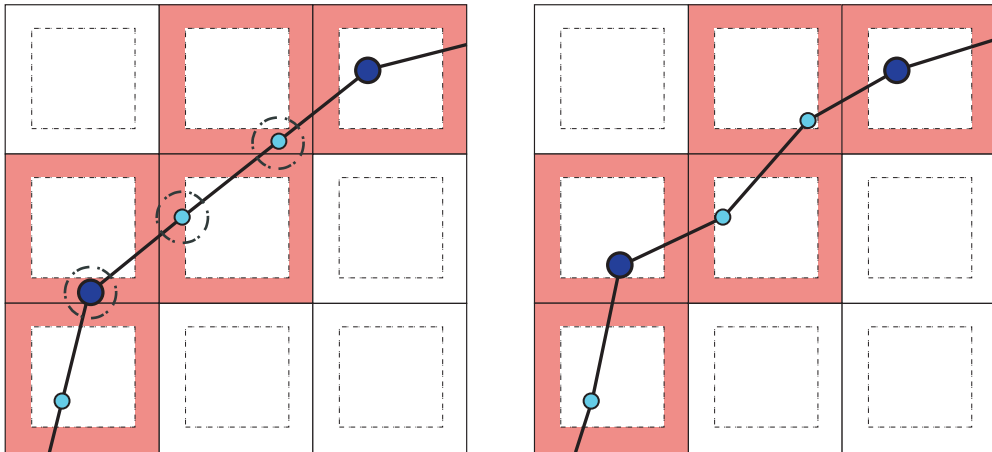


Figure 4: The picture on the left shows a piece of the interface $\tilde{\Gamma}_h$ with three (circled) nodes lying too close to the edges of the cells they fall into. The picture on the right shows the perturbation applied to these nodes. The shaded regions depict the thresholds used for determining if a node is too close to a grid cell edge. Note that the size of these regions is exaggerated for ease of visualization. In practice they are set to have a width proportionate to $h/2$.

Once we have determined which subdivided interface segments in $\tilde{\Gamma}_h$ intersect which grid cell edges, we compute the exact distance to $\tilde{\Gamma}_h$ for all grid nodes incident on a cut cell. That is, for all grid nodes with at least one of its four incident cells cut by a segment in $\tilde{\Gamma}_h$, we compute the analytic distance from that node to each of the segments that intersect any of its incident cells and define the minimum as the distance from the grid node to $\tilde{\Gamma}_h$. We set the distance to be a large positive constant for all grid nodes not incident on a cut cell. Finally, we determine the sign of the level set values depending on whether or not the nodes they correspond to are inside or outside $\tilde{\Gamma}_h$ (see figure 7). This is done with a flood-fill approach. The lower left node of the domain is defined to be outside the domain, we then sweep through the nodes in a dimension-by-dimension fashion. The sign of the next node in the sweep is the same as the previous node if the edge connecting them is crossed an even number of times, otherwise it is given the opposite sign.

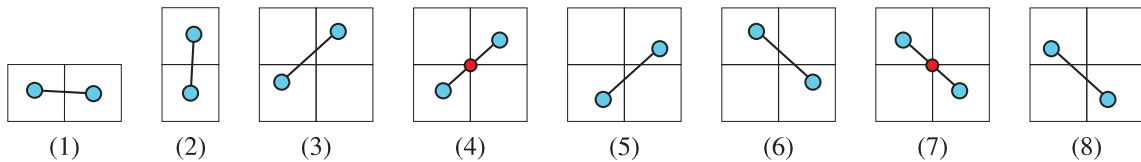


Figure 5: The eight cases to be considered when determining which cell edges are cut by a given segment of the discrete interface $\tilde{\Gamma}_h$. Cases (4) and (7) happen when a segment intersects four cells simultaneously by crossing a corner node (shown as small red circles in the picture).

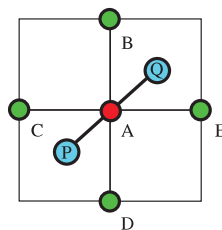


Figure 6: A case which requires special attention on the segment-cell collision detection.

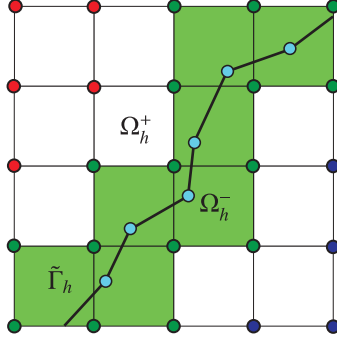


Figure 7: The level set values are exact at the green nodes (which belong to cells which are crossed by the interface $\tilde{\Gamma}_h$; these cells are shaded in green on the picture). The level set values are set to a large positive constant on the red nodes (which belong to Ω_h^+) and to a large negative constant on the blue nodes (which belong to Ω_h^-)

4.2. Computation of Jump Conditions at the Level Set Zero Isocontour

The VNA requires a description of the jump conditions along the embedded interface. It suffices to define these conditions at the center of each segment on a cell-wise Lagrangian approximation to the zero isocontour of the level set. We call this additional Lagrangian curve $\hat{\Gamma}_h$. $\hat{\Gamma}_h$ is an approximation to the zero isocontour of the level set (see Figure 3). It is determined by connecting points on the doubly-fine grid cell edges where the level set interpolates to zero. The VNA uses $\hat{\Gamma}_h$ for quadrature purposes and the accuracy required in the definition of the jump conditions is related to this. Unfortunately, the jump and boundary conditions needed for equations (7), (8) and (9) are naturally defined at the nodes of the original Lagrangian interface Γ_h . We provide a procedure to transfer these conditions from the Lagrangian nodes of Γ_h to the centers of the segments on $\hat{\Gamma}_h$ (see Figure 8).

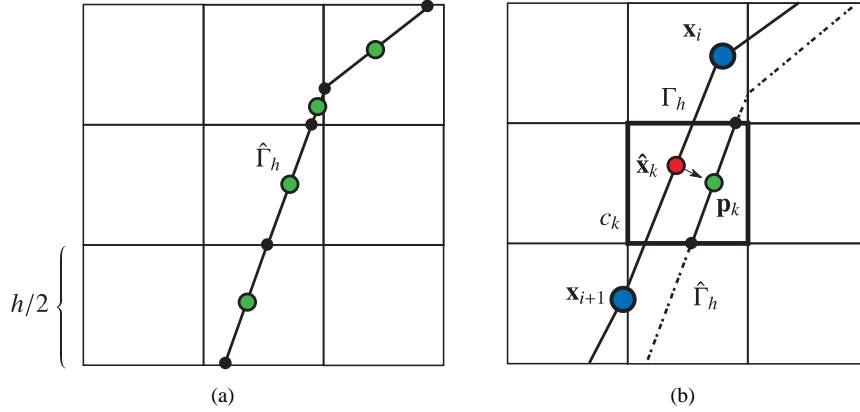


Figure 8: Setting the jump conditions. (a): In the VNA, the jump conditions (Equations (13-18)) must be defined at the centers of the segments on the cell-wise Lagrangian approximation to the zero isocontour of the level set. These cell-wise segment centers are shown in light-green large circles above. The black smaller circles represent points where the level set interpolates to zero for cut cells (these points define the cell-wise Lagrangian approximation to the zero-isocontour). Figure (b): Jump conditions are naturally computed at the nodes \mathbf{x}_i of the discrete interface Γ_h and then transferred to the points \mathbf{p}_k through the points $\hat{\mathbf{x}}_k$. Each point \mathbf{p}_k lies at the center of the $\hat{\Gamma}_h$ segment on the doubly-fine cut cell c_k . The jump at $\hat{\mathbf{x}}_k$ is computed by linearly interpolating the values at the nodes \mathbf{x}_i and \mathbf{x}_{i+1} . Note that the above separation between the level set generated interface $\hat{\Gamma}_h$ and the discrete interface Γ_h is exaggerated for illustrative purposes.

Recall that we denote the points on the ends of the segments in the curve Γ_h as \mathbf{x}_i with $i = 0, 1, \dots, M - 1$. If there are m doubly-fine grid cells cut by the zero isocontour of the levelset, let $\mathcal{P} = \{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{m-1}\}$ denote the set of segment centers on $\hat{\Gamma}_h$ (see Figure 8). Assume we have computed the jump conditions at the points \mathbf{x}_i of Γ_h . For each $k = 0, 1, \dots, m - 1$, let $\hat{\mathbf{x}}_k$ be the point on Γ_h closest to \mathbf{p}_k (see Figure 8). We linearly interpolate the jump condition at $\hat{\mathbf{x}}_k$ from the values at the ends of the segment to which it belongs. This condition is then defined to be the jump at \mathbf{p}_k .

In the VNA [1], we impose the jump conditions over each cut cell c_l^u on the u grid in an integral average sense:

$$\int_{c_l^u \cap \hat{\Gamma}_h} [u(\mathbf{x})] dS = \int_{c_l^u \cap \hat{\Gamma}_h} a^u(\mathbf{x}) dS \quad (20)$$

$$\int_{c_l^u \cap \hat{\Gamma}_h} [\nabla u(\mathbf{x}) \cdot \mathbf{n}] dS = \int_{c_l^u \cap \hat{\Gamma}_h} b^u(\mathbf{x}) dS \quad (21)$$

where $c_l^u \cap \hat{\Gamma}_h$ is the portion of $\hat{\Gamma}_h$ on the l -th cut cell (c_l^u) of the u grid, $a^u(\mathbf{x})$ is the jump in $u(\mathbf{x})$ and $b^u(\mathbf{x})$ is the jump in the normal derivative of $u(\mathbf{x})$. Similar expressions for $[v(\mathbf{x})]$ and $[p(\mathbf{x})]$ enforce the jumps on v and p respectively on the cut cells of their grids. Let L_k denote the length of the segment of $\hat{\Gamma}_h$ on the doubly-fine cut cell c_k . We can approximate the integrals of the jump conditions using the values at the segment centers ($a(\mathbf{p}_k)$ and $b(\mathbf{p}_k)$) and the segment lengths (L_k):

$$\int_{c_l^u \cap \hat{\Gamma}_h} a(\mathbf{x}) dS \approx \sum_{c_k \cap c_l^u} a(\mathbf{p}_k) L_k \quad (22)$$

$$\int_{c_l^u \cap \hat{\Gamma}_h} b(\mathbf{x}) dS \approx \sum_{c_k \cap c_l^u} b(\mathbf{p}_k) L_k, \quad (23)$$

where the sum takes into account only the doubly-fine cells c_k which intersect the cell c_l^u .

4.3. Construction of ∇p

The VNA uses duplicated grid cells along the interface to introduce additional nodal degrees of freedom. These degrees of freedom allow for the more accurate solution of the Poisson interface problems. The MAC grid structure we employ therefore requires this duplication procedure on each of the u , v and p subgrids. See Figure 10 for an illustration of this process. The duplication must be done in a manner that coordinates duplication of the three subgrids. Specifically, the u , v and p degrees of freedom interact through the right hand side terms in Poisson equations for u and v (where p_x and p_y respectively appear). Therefore the duplication process must admit a procedure for the computation of the respective components of ∇p on each of the duplicated grids for u and v . This is complicated by the use of virtual p degrees of freedom in the ∇p stencils on cut cells. Fortunately, we will show here that the definition of the level set over the doubly-fine grid admits a consistent duplication and subsequent ∇p transfer procedure.

The duplication procedure we advocate is trivially defined by inheriting the doubly-fine level set to each of the u , v and p subgrids. We first define the Lagrangian zero isocontour on the doubly-fine grid ($\hat{\Gamma}_h$) as described in section 4.2. This gives a piecewise linear approximation to the zero isocontour over each cut cell in the doubly-fine grid. Then, for each subgrid we define the Lagrangian cell-wise approximation to the zero isocontour as the union of the piecewise linear approximations over the four grid cells in the doubly-fine grid that are contained in the subgrid (see Figure 9 for illustration). Also, a u , v or p cell is considered cut (and then subsequently duplicated) if any of its four sub doubly-fine grid cells are cut. Note that this is a slightly different criteria than that used in the original VNA. There, a grid cell was considered cut if any of its four nodal level set values were of differing sign.

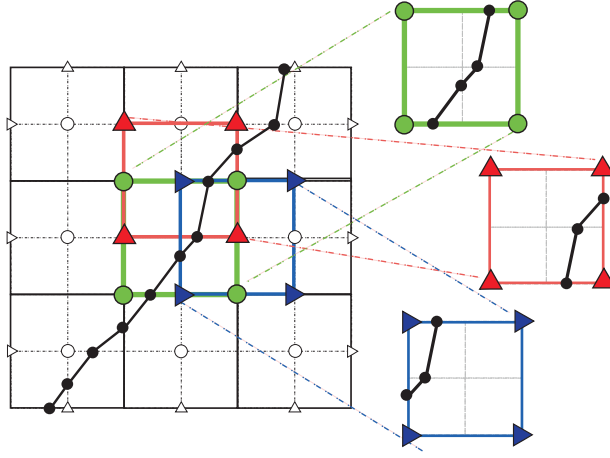


Figure 9: Inheriting the Lagrangian zero isocontour from the doubly-fine grid. The image depicts the definition of the piecewise linear approximation to the zero isocontour over u , v and p grid cells. For each case, the grid cell is considered cut if any of the four sub grid cells are cut. u grid cells are shown in blue with triangular nodes pointing to the right, the v grid cells are show in red with triangular nodes pointing upwards and the p grid cells are shown in green with circular nodes.

The staggering of the variables on the MAC grid naturally facilitates a second order centered finite difference stencil for p_x at each u degree of freedom and for p_y at each v degree of freedom. Not coincidentally, these are precisely the source terms needed in the Poisson equations for u and v respectively. Unfortunately, these stencils are not always well defined for interfacial u and v degrees of freedom. For example consider the left subfigure in Figure 10. Here, a centered difference computation of p_x at the ghost u node (drawn as a large blue triangle pointing to the right) is not possible because there is no p degree of to the left.

Fortunately, the VNA requires only cell-wise averages of the right had side terms of the Poisson equations (see section 4 in [1] for more details). Notably we will never explicitly require components of ∇p at ghost u and v nodes like the one node highlighted in the left subfigure in Figure 10. The required cell-wise averages are:

$$\overline{p_x} = \int_{c_x \cap \Omega_h^\pm} p_x d\mathbf{x} \quad \text{and} \quad \overline{p_y} = \int_{c_y \cap \Omega_h^\pm} p_y d\mathbf{x}.$$

Here, c_x and c_y are cells in the duplicated u and v grids respectively. $c_x \cap \Omega_h^\pm$ is the intersection of the fluid domain with c_x (similar for $c_y \cap \Omega_h^\pm$). Note that these regions are non-trivial for cells in the duplicated grids that intersect the interface. That is, these regions correspond to cut cells. An example of such a cut cell is shown incident to the enlarged u node in left subfigure of Figure 10. While we do not have a p degree of freedom to the left of this node, we do have enough information to compute $\overline{p_x}$ over the cut cell. Specifically, we have enough information to determine a piecewise bilinear approximation to p over $c_x \cap \Omega_h^-$. Note that there is a p cell (c_p) containing region $c_x \cap \Omega_h^-$ in Figure 10. This implies that we can approximate p as bilinear over the required region. In general, there will be at most two p cells overlapping any $c_x \cap \Omega_h^\pm$ or $c_y \cap \Omega_h^\pm$ region and we can always use this to generate a piecewise bilinear approximation to p wherever needed.

The piecewise bilinear representation of p is:

$$p(\mathbf{x}) = \sum_{i=1}^{N_p} p_i N_i^p(\mathbf{x}). \quad (24)$$

Here we assume there are a total of N_p nodes on the duplicated p grid. The function $N_i^p(\mathbf{x})$ is the piecewise bilinear

interpolating function centered at the i -th pressure node. The derivatives of p with respect to x and y are:

$$p_x = \sum_{i=1}^{N_p} p_i N_{i,x}^p \quad \text{and} \quad p_y = \sum_{i=1}^{N_p} p_i N_{i,y}^p \quad (25)$$

The cell-wise averages ($\overline{p_x}$ and $\overline{p_y}$) of p_x and p_y can then be computed exactly as in [1]. This is easily done because the integrands in

$$\overline{p_x} = \int_{c_x \cap \Omega_h^\pm} \sum_{i=1}^{N_p} p_i N_{i,x}^p d\mathbf{x} \quad \text{and} \quad \overline{p_y} = \int_{c_y \cap \Omega_h^\pm} \sum_{i=1}^{N_p} p_i N_{i,y}^p d\mathbf{x}$$

are piecewise bilinear. That is, we need only integrate a low order polynomial in x and y over the polygonal cut cell regions $c_x \cap \Omega_h^\pm$ and $c_y \cap \Omega_h^\pm$. As shown in [1], integrals of this type can be computed exactly with minimal cost.

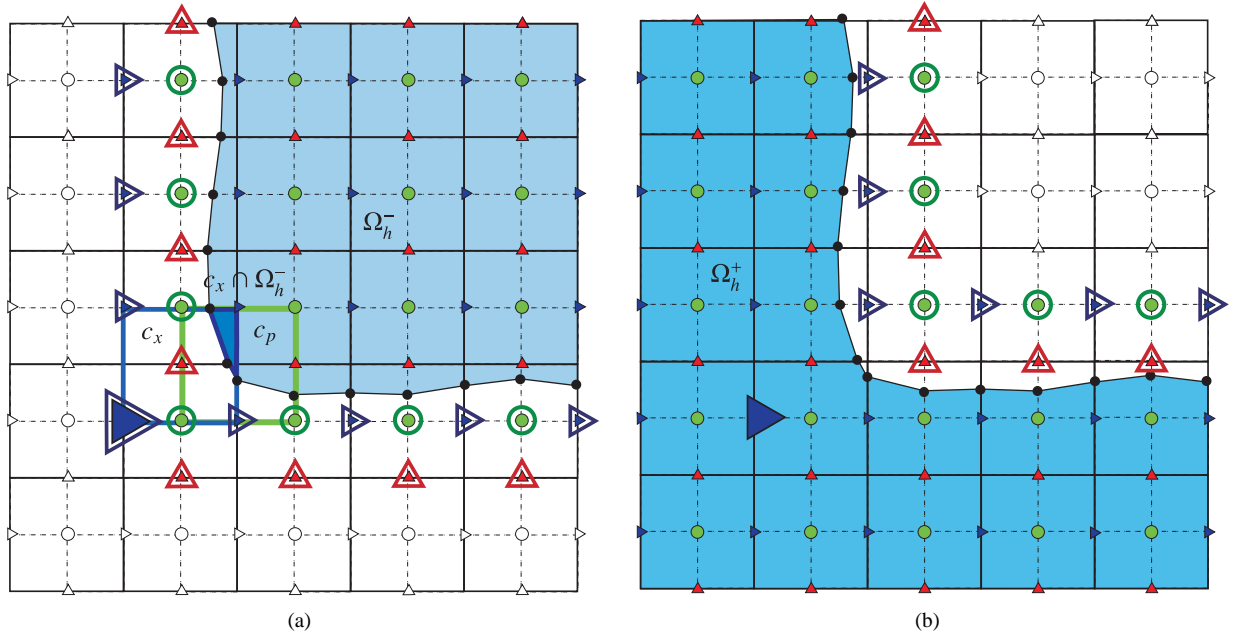


Figure 10: A typical case where the computation of ∇p is not trivial. The colored nodes in figure (a) are either virtual (shown with an extra border outline) or material nodes of the u , v and p grids associated with Ω_h^- , while the colored nodes on figure (b) are the equivalent ones for Ω_h^+ . The large blue triangle corresponds to a virtual node on the Ω_h^- u grid. The original node is shown enlarged on the right in Ω_h^+ . Notice that simple methods such as centered differences cannot be used to compute p_x numerically on the node on the Ω_h^- side since there is no p node of Ω_h^- on its left. Note that we highlight the cells c_x and c_p as well as the region $c_x \cap \Omega_h^-$ in the image at the left to aid in the discussion from section 4.3.

4.4. Projection Onto Divergence-free Space

When we use the Poisson formulation (equations (8) and (9)) for the discretization of u and v , we get second order (in L^∞) accurate velocities. However, they do not in general satisfy a discrete divergence free condition. While any consistent approximation of the divergence will converge to zero under refinement with the u and v we generate from the Poisson equations, it is often advantageous to enforce a discrete divergence free condition. We satisfy this divergence free condition via projection and our numerical experiments suggest that this process does not degrade the L^∞ accuracy.

The divergence free condition is defined over each cell in a p -node centered grid. For every p node (including virtual p nodes created in the duplication procedure), we define the p -centered grid cell (C_p) to consist of the four

surrounding sub cells in the doubly-fine grid (see Figure 11). These cells are also duplicated whenever the p node at its center is virtual. We enforce the following conditions at all p -centered grid cells C_p :

$$\int_{C_p \cap \Omega_h^\pm} \nabla \cdot \mathbf{u} \, d\mathbf{x} = \int_{C_p \cap \Omega_h^\pm} (u_x + v_y) \, d\mathbf{x} = \int_{C_p \cap \Omega_h^\pm} \left[\sum_{i=1}^{N_u} u_i N_{i,x}^u + \sum_{i=1}^{N_v} v_i N_{i,y}^v \right] d\mathbf{x} = 0. \quad (26)$$

Here N_u is the number of nodes (including virtual nodes) on the u grid and N_v is the number of nodes on the v grid. Note that this defines a linear set of constraints on the u and v degrees of freedom. If we define \mathbf{w} to be the vector of all nodal u and v unknowns:

$$\mathbf{w} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \quad (27)$$

then we can symbolically represent the linear constraints in equations (26) as $\mathbf{D}\mathbf{w} = \mathbf{0}$. The limited support of the bilinear functions N_i implies that \mathbf{D} is a sparse matrix (see Figure 11). Since we denoted the number of p nodes (including virtual nodes) as N_p , then $\mathbf{D} \in \mathbb{R}^{N_p \times (N_u + N_v)}$. Note that the evaluation of the entries in \mathbf{D} must be done using the cut-cell integration of low order polynomials over polygons as discussed in section 4.3. The projection of the \mathbf{w} obtained by solving the discrete equations (8) and (9) can then be performed as:

1. Solve $\mathbf{D}\mathbf{D}^T \hat{\mathbf{p}} = \mathbf{D}\mathbf{w}$, where $\hat{\mathbf{p}} \in \mathbb{R}^{N_p}$
2. $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{D}^T \hat{\mathbf{p}}$.

The linear system $\mathbf{D}\mathbf{D}^T \hat{\mathbf{p}} = \mathbf{D}\mathbf{w}$ is symmetric positive definite and can be solved with the conjugate gradient method.

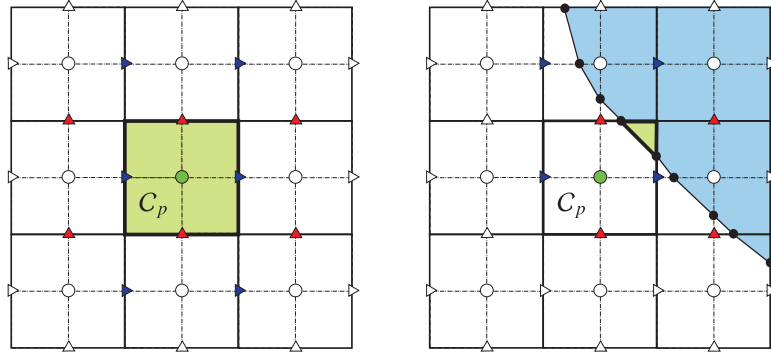


Figure 11: The stencil of the matrix \mathbf{D} for the divergence-free projection algorithm proposed involves only the colored nodes on the figures above (blue triangles pointing to the right are nodes of the u grid; red triangles pointing up are nodes of the v grid). Far away from the interface (left figure), the stencil encompasses a maximum of 12 points (6 from the u grid and 6 from the v grid). For cut cells, fewer u and v values are necessary to represent the stencil (figure on the right).

5. Numerical Examples

We will now discuss a number of numerical tests we used to demonstrate that our method is capable of achieving second order accuracy in L^∞ for u , v and p . First we will define the precise notion of order of convergence that we use and discuss how we compute it for u , v and p , then we will present our discretization and convergence results for interfacial elastic and surface tension forces as well as problems in irregular domains.

5.1. Convergence Measure

Denote the number of MAC grid cells per direction as N . Denote the numerical approximation to a field $u_{\text{exact}}(x, y)$ as $u_N(x, y)$. Let e_N be the exact error of u_N in the L^∞ norm. Then, if the method is r -th order accurate:

$$e_N = \|u_N - u_{\text{exact}}\|_\infty \leq Ch^r \quad (28)$$

for some constant C with $h \propto 1/N$ being the cell width. Hence:

$$\begin{aligned}
\|u_{2N} - u_N\|_\infty &\leq \|u_{2N} - u_{\text{exact}}\|_\infty + \|u_N - u_{\text{exact}}\|_\infty \\
&\leq C(h_{2N}^r + h_N^r) \\
&= C' \left(\frac{1}{(2N)^r} + \frac{1}{N^r} \right) \\
&= C' \frac{1}{N^r} \left(\frac{1}{2^r} + 1 \right) \\
&= C'' \frac{1}{N^r}
\end{aligned} \tag{29}$$

where all constants were incorporated into C'' . Taking the \log_{10} on both sides of the equation above and defining $a = \log_{10} C''$, we get:

$$\log_{10} \|u_{2N} - u_N\|_\infty \leq a - r \log_{10} N. \tag{30}$$

In other words, the negative of the slope of the plot of $\log_{10} \|u_{2N} - u_N\|_\infty$ versus $\log_{10} N$ is the order of convergence of the method on the L^∞ norm. We used this procedure to compute the order of convergences of the results below.

5.2. Details About the Convergence Measures of u , v , p and Γ

Since the velocities and the pressure are not always continuously differentiable across the interface $\hat{\Gamma}_h$, special care needs to be taken when computing interpolation errors for these quantities defined on grids with different resolutions. In what follows, we compute the interpolation errors for the quantity $q = u/v/p$ at the MAC grid with N cells per direction by sweeping through the material nodes of the q_N grid (virtual nodes are not considered) and comparing them with the values at the nodes of the q_{2N} grid at the same Cartesian positions. However, the material copies of these nodes on the two considered grids might not be associated with the discrete regions Ω_h^+ and $\Omega_{h/2}^+$ or with the regions Ω_h^- and $\Omega_{h/2}^-$ respectively. Such nodes are then not taken into account on the convergence measure since they represent values of the quantity q on different sides of the interface on each grid. The maximum difference between the compared values of q_N and q_{2N} is taken to be the interpolation error on the L^∞ norm for the quantity q on the grid with N cells per direction.

The interpolation error for the discrete interfaces Γ_h and $\Gamma_{h/2}$ on the L^∞ norm is taken to be the maximum distance between the segments of Γ_h and the segments of $\Gamma_{h/2}$.

5.3. Elastic Interface Discretization

Our discretization of interfacial forces is very standard but we briefly cover it here for completeness. Let $\mathbf{x}(\lambda, t)$ be a parametrization of the interface Γ . We assume that the range of λ is $[0, L_0]$, where L_0 is the equilibrium length of the elastic interface. This equilibrium length L_0 is defined such that all elastic forces vanish at all points of Γ if its configuration is a circle of radius $R_0 = 2\pi/L_0$. The elastic force density (per unit length of the parameter λ) at a given point $\mathbf{x}(\lambda, t) \in \Gamma$ is given by the equation:

$$\mathbf{F}(\lambda, t) = \frac{\partial}{\partial \lambda} (T(\lambda, t)\tau(\lambda, t)) \tag{31}$$

where $\tau(\lambda, t)$ is the unit vector tangential to the interface at the point $\mathbf{x}(\lambda, t)$:

$$\tau(\lambda, t) = \frac{\partial \mathbf{x}}{\partial \lambda} \left/ \left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\| \right. \tag{32}$$

and $T(\lambda, t)$ is the tension at that point:

$$T(\lambda, t) = \kappa(\lambda) \left(\left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\| - 1 \right). \tag{33}$$

The function $\kappa(\lambda)$ determines the elastic properties at each point of the interface. Here we will take $\kappa(\lambda)$ to be a constant. The interface is discretely represented with M nodes and we compute the force density \mathbf{F}_i on each node \mathbf{x}_i of Γ_h as:

$$\tau_{i+1/2} = \left(\frac{\partial \mathbf{x}}{\partial \lambda} \right)_{i+1/2} \bigg/ \left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\|_{i+1/2} = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|}, \quad (34)$$

$$T_{i+1/2} = \kappa \frac{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|}{\Delta \lambda} \quad (35)$$

and

$$\mathbf{F}_i = \frac{T_{i+1/2} \tau_{i+1/2} - T_{i-1/2} \tau_{i-1/2}}{\Delta \lambda} \quad (36)$$

with $\Delta \lambda = L_0/M$. Here the subscript $i + 1/2$ refers to the point at the center of the segment which connects the nodes \mathbf{x}_i and \mathbf{x}_{i+1} . We divide the force density (36) by $\left\| \frac{\partial \mathbf{x}}{\partial \lambda} \right\|_i = \frac{\|\mathbf{x}_i - \mathbf{x}_{i-1}\| + \|\mathbf{x}_{i+1} - \mathbf{x}_i\|}{2\Delta \lambda}$ to define the jump conditions:

$$\mathbf{f}_i^i = \frac{2\Delta \lambda}{\|\mathbf{x}_i - \mathbf{x}_{i-1}\| + \|\mathbf{x}_{i+1} - \mathbf{x}_i\|} \mathbf{F}_i. \quad (37)$$

For all elastic interface tests we use a rectangular periodic domain $[-1, 1] \times [-1, 1]$ with an initial elliptical interface of semi-major radius $a = 0.7$ and semi-minor radius $b = 0.4$ centered at the domain origin at $t = 0$. The ellipse is given uniform elasticity constant $\kappa = 10$. The viscosity of the fluid is set to $\mu = 1$ (uniformly). For a MAC grid with N cells per direction, the interface is represented with $M = N/2$ segments. The nodes of the interface are positioned initially according to the expression:

$$\mathbf{x}_i = (a \cos \theta_i, b \sin \theta_i), \quad (38)$$

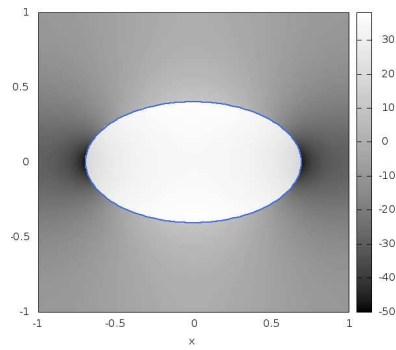
where $\theta_i = 2\pi i/N_b = 4\pi i/N$ for $i = 1, 2, \dots, N/2$. This choice of the parametrization of the interface ensures each of its segments has length between $2.5h$ and $4.5h$ at all times of the simulation. The linear systems for the pressure and for the velocities are solved using multigrid. The time step was $\Delta t = 5h^2 = 20/N^2$. We verified the order of convergence for the pressure, for the velocities and for the positions of interface nodes Γ_h according to the L^∞ norm at the times $t_1 = 0.1$, $t_1 = 0.2$, $t_3 = 0.3$ and $t_4 = 0.4$. Also, we checked how the total interface volume change $\Delta V_h = (\text{interface volume at time } t) - (\text{interface volume at time } t = 0)$ converged to zero as the grid was refined and the order of convergence of the maximum and minimum distances between the interface Γ_h and the origin of the rectangular domain (these distances are denoted below as R_{\max} and R_{\min} respectively). The numbers of grid cells per direction used on the convergence test were $N = 64, 128, 256$. As shown on the tables below, all the results obtained are second order accurate for all these quantities:

$t_1 = 0.1$	
Quantity	Order of convergence
u	2.44
v	2.31
p	2.74
Γ_h	2.63
R_{\max}	2.10
R_{\min}	1.76
ΔV_h	2.05

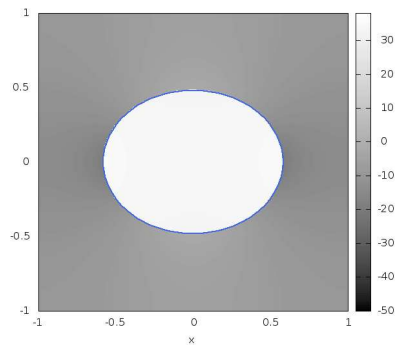
$t_2 = 0.2$	
Quantity	Order of convergence
u	2.30
v	2.20
p	2.01
Γ_h	2.24
R_{\max}	2.05
R_{\min}	1.95
ΔV_h	2.01

$t_3 = 0.3$	
Quantity	Order of convergence
u	2.22
v	2.26
p	2.23
Γ_h	2.11
R_{\max}	2.02
R_{\min}	1.98
ΔV_h	2.01

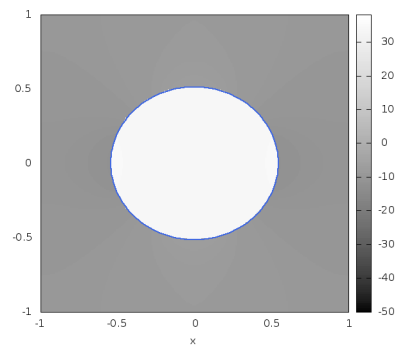
$t_4 = 0.4$	
Quantity	Order of convergence
u	2.24
v	2.15
p	2.23
Γ_h	2.10
R_{\max}	2.02
R_{\min}	1.99
ΔV_h	2.01



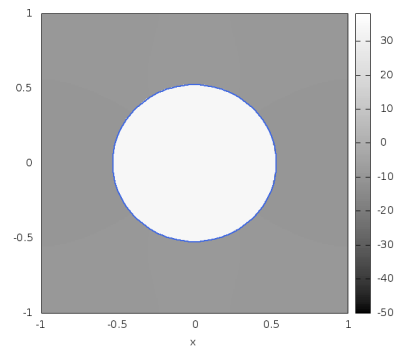
(a) $t=0.0$



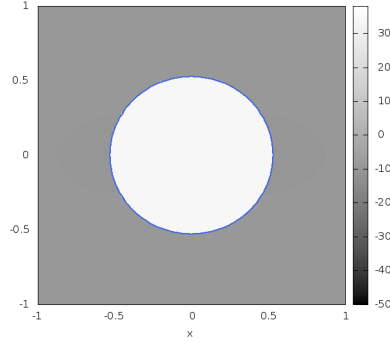
(b) $t=0.1$



(c) $t=0.2$



(d) $t=0.3$



(e) $t=0.4$

Figure 12: Configuration of the interface Γ_h at different times. The pressure values are also shown using different tones of gray, with a color table on the right. Dark regions have lower pressure; brighter regions have higher pressure.

5.4. Surface Tension

Let $\mathbf{x}(s, t)$ be the parameterization of the interface with respect to arclength. The surface tension force density (per unit interface length) at a given point $\mathbf{x}(s, t) \in \Gamma$ is:

$$\mathbf{f}^i(s, t) = 2\sigma \frac{\partial \boldsymbol{\tau}(s, t)}{\partial s} \quad (39)$$

where again $\boldsymbol{\tau}(s, t)$ is the unit vector tangential to the interface at the point $\mathbf{x}(s, t)$ and σ is the surface tension constant of the interface Γ . Since we are using the interface arclength as our parameterization, we have:

$$\left\| \frac{\partial \mathbf{x}}{\partial s} \right\| = 1, \quad (40)$$

The surface tension force density and the jump conditions are the same in this case:

$$\mathbf{f}_i^i = 2\sigma \frac{\boldsymbol{\tau}_{i+1/2} - \boldsymbol{\tau}_{i-1/2}}{\overline{\Delta s}}, \quad \overline{\Delta s} = \frac{\|\mathbf{x}_{i+1} - \mathbf{x}_i\| + \|\mathbf{x}_i - \mathbf{x}_{i-1}\|}{2}. \quad (41)$$

Here, $\boldsymbol{\tau}_{i+1/2}$ is the same as in equation (34). Note that the surface tension force density (39) is always normal to the interface. Because of this, the only discontinuous quantity across the interface is the pressure (the velocities are continuously differentiable across it).

For all elastic interface tests we use a rectangular periodic domain $[-1, 1] \times [-1, 1]$ with an elliptical interface of semi-major radius $a = 0.7$ and semi-minor radius $b = 0.4$ centered at the domain origin at $t = 0$. The fluid viscosity is again set to $\mu = 1$, the surface tension constant is $\sigma = 10$. We use the same number of interface nodes as in the elastic interface examples: $M = N/2$ segments. The nodes of the interface are positioned initially as in the elastic interface case. Also the time step is again $\Delta t = 5h^2 = 20/N^2$.

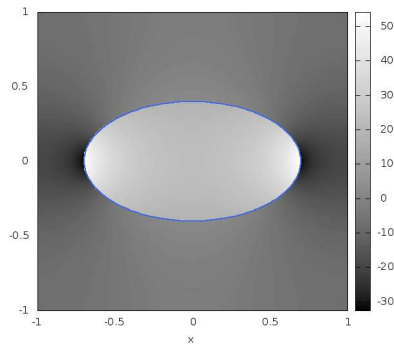
We verified the order of convergence for the pressure, for the velocities and for the positions of the interface nodes of Γ_h according to the L^∞ norm at the times $t_1 = 0.1$, $t_2 = 0.2$, $t_3 = 0.3$ and $t_4 = 0.4$. Also, we again checked the order of convergence of the total interface volume change ΔV_h towards zero as the grid was refined and the order of convergences of R_{\max} and R_{\min} . The numbers of MAC grid cells per direction used on the convergence tests were $N = 64, 128, 256$. As shown on the tables below, all the results obtained are second order accurate for all of these quantities:

$t_1 = 0.1$	
Quantity	Order of convergence
u	2.25
v	2.05
p	1.75
Γ_h	2.06
R_{\max}	2.24
R_{\min}	1.96
ΔV_h	1.83

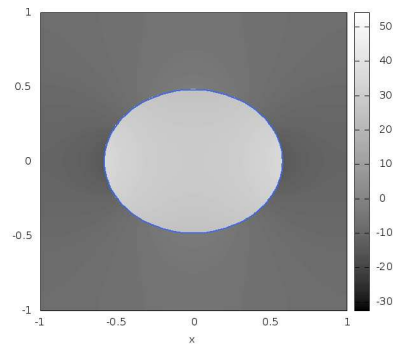
$t_2 = 0.2$	
Quantity	Order of convergence
u	1.90
v	1.84
p	1.79
Γ_h	1.95
R_{\max}	1.95
R_{\min}	1.96
ΔV_h	1.90

$t_3 = 0.3$	
Quantity	Order of convergence
u	1.99
v	2.06
p	1.92
Γ_h	2.37
R_{\max}	1.93
R_{\min}	1.97
ΔV_h	1.91

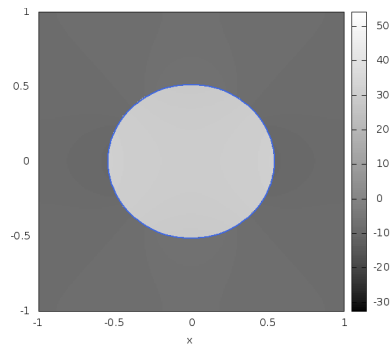
$t_4 = 0.4$	
Quantity	Order of convergence
u	2.05
v	2.12
p	2.05
Γ_h	2.10
R_{\max}	1.95
R_{\min}	1.97
ΔV_h	1.92



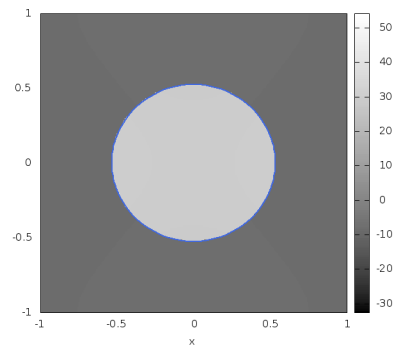
(a) $t=0.0$



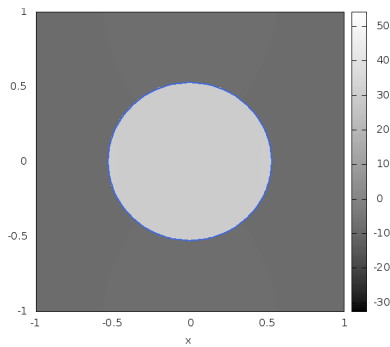
(b) $t=0.1$



(c) $t=0.2$



(d) $t=0.3$



(e) $t=0.4$

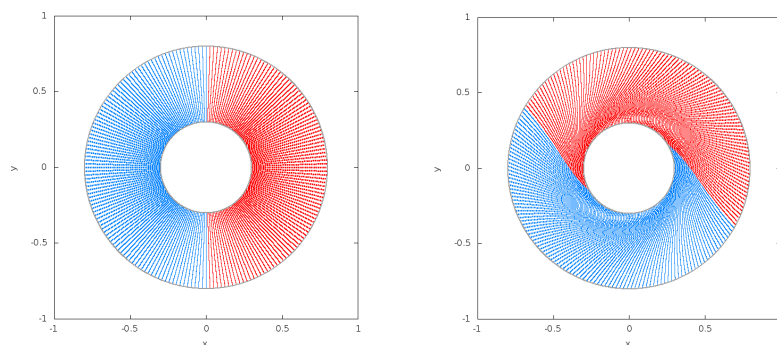
Figure 13: Configuration of the interface Γ_h at different times. The pressure values are also shown using different tones of gray, with a color table on the right. Dark regions have lower pressure; brighter regions have higher pressure. Compare these results with the ones of figure 12.

5.5. Irregular Domain

We also demonstrate second order convergence with the following irregular domain problem. Consider two concentric circles with radii $R_1 = 0.3$ and $R_2 = 0.8$ respectively with a Stokesian fluid in between having viscosity $\mu = 1$. The inner cylinder rotates counterclockwise with angular velocity $\omega_1 = 2$ and the outer circle rotates counterclockwise with angular velocity $\omega_2 = 1$. We used the values $N = 40, 80, 160, 320, 640$ for the MAC grid resolutions. The convergence rates of u and v are given on the table below.

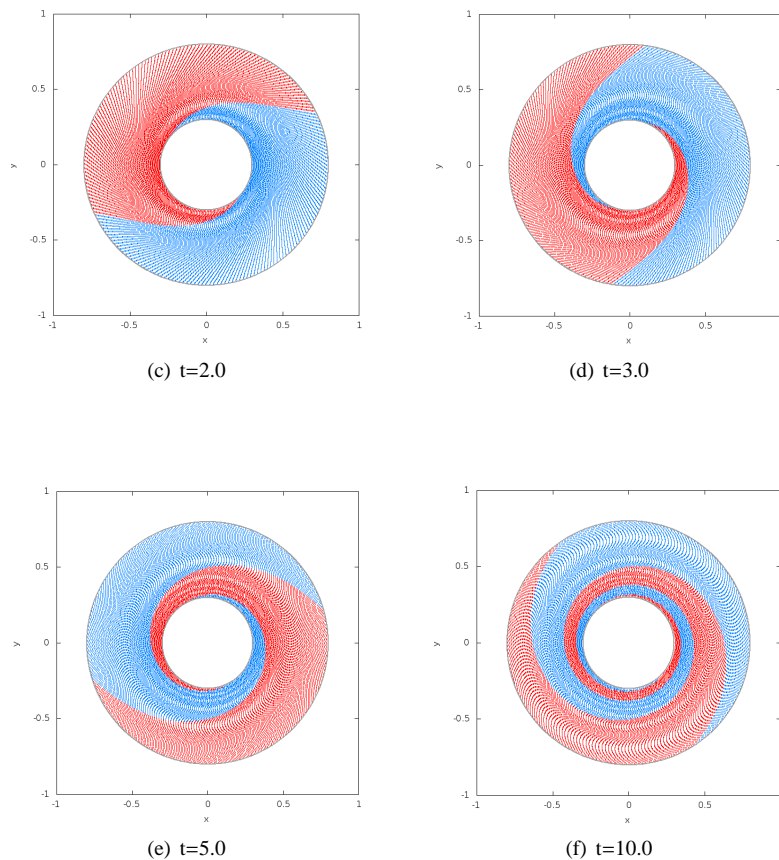
Quantity	Order of convergence
u	2.16
v	2.11

In order to visualize the fluid motion, we divided the fluid domain in two halves: $x > 0$ and $x < 0$. Marker particles were placed at these two domains and allowed to be advected with the fluid using Forward Euler. The figures below shows the position of these marker particles at different time steps:



(a) $t=0.0$

(b) $t=1.0$



6. Conclusion and future work

We have presented an efficient and easy to implement second order accurate method which solves the Stokes equations with immersed interfaces and geometrically irregular domains. The assumed spatial uniformity of the fluid viscosity permits a reduction of the problem to solving three independent Poisson equations for the velocities and the pressure. This task can be optimally performed using fast Poisson solvers such as multigrid. An additional Poisson equation can then be solved to enforce a discrete divergence free condition for the velocities without sacrificing second order accuracy. The linear system associated with each of these Poisson equations is symmetric positive definite. Examples with elastic interfaces and surface tension were presented as well as flow of a single fluid in the region separating two rotating concentric cylinders. All results obtained have shown second order accuracy.

It is the intention of the authors to extend the presented methods to allow for the solution of the Stokes equations with interfaces when the fluids involved have different viscosities. Difficulties arising from the coupling of the jump conditions across the contact interfaces [23] will then need to be taken into account.

References

- [1] 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.
- [2] A. Lew, G. Buscaglia, A discontinuous-Galerkin-based immersed boundary method, *Internat. J. Numer. Methods Engrg.* 76 (4) (2008) 427–454.
- [3] F. Harlow, J. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Phys. Fluids* 8 (12) (1965) 2182–2189.
- [4] C. Peskin, Flow patterns around heart valves: a numerical method, *J. Comput. Phys.* 10 (1972) 252–271.
- [5] C. S. Peskin, Numerical Analysis of Blood Flow in the Heart, *J. Comput. Phys.* 25 (1977) 220–+.

- [6] 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.
- [7] 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.
- [8] 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.
- [9] C. S. Peskin, The immersed boundary method, *Acta Numer.* 11 (2002) 479–517.
- [10] 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.
- [11] B. E. Griffith, C. S. Peskin, On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems, *J. Comput. Phys.* 208 (2005) 75–105.
- [12] 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.
- [13] A. M. Roma, C. S. Peskin, M. J. Berger, An adaptive version of the immersed boundary method, *J. Comput. Phys.* 153 (2) (1999) 509–534.
- [14] 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.
- [15] R. J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, *SIAM J. Numer. Anal.* 31 (1994) 1019–1044.
- [16] R. J. LeVeque, Z. Li, Immersed interface methods for stokes flow with elastic boundaries or surface tension, *SIAM J. Sci. Comput.* 18 (1997) 709–735.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] L. Lee, R. J. LeVeque, An immersed interface method for incompressible navier-stokes equations, *SIAM J. Sci. Comput.* 25 (2003) 832–856.
- [22] T. Y. Hou, Z. Li, S. Osher, H. Zhao, A hybrid method for moving interface problems with application to the hele-shaw flow, *J. Comput. Phys.* 134 (1997) 236–252.
- [23] 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.
- [24] 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.
- [25] Y. C. Zhou, S. Zhao, M. Feig, G. W. Wei, High order matched interface and boundary method for elliptic equations with discontinuous coefficients and singular sources, *J. Comput. Phys.* 213 (2006) 1–30.
- [26] S. Xu, Z. J. Wang, A 3d immersed interface method for fluid-solid interaction, *Comput. Methods Appl. Mech. Engrg* 197 (25-28) (2008) 2068–2086.
- [27] Z. Li, An overview of the immersed interface method and its applications, *Taiwanese J. Math.* 7 (1) (2003) 1–49.
- [28] 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.
- [29] M. Kang, R. P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, *J. Sci. Comput.* 15 (2000) 323–360.
- [30] M. Hyman, Non-iterative numerical solution of boundary-value problems, *Appl. Scientific Research, Section B* 2 (1) (1952) 325–351.
- [31] V. Sau’lev, On solving boundary value problems on high-performance computers by fictitious domain methods, *Siberian Mathematical J.* 4 (1963) 912–925.
- [32] R. Glowinski, T. Pan, T. Hesla, D. Joseph, A distributed Lagrange multiplier/fictitious domain method for particulate flows, *Int. J. Multiphase Flow* 25 (5) (1999) 755–794.
- [33] 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. Engrg* 112 (1-4) (1994) 133–148.
- [34] 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.
- [35] L. Parussini, V. Pediroda, Fictitious domain approach with hp-finite element approximation for incompressible fluid flow, *J. Comput. Phys.* 228 (2009) 3891–3910.
- [36] L. Parussini, Fictitious domain approach via lagrange multipliers with least squares spectral element method, *Journal of Scientific Computing* 37 (2008) 316–335.
- [37] F. Bertrand, P. Tanguy, F. Thibault, A three-dimensional fictitious domain method for incompressible fluid flow problems, *Internat. J. Numer. Methods Fluids* 25 (6) (1997) 719–736.
- [38] J. Teran, C. 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.
- [39] 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.
- [40] V. Rutka, A staggered grid-based explicit jump immersed interface method for two-dimensional stokes flows, *Internat. J. Numer. Methods Fluids* 57 (10) (2008) 1527–1543.

- [41] C. Daux, N. Moës, J. Dolbow, N. Sukumar, T. Belytschko, Arbitrary branched and intersecting cracks with the extended finite element method, *Internat. J. Numer. Methods Engrg.* 48 (12) (2000) 1741–1760.
- [42] 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. Engrg* 190 (46-47) (2001) 6183–6200.
- [43] A. Almgren, J. Bell, P. Colella, T. Marthaler, A cartesian grid projection method for the incompressible euler equations in complex geometries, *SIAM Journal of Scientific Computing* 18 (5) (1997) 1289–1309.
- [44] N. Moës, E. Béchet, M. Tourbier, Imposing dirichlet boundary conditions in the extended finite element method, *Internat. J. Numer. Methods Engrg.* 67 (12) (2006) 1641–1669.
- [45] J. Dolbow, A. Devan, Enrichment of enhanced assumed strain approximations for representing strong discontinuities: addressing volumetric incompressibility and the discontinuous patch test, *Internat. J. Numer. Methods Engrg.* 59 (1) (2004) 47–67.
- [46] G. Wagner, N. Moës, W. Liu, T. Belytschko, The extended finite element method for rigid particles in stokes flow, *Internat. J. Numer. Methods Engrg.* 51 (3) (2001) 293–313.
- [47] R. Becker, E. Burman, P. Hansbo, A nitsche extended finite element method for incompressible elasticity with discontinuous modulus of elasticity, *Comput. Methods Appl. Mech. Engrg* 198 (41-44) (2009) 3352–3360.
- [48] A. Coppola-Owen, R. Codina, Improving eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions, *Internat. J. Numer. Methods Fluids* 49 (12) (2005) 1287–1304.
- [49] J. Chessa, T. Belytschko, An extended finite element method for two-phase fluids, *J. Appl. Math.* 70 (1) (2003) 10–17.
- [50] A. Gerstenberger, W. Wall, An extended finite element method/lagrange multiplier based approach for fluid-structure interaction, *Comput. Methods Appl. Mech. Engrg* 197 (19-20) (2008) 1699–1714.
- [51] 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.
- [52] Y. Ng, C. Min, F. Gibou, An efficient fluid-solid coupling algorithm for single-phase flows, *J. Comput. Phys.* 228 (23) (2009) 8807–8829.
- [53] C. Batty, F. Bertails, R. Bridson, A fast variational framework for accurate solid-fluid coupling, *ACM Transactions on Graphics* 26.
- [54] I. Bijelonja, I. Demirdžić, S. Muzaferija, A finite volume method for incompressible linear elasticity, *Comput. Methods Appl. Mech. Engrg* 195 (44-47) (2006) 6378–6390.
- [55] L. Beirão da Veiga, V. Gyrya, K. Lipnikov, G. Manzini, Mimetic finite difference method for the stokes problem on polygonal meshes, *J. Comput. Phys.* 228 (19) (2009) 7215–7232.
- [56] P. Barton, D. Drikakis, An Eulerian method for multi-component problems in non-linear elasticity with sliding interfaces, *J. Comput. Phys.* 229 (15) (2010) 5518–5540.
- [57] D. Hill, D. Pullin, M. Ortiz, D. Meiron, An eulerian hybrid weno centered-difference solver for elastic-plastic solids, *J. Comput. Phys.* 229 (24) (2010) 9053–9072.
- [58] K. Ito, Z. Li, Interface conditions for stokes equations with a discontinuous viscosity and surface sources, *Appl. Math. Letters* 19 (3) (2006) 229 – 234.
- [59] F. H. Harlow, J. E. Welch, Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface, *Phys. Fluids* 8 (12) (1965) 2182–2189.