Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 6473–6489
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A grid-alignment finite element technique for incompressible multicomponent flows

B. Bejanov [a], J.L. Guermond [b,c], P.D. Minev [a,*]

[a] Department of Mathematical and Statistical Sciences, University of Alberta, Edmonton, Canada T6G 2G1
[b] Department of Mathematics, Texas A&M University 3368 TAMU, College Station, TX 77843-3368, USA
[c] LIMSI-CNRS UPR 3251, BP133 Orsay, France

Received 14 November 2006; received in revised form 28 January 2008; accepted 7 March 2008
Available online 20 March 2008

Abstract

We present a numerical method for solving the multicomponent incompressible Navier–Stokes equations. The methods employs a moving grid technique and a projection scheme based on low-order Crouzeix–Raviart and $\mathbb{P}_1$ finite elements. The computational grid is aligned with the fluid interface at each time step to keep the interpolation order optimal. The connectivity of the grid is fixed during the entire time simulation to ensure efficient assembling and parallelization. The method is tested on several examples.

Keywords: Free boundary problems; Finite element method; Incompressible flow

1. Introduction

Flow problems involving moving interfaces like shock waves, immiscible liquids, biological membranes, and many others, occur often in various applications. This paper presents an approximation technique for solving incompressible flows with sharp capillary interfaces between two or more immiscible fluids. There are several major techniques for tracking moving interfaces and computating surface tension. The reader is referred to [22] for a very comprehensive review on the subject. Well focused presentations of the developments of the volume-of-fluid (VOF) method can be found in [7,14]. For reviews on the level set methods the reader is referred to [17,15], and for front tracking techniques see [9,6]. For particular implementations of those techniques in the context of the finite element method the reader is referred to [4,16,21]. Very impressive results using adaptive soroban grids, a shock capturing technique, and a finite difference method have been recently presented by Yabe et al. [20]. These are just several of the most recent works on a subject that generated a large number of publications in the last two decades. This list of references is far from being

* Corresponding author.
E-mail addresses: bbejanov@math.ualberta.ca (B. Bejanov), guermond@math.tamu.edu (J.L. Guermond), minev@ualberta.ca (P.D. Minev).

0021-9991/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2008.03.011

complete; however, these publications together with the references therein, provide a fairly good picture of the techniques available in the literature.

In this paper we present a new technique, which evolved from our experience in this field (see [18,16,2,19]). Before that, we summarize the specific problems arising in free boundary problems and we show how we propose to address them in the present paper. The main difficulties arising in these problems are the following

- The proper approximation of the velocity and pressure in the vicinity of the free boundary. The Eulerian approach (VOF or level set methods for example) uses a fixed grid, and the free boundary intersects the edges of the grid cells (lattices in FD terminology or elements in the FE terminology). Such a discretization producing a smooth approximation within the intersected grid cells contradicts the fact that the exact solution or its derivatives may have a jump across the interface. Different ways to fix this problem have been proposed, such as the immersed interface method (see [11]) in case of finite difference approximations, the basis enrichment by [16], or a local mesh refinement (see [2]). But none of these methods is completely satisfactory. Local refinement changes significantly the grid structure around the boundary, which leads to problems with parallelization. Local basis enrichment is relatively easier to parallelize, but the spatial interpolation close to the interface is not optimal. The immersed interface method of Levesque and Li [11] is limited to the solution of the steady Stokes problem. Moreover, it can be used in the context of finite difference methods only. Finally, most Eulerian methods suffer from local mass conservation problems across the interface, which can produce so-called spurious velocities. The technique presented in this paper aligns the element faces with the free boundary at each time step, thus adapting the grid in the vicinity of the boundary, keeping invariant the grid connectivity, and preserving the interpolation optimality. In this respect the proposed method resembles the grid-based front tracking method of [8].
- Conservative and stable advection of free boundaries. The free boundary is controlled by a nonlinear advection equation which is well-known for the numerical difficulties that it poses. There is a large variety of stabilized methods dedicated to solving this problem, but no ultimate satisfactory solution exists. In the present paper, the volume of the different fluid particles involved in the motion of the interface is corrected by appropriately correcting the position of the nodes defining the free boundaries.
- Approximation of the curvature of the interface. Standard C^0-discretizations of free boundaries are not smooth enough to properly compute the boundary curvature, which is needed to account for the surface tension. Again, there is a variety of approaches involving, for example, cubic spline approximations (see [11]) or computing the curvature from a level set function (see [15]). The present technique employs a simple local regularization of the approximation, which was first briefly tried in [18]. It approximates the curvature in a way that the surface tension contribution is exactly equal to zero if the free boundary is circular (in 2D) or spherical (in 3D), thus eliminating spurious oscillations of the boundary. Spurious currents occur in most Eulerian methods (see [22,7]) and, although they are controlled by the truncation error of the discretization, they can be significant if the surface tension is large. Spurious oscillations can occur in Lagrangian methods too if the curvature approximation is not properly coordinated with the velocity approximation.

The remainder of the paper is organized as follows. In Section 2 we discuss the formulation of the problem. The details of the temporal and spatial discretizations are presented in Section 3. We detail the grid alignment algorithm and the algorithm for approximating the surface tension in Section 4. Numerical tests validating the method are reported in Section 5.

2. Formulation of the problem

Consider a flow of two incompressible, Newtonian, and immiscible fluids in a bounded domain Ω. In this paper we consider the 2D version of the method. The extension to 3D is not straightforward and requires some adjustments of the algorithms for tracking the free boundaries and the solution of the linear system arising from the projection step. Nevertheless, it has been tested and the 3D algorithm will be reported elswhere. The fluid $i (i \in \{1,2\})$ occupies a domain Ω_i, has a velocity $\mathbf{u}_i$, pressure p_i, density ρ_i, viscosity μ_i, and is subject to the gravity force. The density and viscosity in each phase are assumed to be constant. Ω_1 and Ω_2 do not overlap ($\Omega_1 \cap \Omega_2 = \emptyset$) and are separated by a smooth interface $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$. In all test cases presented in

Section 5 the domain Ω_2 consists of only one fluid particle. Therefore, for the sake of simplicity, this is assumed in the remainder of the paper and we will sometimes refer to Ω_2 as"the particle". It will become clear that the algorithm can be easily generalized to the case when neither of the fluids is distributed in the other fluid, or to the case of multiple fluid particles distributed in another fluid (as long as the distance between them is not too small). It is convenient to define a global velocity and pressure $\mathbf{u}$ and p such that $\mathbf{u} = \mathbf{u}_i, p = p_i$ in Ω_i and introduce the notations

$$\lambda_\rho = \begin{cases} 1 & \text{in } \Omega_1 \\ \frac{\rho_2}{\rho_1} & \text{in } \Omega_2 \end{cases} \quad \text{and} \quad \lambda_\mu = \begin{cases} 1 & \text{in } \Omega_1 \\ \frac{\mu_2}{\mu_1} & \text{in } \Omega_2. \end{cases} \tag{2.1}$$

Then, after proper nondimensionalization, we rewrite the Navier–Stokes equations in the following form:

$$\lambda_\rho \frac{D\mathbf{u}}{Dt} = \frac{\lambda_\mu}{Re}\nabla^2\mathbf{u} - \nabla p - \frac{\lambda_\rho}{Fr^2}\mathbf{e}_y \quad \text{in } \Omega, \quad \text{for } 0 < t \leqslant T \tag{2.2}$$

$$\nabla \cdot \mathbf{u} = 0, \quad \text{in } \Omega, \quad \text{for } 0 < t \leqslant T, \tag{2.3}$$

where $\mathbf{e}_y$ is the unit vector in the direction opposite to the direction of gravity, while Re and Fr are the Reynolds and Froude numbers

$$Re = \frac{\rho_1 u_c l_c}{\mu_1}, \quad Fr = \frac{u_c}{\sqrt{l_c g}}, \tag{2.4}$$

with u_c and l_c being the characteristic velocity and length, respectively. The quantity $\frac{D\mathbf{u}}{Dt}$ is the so-called material derivative of $\mathbf{u}$.

For simplicity we assume homogeneous Dirichlet boundary condition for the velocity on the external boundary

$$\mathbf{u}|_{\partial\Omega} = \mathbf{0} \quad \text{for } 0 < t \leqslant T. \tag{2.5}$$

On the interface Γ the velocity is continuous

$$\mathbf{u}_1|_\Gamma = \mathbf{u}_2|_\Gamma, \tag{2.6}$$

and the stress satisfies the following force balance condition

$$\left[p_2\mathbf{I} - p_1\mathbf{I} - \frac{1}{Re}\left(\frac{\mu_2}{\mu_1}\nabla\mathbf{u}_2 - \nabla\mathbf{u}_1\right)\right] \cdot \mathbf{n} = \frac{1}{We}\varkappa\mathbf{n} \quad \text{on } \Gamma, \tag{2.7}$$

where We is the Weber number

$$We = \frac{\rho_1 u_c^2 l_c}{\sigma}, \tag{2.8}$$

σ is the surface tension coefficient, $\varkappa$ is the curvature of Γ, and $\mathbf{n}$ is the unit normal pointing out of Ω_2. This condition differs from the standard force balance condition

$$\left[p_2\mathbf{I} - p_1\mathbf{I} - \frac{1}{Re}\left(\frac{\mu_2}{\mu_1}\left(\nabla\mathbf{u}_2 + (\nabla\mathbf{u}_2)^T\right) - \left(\nabla\mathbf{u}_1 + (\nabla\mathbf{u}_1)^T\right)\right)\right] \cdot \mathbf{n} = \frac{1}{We}\varkappa\mathbf{n} \quad \text{on } \Gamma. \tag{2.9}$$

It is a natural boundary condition for the momentum Eq. (2.2) and is therefore more convenient for deriving a weak formulation. Our previous numerical experience (see [2],[19]) shows that this condition gives results that compare very well with other numerical and experimental results on a wide range of parameters. Nevertheless, it can be expected that at low Reynolds numbers, (2.7) would not be accurate. One possible way to impose (2.9) is to use the semi-implicit approach proposed in [12]. As it will become clear below, the current splitting scheme can be easily modified to incorporate such an approach. It requires to add to Eq. (3.1) explicit terms accounting for the missing part of the divergence of the stress. Consequently, Eq. (3.3) must also be adjusted by adding implicitly all missing stress terms in the i-th momentum equation that involve the i-th component of the velocity. Using a normal modes analysis, it was demonstrated in [12] that such a semi-implicit scheme would be unconditionally stable. Of course, we can also treat the full stress tensor implicitly but this would destroy the block-diagonal structure of the momentum Eq. (3.3).

The initial velocity is assumed to be incompressible in each phase

$$\mathbf{u}|_{t=0} = \mathbf{u}_0, \quad \nabla \cdot \mathbf{u}_0 = 0 \quad \text{in } \Omega_i, \quad i = 1, 2. \tag{2.10}$$

In the sequel we make use of the following notation

$$\mathbf{L}^2(\Omega) = [L^2(\Omega)]^2, \quad \mathbf{H}_0^1(\Omega) = [H_0^1(\Omega)]^2,$$
$$\mathbf{H} = \{\mathbf{v} \in \mathbf{L}^2(\Omega); \quad \nabla \cdot \mathbf{v} \in L^2(\Omega); \quad \mathbf{v} \cdot \mathbf{n}|_{\partial\Omega} = 0\}.$$

Multiplying (2.2) by a test function $\mathbf{v} \in \mathbf{H}_0^1(\Omega)$, integrating by parts the stress terms, and incorporating the stress balance condition (2.7), we obtain the following weak formulation of the momentum equation:

$$\lambda_\rho \left(\frac{D\mathbf{u}}{Dt}, \mathbf{v}\right) + \frac{\lambda_\mu}{Re}(\nabla \mathbf{u}, \nabla \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) = -\frac{\lambda_\rho}{Fr^2}(\mathbf{e}_y, \mathbf{v}) - \frac{1}{We} \int_\Gamma \varkappa \mathbf{v} \cdot \mathbf{n} \, dl, \quad \forall \mathbf{v} \in \mathbf{H}_0^1(\Omega). \tag{2.11}$$

The weak form of the incompressibility constraint is given by

$$(\nabla \cdot \mathbf{u}, q) = 0, \quad \forall q \in L^2(\Omega). \tag{2.12}$$

3. Discretization

3.1. Time splitting

The weak formulation is discretized in time by a first-order standard version of the velocity-correction scheme discussed in [10].

Let N be a positive integer. We divide the time interval $[0, T]$ into N subintervals of equal length $\Delta t = T/N$ and denote the time levels by $t_n = n\Delta t, n = 0, \ldots, N$. At each time level n the velocity-correction scheme requires two approximations for the velocity $\tilde{\mathbf{u}}^n \in \mathbf{H}_0^1(\Omega)$ and $\mathbf{u}^n \in \mathbf{H}$, as well as an approximation for the pressure $p^n \in L^2(\Omega)$. Assuming that $\tilde{\mathbf{u}}^n$ and $\mathbf{u}^n$ are known at some time t_n (at $n = 0$ we choose $\tilde{\mathbf{u}}^0 = \mathbf{u}^0 = \mathbf{u}_0$) the splitting proceeds as follows:

- Projection step: Find $\mathbf{u}^{n+1} \in \mathbf{H}$ and $p^{n+1} \in L^2(\Omega)$ such that for all $\boldsymbol{\phi} \in \mathbf{H}_0^1(\Omega)$ and $q \in L^2(\Omega)$

$$\lambda_\rho \left(\frac{\mathbf{u}^{n+1} - \tilde{\mathbf{u}}^n}{\Delta t}, \boldsymbol{\phi}\right) + \lambda_\rho((\tilde{\mathbf{u}}^n \cdot \nabla)\tilde{\mathbf{u}}^n, \boldsymbol{\phi}) + \frac{\lambda_\mu}{Re}(\nabla \tilde{\mathbf{u}}^n, \nabla \boldsymbol{\phi}) - (p^{n+1}, \nabla \cdot \boldsymbol{\phi}) = -\frac{\lambda_\rho}{Fr^2}(\mathbf{e}_y, \boldsymbol{\phi}) - \frac{1}{We} \int_\Gamma \varkappa \boldsymbol{\phi} \cdot \mathbf{n} \, dl, \tag{3.1}$$

$$(\nabla \cdot \mathbf{u}^{n+1}, q) = 0. \tag{3.2}$$

- Correction step: Find $\tilde{\mathbf{u}}^{n+1} \in \mathbf{H}_0^1(\Omega)$ such that for all $\boldsymbol{\phi} \in \mathbf{H}_0^1(\Omega)$

$$\lambda_\rho \left(\frac{\tilde{\mathbf{u}}^{n+1} - \mathbf{u}^{n+1}}{\Delta t}, \boldsymbol{\phi}\right) + \frac{\lambda_\mu}{Re}(\nabla \tilde{\mathbf{u}}^{n+1} - \nabla \tilde{\mathbf{u}}^n, \nabla \boldsymbol{\phi}) = 0. \tag{3.3}$$

Note that the projection step is a mixed form of a Poisson equation for the pressure.

3.2. Spatial discretization

We recently developed a pressure-correction projection scheme using a combination of $\mathbb{P}_1$ and non-conforming linear Crouzeix–Raviart finite elements (see [3]) for the velocity and a piecewise constant pressure. Here we will only present a brief description of how the same idea is applied to the velocity-correction scheme above. For a detailed description and analysis the reader is referred to [1].

Computational grids on Ω are denoted by $\mathcal{G}_{h,n} = (\mathcal{N}_{h,n}, \mathcal{E}_{h,n}, \mathcal{T}_{h,n})$ where h is the characteristic meshsize. The mesh is composed of conforming triangular elements. The set of all triangles is denoted by $\mathcal{T}_{h,n}$. The sets of the vertices and the edges of all the triangles in $\mathcal{T}_{h,n}$ are denoted by $\mathcal{N}_{h,n}$ and $\mathcal{E}_{h,n}$, respectively. We set $N_N := \text{card}(\mathcal{N}_{h,n})$, $N_E := \text{card}(\mathcal{E}_{h,n})$ and $N_T := \text{card}(\mathcal{T}_{h,n})$. We also denote by $\mathcal{M}_{h,n}$ the set of the midpoints

of the edges. As it will become clear below, the grids used in the computations are different for each time level and therefore, we use the subscript n to denote the time level to which the given set corresponds. Also, from now on we use a superscript m to denote the time level to which a given discrete quantity, say r_h, corresponds, while a subscript n denotes the time level of the grid on which r_h is defined, i.e., $r^m_{h,n}$ is the quantity r_h computed at time level m on the grid $\mathcal{G}_{h,n}$.

Let us define the following discrete spaces

$$\mathbf{U}_{h,n} = \{\mathbf{u}_h \in H^1_0(\Omega)^2 \mid \mathbf{u}_h|_\tau \in \mathbb{P}^2_1, \text{for all } \tau \in \mathcal{T}_{h,n}\}, \tag{3.4}$$

$$\mathbf{V}_{h,n} = \{\mathbf{v}_h \in L^2(\Omega)^2 \mid \mathbf{v}_h|_\tau \in \mathbb{P}^2_1, \text{for all } \tau \in \mathcal{T}_{h,n}, \mathbf{v}_h \text{ is continuous at all } m \in \mathcal{M}_{h,n}\}, \tag{3.5}$$

$$\mathbf{W}_{h,n} = \{\mathbf{w}_h \in \mathbf{V}_{h,n} \mid (\nabla \cdot \mathbf{w}_h)|_\tau = 0, \text{for all } \tau \in \mathcal{T}_{h,n}\}, \tag{3.6}$$

$$Q_{h,n} = \{q \in L^2(\Omega) \mid q|_\tau = \text{const}, \text{ for all } \tau \in \mathcal{T}_{h,n}\}, \tag{3.7}$$

where $\mathbb{P}_1$ is the space of linear polynomials of two variables. For a given grid $\mathcal{G}_{h,n}$, $\mathbf{U}_{h,n}$ is the standard functional space associated with $\mathbb{P}_1$ elements, $\mathbf{V}_{h,n}$ is the space of velocities of the linear Crouzeix–Raviart elements, $\mathbf{W}_{h,n}$ is the divergence free subspace of $\mathbf{V}_{h,n}$, and $Q_{h,n}$ is the space of piecewise constant pressures.

At each time level m and for each grid $\mathcal{G}_{h,n}$, we have two discrete velocities $\tilde{\mathbf{u}}^m_{h,n} \in \mathbf{U}_{h,n}$ and $\mathbf{u}^m_{h,n} \in \mathbf{W}_{h,n}$. The projection step is approximated by Crouzeix–Raviart elements and the velocity correction step by $\mathbb{P}_1$ elements. The advantage of this setting is that the projected velocity is locally mass conservative, which is particularly important when multicomponent flows are considered, while the correction step is resolved with relatively fewer degrees of freedom using $\mathbb{P}_1$ triangles (two times the number of nodes in the grid) compared to the degrees of freedom involved in a Crouzeix–Raviart approximation (twice the number of edges). This setting is not *inf-sup* stable and indeed, at very small time steps spurious oscillations in the pressure can be observed. However, as shown in [1], it is optimally accurate for the velocity if the so-called cross grids are used. In addition, some heuristic arguments and the numerical evidence provided in [1] demonstrate that the locking of the velocity which is typical for the $P_1 - P_0$ element on diagonal grids is prevented in this setting. In the same reference, the authors provide an example of a postprocessing procedure for the pressure which filters out the spurious modes and provides optimal pressure approximation. But in order to insure an optimal convergence rate for the velocity, the computational grid must be aligned with the interface at each time step. We also require that the grid alignment does not change the connectivity of the grid and does not introduce any new nodes, edges, elements and/or degrees of freedom. This guarantees that, even though the grid is different and all the matrices must be reassembled, the structure of the matrices remains unchanged which facilitates the parallelization of the algorithm. Morover, only a small number of entries in the matrices needs be changed, since we confine the alignment of the grid to the narrow strip of elements containing the interface; the reassembly procedure is thus greatly accelerated.

The general idea is to maintain a reference grid $\mathcal{R}_h$, which can be any triangulation of Ω so as to produce a grid that meets our requirements. Assuming that the current position of the interface is known, we can find the nodes in $\mathcal{R}_h$ that are the closest to Γ and move them so as to position them on Γ, provided the required displacement does not exceed the diameters of the cells that are involved. The new grid is then aligned with the interface in the sense that Γ does not intersect any edge and is approximated by a set of edges of elements in the modified grid. These edges form a piecewise linear approximation of the interface.

Suppose that $\mathcal{G}_{h,n}$ is constructed so that it fits Γ at time t_n. Suppose also that $\tilde{\mathbf{u}}^n_{h,n}$ and $\mathbf{u}^n_{h,n}$ are known. Then the algorithm proceeds to the next time level as follows.

- *Calculation of the position of the interface at time* t_{n+1}. The curve formed by all the edges of $\mathcal{G}_{h,n}$ that belong to the interface provides a piecewise linear approximation to the interface at time t^n which we denote by $\Gamma^n_{h,n}$. The motion of the free boundary is approximated by moving the nodes of $\Gamma^n_{h,n}$ along the characteristics of the flow that pass at time t_n through the nodes. For the approximation of the characteristics we need to know explicitly the fluid velocity in each of the nodes. In the present case, the only available approximation to this velocity which is defined nodewise is the end of step velocity $\tilde{\mathbf{u}}^n_{h,n}$. On the other hand, the velocity resulting from the projection step (3.1), $\mathbf{u}^n_{h,n}$, is locally conservative (divergence-free within each element) and so, more appropriate to be used as an advection velocity. However, it is continuous only at the midpoints of the element edges and cannot be used directly for approximating the characteristics passing

through the nodes of the grid. In order to use $\mathbf{u}^n_{h,n}$ to advect the boundary, we first project it onto $\mathbf{U}_{h,n}$ with an L^2-projection. The resulting velocity, which we denote by $\hat{\mathbf{u}}^n_{h,n}$, is no longer locally conservative, however, its divergence is significantly smaller than that of $\tilde{\mathbf{u}}^n_{h,n}$. Moreover, since it is defined at the grid nodes, we use it to apply the method of characteristics to advect the grid. Let $\mathbf{x}_i \in \mathcal{N}_h$ be a node in the reference grid $\mathcal{R}_h$. We use $\mathbf{x}^n_i \in \mathcal{N}_{h,n}$ to denote the position of this node in the current grid $\mathcal{G}_{h,n}$. The position of this node after advection is denoted by $\hat{\mathbf{x}}^{n+1}_i$ and is computed via the following Euler explicit scheme

$$\frac{\hat{\mathbf{x}}^{n+1}_i - \mathbf{x}^n_i}{\Delta t} - \hat{\mathbf{u}}^n_{h,n}(\mathbf{x}^n_i) = 0. \tag{3.8}$$

When applying the maps $\mathbf{x}^n_i \mapsto \hat{\mathbf{x}}^{n+1}_i$ to the entire grid $\mathcal{G}_{h,n}$ we obtain the so-called advected grid $\hat{\mathcal{G}}_{h,n+1}$. The image of $\Gamma^n_{h,n}$ by the above map provides an approximation $\Gamma^{n+1}_{h,n}$ for the position of the free boundary at t_{n+1}. Note that no computation is performed on the advected grid $\hat{\mathcal{G}}_{h,n+1}$; this grid is used only to define the new position of the interface for the purpose of grid alignment and for interpolating the velocity if one chooses the method of characteristics to approximate the nonlinear term at the projection step described below.

- *Reference grid alignment with $\Gamma^{n+1}_{h,n}$ and generation of $\mathcal{G}_{h,n+1}$*. The details of this procedure are given in Section 4 below. Note that the nodes in $\mathcal{G}_{h,n+1}$ that define the interface are not necessarily the same as the nodes on $\Gamma^{n+1}_{h,n}$, i.e., we now have a new approximation of the interface, $\Gamma^{n+1}_{h,n+1}$, embedded in $\mathcal{G}_{h,n+1}$.
- *Projection step*. We now describe three possible ways for implementing the projection step. These possibilities depend on how the approximation of $\tilde{\mathbf{u}}^n$ is constructed on the new grid $\mathcal{G}_{h,n+1}$ and how the nonlinear term is treated. Since the velocity $\tilde{\mathbf{u}}^n_{h,n}$ is only defined on the grid $\mathcal{G}_{h,n}$ we need to transfer it to the new grid $\mathcal{G}_{h,n+1}$ in order to perform the current and the next step.

The first option we consider consists of interpolating $\tilde{\mathbf{u}}^n_{h,n}$ on $\mathcal{G}_{h,n+1}$, i.e., we define $\tilde{\mathbf{u}}^n_{h,n+1} \in \mathbf{U}_{h,n+1}$ so that

$$\tilde{\mathbf{u}}^n_{h,n+1}(\mathbf{x}^{n+1}_i) = \tilde{\mathbf{u}}^n_{h,n}(\mathbf{x}^{n+1}_i), \quad \forall \mathbf{x}^{n+1}_i \in \mathcal{N}_{h,n+1}. \tag{3.9}$$

Then the projection step is approximated as follows: Find $\mathbf{u}^{n+1}_{h,n+1} \in \mathbf{W}_{h,n+1}$ such that for all $\boldsymbol{\psi}_h \in \mathbf{W}_{h,n+1}$

$$\begin{aligned}\lambda_\rho \left\{ \frac{\mathbf{u}^{n+1}_{h,n+1} - \tilde{\mathbf{u}}^n_{h,n+1}}{\Delta t}, \boldsymbol{\psi}_h \right. &+ \lambda_\rho((\tilde{\mathbf{u}}^n_{h,n+1}\cdot\nabla)\tilde{\mathbf{u}}^n_{h,n+1}, \boldsymbol{\psi}_h) + \frac{\lambda_\mu}{Re}(\nabla\tilde{\mathbf{u}}^n_{h,n+1}, \nabla\boldsymbol{\psi}_h) \\ &= -\frac{\lambda_\rho}{Fr^2}(\mathbf{e}_y, \boldsymbol{\psi}_h) - \frac{1}{We}\int_\Gamma \varkappa_{n+1}\boldsymbol{\psi}_h\cdot\mathbf{n}\,\mathrm{d}l.\end{aligned} \tag{3.10}$$

When the ratio of the two viscosities is large, the velocity may vary significantly across the interface and thus the interpolation process producing $\tilde{\mathbf{u}}^n_{h,n+1}$ may not be very accurate. This leads to an additional stability restriction, which can be severe for large viscosity ratios λ_μ. Note, that this problem is even more severe if the pressure from the previous time step needs to be transfered, as a pressure-correction method would require, the reason being that the pressure is usually discontinuous across the interface and such an interpolation would not be consistent. This is the main reason why we choose a velocity-correction technique rather than a pressure-correction scheme to march in time. Two possible alternatives to the above interpolation problem are proposed below.

The second option consists of using the forward method of characteristics for integration of the interface trajectory and the advection terms in the Navier–Stokes equations. The method of characteristics is known to possess better stability properties than most other explicit methods. The numerical tests below confirm that it has the best stability properties among the three possibilities considered in this paper. After advecting the nodes and producing the advected grid according to (3.8), we define the advected velocity $\bar{\mathbf{u}}^{n+1}_{h,n} \in \mathbf{U}_{h,n+1}$ based on $\mathcal{G}_{h,n+1}$ via $\bar{\mathbf{u}}^{n+1}_{h,n}(\hat{\mathbf{x}}^{n+1}_i) = \tilde{\mathbf{u}}^n_{h,n}(\mathbf{x}^n_i)$.

If the time step is small enough so that the elements in $\mathcal{G}_{h,n+1}$ have positive Jacobeans (this is the stability restriction of the method), these nodal values of the advected velocity define a piecewise linear interpolant on the advected grid $\mathcal{G}_{h,n+1}$. The transfered velocity $\tilde{\mathbf{u}}^n_{h,n+1} \in \mathbf{U}_{h,n+1}$ is produced by interpolating the values of $\bar{\mathbf{u}}^{n+1}_{h,n}$ at the nodes of $\mathcal{G}_{h,n+1}$. Since the advection terms in the momentum equation are already taken into account in the definition of $\tilde{\mathbf{u}}^n_{h,n+1}$ the projection step reduces to the following: Find $\mathbf{u}^{n+1}_{h,n+1} \in \mathbf{W}_{h,n+1}$ such that for all $\boldsymbol{\psi}_h \in \mathbf{W}_{h,n+1}$

$$\lambda_\rho \left\langle \frac{\mathbf{u}^{n+1}_{h,n+1} - \tilde{\mathbf{u}}^{n}_{h,n+1}}{\Delta t}, \boldsymbol{\psi}_h \right\rangle + \frac{\lambda_\mu}{Re}(\nabla \tilde{\mathbf{u}}^{n}_{h,n+1}, \nabla \boldsymbol{\psi}_h) = -\frac{\lambda_\rho}{Fr^2}(\mathbf{e}_y, \boldsymbol{\psi}_h) - \frac{1}{We} \int_\Gamma \varkappa_{n+1} \boldsymbol{\psi}_h \cdot \mathbf{n}\, \mathrm{d}l. \tag{3.11}$$

The third alternative consists of using an Arbitrary Eulerian–Lagrangian (ALE) approach (see [5]). Both grids, $\mathcal{G}_{h,n}$ and $\mathcal{G}_{h,n+1}$ are produced from the same reference grid by only changing the position of some nodes. Therefore, there are natural one-to-one mappings between the nodes $\mathbf{x}^n_i$ in $\mathcal{G}_{h,n}$, the nodes $\mathbf{x}^{n+1}_i$ in $\mathcal{G}_{h,n+1}$ and the nodes $\mathbf{x}_i$ in the reference grid. Then we define the discrete mesh velocity via

$$\check{\mathbf{u}}^n_i = \frac{\mathbf{x}^{n+1}_i - \mathbf{x}^n_i}{\Delta t} \tag{3.12}$$

This discrete velocity can be extended to a piecewise linear velocity using the standard finite element interpolation. For this purpose, we define $\check{\mathbf{u}}^n_{h,n+1} \in \mathbf{U}_{h,n+1}$, such that

$$\check{\mathbf{u}}^n_{h,n+1}(\mathbf{x}^{n+1}_i) = \check{\mathbf{u}}^n_i, \quad \forall \mathbf{x}^{n+1}_i \in \mathcal{N}_{h,n+1}. \tag{3.13}$$

We also define the transferred material velocity $\tilde{\mathbf{u}}^n_{h,n+1} \in \mathbf{U}_{h,n+1}$ as

$$\tilde{\mathbf{u}}^n_{h,n+1}(\mathbf{x}^{n+1}_i) = \tilde{\mathbf{u}}^n_{h,n}(\mathbf{x}^n_i). \tag{3.14}$$

With these definitions, the projection step 3.10 becomes: Find $\mathbf{u}^{n+1}_{h,n+1} \in \mathbf{W}_{h,n+1}$ such that for all $\boldsymbol{\psi}_h \in \mathbf{W}_{h,n+1}$

$$\begin{aligned} &\lambda_\rho \left\langle \frac{\mathbf{u}^{n+1}_{h,n+1} - \tilde{\mathbf{u}}^{n}_{h,n+1}}{\Delta t}, \boldsymbol{\psi}_h \right\rangle + \lambda_\rho((\mathbf{u}^n_{h,n+1} \cdot \nabla)\tilde{\mathbf{u}}^n_{h,n+1}, \boldsymbol{\psi}_h) - \lambda_\rho(\nabla \cdot (\check{\mathbf{u}}^n_{h,n+1}\tilde{\mathbf{u}}^n_{h,n+1}), \boldsymbol{\psi}_h) + \frac{\lambda_\mu}{Re}(\nabla \tilde{\mathbf{u}}^n_{h,n+1}, \nabla \boldsymbol{\psi}_h) \\ &\quad = -\frac{\lambda_\rho}{Fr^2}(\mathbf{e}_y, \boldsymbol{\psi}_h) - \frac{1}{We} \int_\Gamma \varkappa_{n+1} \boldsymbol{\psi}_h \cdot \mathbf{n}\, \mathrm{d}l. \end{aligned} \tag{3.15}$$

These methods for transfering data between the two grids $\mathcal{G}_{h,n}$ and $\mathcal{G}_{h,n+1}$ are evaluated numerically in Section 5; the characteristic (second) approach seems to be the most stable when the viscosity ratio is large.

- *Correction step*. Correcting the velocity does not pose any particular difficulty. This step is formulated as follows: Find $\tilde{\mathbf{u}}^{n+1}_{h,n+1} \in \mathbf{U}_{h,n+1}$ such that for all $\boldsymbol{\phi}_h \in \mathbf{U}_{h,n+1}$

$$\lambda_\rho \left\langle \frac{\tilde{\mathbf{u}}^{n+1}_{h,n+1} - \mathbf{u}^{n+1}_{h,n+1}}{\Delta t}, \boldsymbol{\phi}_h \right\rangle + \frac{\lambda_\mu}{Re}(\nabla \tilde{\mathbf{u}}^{n+1}_{h,n+1} - \nabla \tilde{\mathbf{u}}^n_{h,n+1}, \nabla \boldsymbol{\phi}_h) = 0 \tag{3.16}$$

4. Alignment of the grid

Let us assume that the reference grid $\mathcal{R}_h$ and a piecewise linear approximation of the interface Γ are given. There are many different ways to produce a new grid $\mathcal{G}_h$, which has the same connectivity as $\mathcal{R}_h$ and is aligned with Γ. We use the algorithm outlined below.

4.1. General algorithm

The steps of the algorithm are:

(1) Determine the phase (either Ω_1 or Ω_2) to which the nodes in $\mathcal{R}_h$ belong and label them appropriately with a phase marker. More details are given in the next subsection.
(2) Create a list containing all the edges intersecting the interface and a list of the end points of those edges (called pending nodes) using the phase marker defined in the previous item. In addition, create an edge counter associated with each pending node, which counts the number of edges connected to this node and intersecting the interface.
(3) Find the projection on Γ of each pending node and calculate the distance to the interface. Then sort the list of the pending nodes according to their distance to Γ.

(4) Process the pending nodes starting with the closest to Γ. If a pending node is on the interface or on the outer boundary of Ω, it is removed from the list of pending nodes. Otherwise, it is replaced by its projection on Γ and removed from the pending nodes list. All the edges connected to this node and previously intersecting Γ are deleted from the intersected edges list and the edge counters of all the end points of those edges are decreased by one. A pending node whose edge counter becomes 0 is deleted from the list. At the end of this procedure Γ does not cross any edge.

(5) The projection of nodes on the interface can severely decrease the quality of the grid in its vicinity (as a measure for the quality of an element we use the ratio of the radii of the inscribed and circumscribed circles). It can also create finite elements with negative Jacobeans of their affine mapping to the standard triangle. Therefore, we apply a simple local mesh regularization procedure which proceeds as follows. First we check for each node which is on the interface, or is connected via an edge to an interface node, if it is inside the polygon defined by the nodes connected to it with an edge. A node that is outside of its polygon and is not on the interface, is moved to the centroid of the polygon. An interface node which is outside of its polygon is moved to the centroid only if in the new configuration neither of the edges connected to it are intersected by the interface again. Otherwise, the quality of the given element is improved when some of the neighbouring interface points are removed from the interface. Note that if the surrounding polygon is non-convex and the indentation of its surface is extremely large, the centroid itself may be outside the polygon; however, this has never occurred on the meshes that we have used so far (see Fig. 1).

(6) Check if the grid contains elements having three nodes on the interface. Since for such an element it is difficult to determine to which fluid it belongs, one of its nodes is moved to the centroid of its polygon. Note that there is at least one node in the element such that all of its neighbours, except those on Γ, belong to the same phase (see Fig. 2), and this is the node which is removed from the interface.

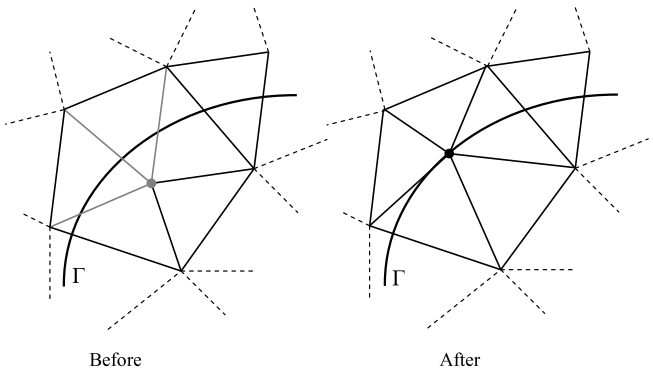


Fig. 1. Projecting a pending node on the interfaces. In the new position, no edge coming out of the node intersects Γ.

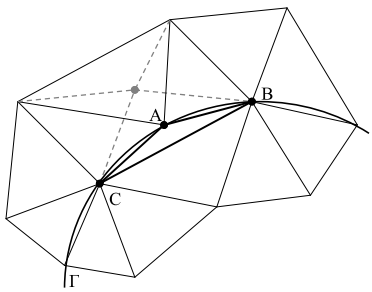


Fig. 2. Element with three nodes on Γ. At least one node (A) will have all neighbours on the same side of Γ.

(7) Verify the integrity of the new approximation of the interface. Since we presume that it is a closed curve without self intersections, at this stage of the algorithm each node on the interface should have exactly two neighbours that are also on the interface.

(8) Correct the volume of the region enclosed by the approximate interface. Note, that the grid alignment can lead to mass loss since the nodes moved to the interface are not necessarily moved to the positions occupied previously by interface nodes. One way to guarantee global mass conservation is to enforce it during the projection on the interfaces as a side constraint, using one Lagrange multiplier for each interface. We use a simpler procedure which nevertheless has performed excellently in all test cases that we considered. All interface nodes are moved away from or towards the centre of mass of the particle, until its proper volume is restored. If $\mathbf{x}_j^{\text{old}}$ are the coordinates of a node on Γ_h before the volume correction then its new coordinates $\mathbf{x}_j^{\text{new}}$ are computed according to:

$$\mathbf{x}_j^{\text{new}} = \mathbf{x}_c + \left(\frac{V}{V^{\text{old}}}\right)^{1/d} (\mathbf{x}_j^{\text{old}} - \mathbf{x}_c),$$

where $\mathbf{x}_c$ are the coordinates of the centre of mass of the particle, $d = 2$ is the number of spatial dimensions, V is the desired volume and V^{old} is the volume before correction. If the correction is performed at each time step, the factor $(\frac{V}{V^{\text{old}}})^{1/d}$ is of order of the truncation error of the method and therefore this correction does not spoil the consistency of the discretization.

(9) At the volume correction step, some of the elements around the boundary can become intolerably stretched. In such case the nodes near the interface (but not on it) are moved to the centroids of their polygons once more.

A major advantage of this algorithm is that it preserves the grid connectivity at all time levels and only alters the position of some nodes near the interface. This allows for a straightforward and balanced parallelization of the linear solvers because the matrix structure remains the same at all time steps. In addition, only the entries corresponding to nodes that were moved or their neighbours need to be updated so that the matrix updates are computationally inexpensive. In addition, the mass of the particle is conserved at each time step.

4.2. Computing the phase marker and projection of a point on the boundary

Let us introduce a discontinuous marker function having value 1 in phase 1 and 2 in phase 2 (more phases can be introduced). Suppose also, that the grid at $t = 0$ is properly aligned with the interface and that the marker function is properly initialized, i.e., all nodes are labeled with the number of the domain that they correspond to. The nodes on the interface are marked with a separate marker indicating the number of the interface and the two domains separated by it. At the first step of the solution algorithm described in Section 3.2, all nodes of the grid $\mathcal{G}_{h,n}$ are advected alongside an approximation of the characteristics passing through them. As a result, if they have been properly labeled at time level n their labels in the new position of the grid define an approximation of the marker function at time level $n + 1$. In order to compute its values in the nodes of the reference grid we only need to locate the element in the advected grid containing a given node of the reference grid.

To find this element quickly, we start searching with the elements connected to the given node, and if necessary the search continues within their neighbours, etc.

The first step of the solution algorithm described in Section 3.2, yields a piecewise linear approximation of the interface, $\Gamma_{h,n}^{n+1}$, which contains a number of points and the line segments connecting them. In the next step, we need to construct a new piecewise linear approximation of the interface, $\Gamma_{h,n+1}^{n+1}$, by projecting the nearest points of the reference grid onto $\Gamma_{h,n}^{n+1}$. Since $\Gamma_{h,n}^{n+1}$ is not smooth, the projection of a point onto it can often cause nonphysical oscillations. This effect can be avoided by employing a smooth approximation with cubic splines, for example, but the spline computation is costly and it is relatively difficult to generalize to 3D. An alternative used in this study, which smooths Γ only locally, is to approximate it with an arc of a circle. Suppose that we have a point P anywhere in Ω and we need to construct its projection $P\prime$ onto a given interface $\Gamma_{h,n}^{n+1}$. Then the projection procedure employed in the present technique can be summarized as follows (see Fig. 3 for the geometric details):

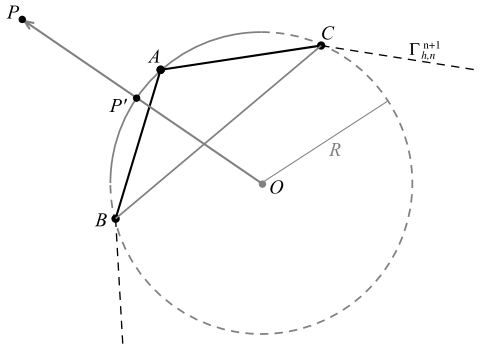


Fig. 3. Construction of the projection P' of a point P on the interface by approximating Γ locally with a circle.

- Compute the distance from P to every point on $\Gamma_{h,n}^{n+1}$. Find the point A in $\Gamma_{h,n}^{n+1}$ that is closest to P.
- Find points B and C – the left and right neighbours of A on $\Gamma_{h,n}^{n+1}$.
- Construct point O – the circumcentre of $\triangle ABC$. Also compute the radius R of the circumscribed circle.
- Construct point $P\prime$ – it is the point of intersection of the circumscribed circle of $\triangle ABC$ and the line OP. We compute its position as the point on the ray $\overrightarrow{OP}$, which is at a distance R from O.

An advantage of this procedure is that, if Γ is an exact circle, then the points of the new approximation $\Gamma_{h,n+1}^{n+1}$ will be positioned on this circle. Consequently, as shown in the next subsection, the surface tension can be approximated so that its contribution is equal to zero. This prevents the formation of the so called parasitic currents around a circular interface at rest (see [22]). Many Eulerian and even some Lagrangian algorithms, suffer of nonphysical oscillations of the capillary interface due to such currents.

4.3. Computation of the surface tension

For a given point on the interface, the curvature and the outer normal are approximated with the curvature and the normal to the locally circular approximation constructed at this point, as explained in the previous section. Note that this approximation is used only for their computation, and the surface tension integral in (3.10) is computed on the piecewise linear approximation of the interface comprised by the elemental edges aligned with it. Therefore, the normal vector toward a given edge e_k is constant. It is approximated as the average of the two endpoint unit normal vectors $\mathbf{n}_{k,0}$ and $\mathbf{n}_{k,1}$, which is then normalized to an unit length. Furthermore, the curvature is approximated linearly over the edge e_k. Note that the Crouzeix–Raviart basis function associated with the edge e_k is constant on this edge. Therefore, the approximation of the surface tension integral in (3.10) involves the integration of a linear function along a straight line and thus, the midpoint quadrature is exact. The final approximation of the surface integral is

$$\frac{1}{We}\int_{e_k}\varkappa\mathbf{n}\psi_k\,\mathrm{d}l \approx \frac{1}{We}\,\frac{\varkappa_{k,0}+\varkappa_{k,1}}{2}\,\frac{\mathbf{n}_{k,0}+\mathbf{n}_{k,1}}{\|\mathbf{n}_{k,0}+\mathbf{n}_{k,1}\|}\,l_k, \tag{4.1}$$

where $\varkappa_{k,0}$ and $\varkappa_{k,1}$ are the curvatures at the two endpoints of e_k and l_k is the length of the edge. This approximation guarantees that a static circular drop at zero gravity will remain at rest because the surface tension integral in (3.10) is equal to zero. Indeed, this surface tension integral, when tested with the basis function $\psi_k \in \mathbf{V}_{h,n+1}$ associated with edge e_k in $\Gamma_{h,n+1}^{n+1}$, becomes perpendicular to the edge and proportional to its length. This guarantees that the integral over the entire circular interface is equal to zero. It can be verified directly that this is generally not true when a higher order approximation for the interface is used. Higher smoothness of the approximation makes it in a way"incompatible" with the solenoidal projection; the piecewise constant pressure is not capable of neutralizing the more accurate surface tension and as a result parasitic velocities appear in $\mathbf{u}_{h,n+1}^{n+1}$.

5. Numerical validation

The proposed algorithm has been tested on a number of validation problems.

5.1. Lid-driven cavity flow

In order to validate the grid alignment algorithm we solve the well known lid-driven cavity problem on two different grids. The first one is a uniform triangular grid consisting of 40×40 squares subdivided into triangles and it is fixed during the period of the simulation $(0, T]$. A sample grid consisting of 5×5 nodes with the same structure is illustrated in Fig. 4. The second grid is produced from the same background grid by aligning it at each time step with an interface advected with the flow. Initially this interface is a circle of a radius $r = 0.2$ and it is centered at the geometric centre of the cavity. The Reynolds number is $Re = 40$ and the surface tension and gravity forces are equal to zero. At the final time $T = 2.0$, the interface is significantly deformed, but the results presented in Fig. 5 show that the velocity fields resulting from the two simulations are essentially the same (the initial and final positions of the interface are also shown in the right panel of this figure).

5.2. A convergence test

In order to study quantitatively the influence of the grid alignment on the accuracy of the scheme we repeat the previous test, however, now the initial and boundary conditions for the velocity are given by the following analytical solution

$$u = \sin(x)\sin(y + t)$$
$$v = \cos(x)\cos(y + t)$$
$$p = \cos(x)\sin(y + t)$$

with an appropriate source term added to the right-hand side of the governing equations. The case with a moving interface is solved using the three techniques discussed in Section 3.2–interpolation, ALE, and method of characteristics. Fig. 6 presents the convergence in space and time of $\tilde{\mathbf{u}}$ in L_2- and H_1-norms. It is clear that the presence of a moving boundary does not change the convergence of the algorithm. The method of characteristics is the least accurate, although it exhibits the same order of convergence as the other methods.

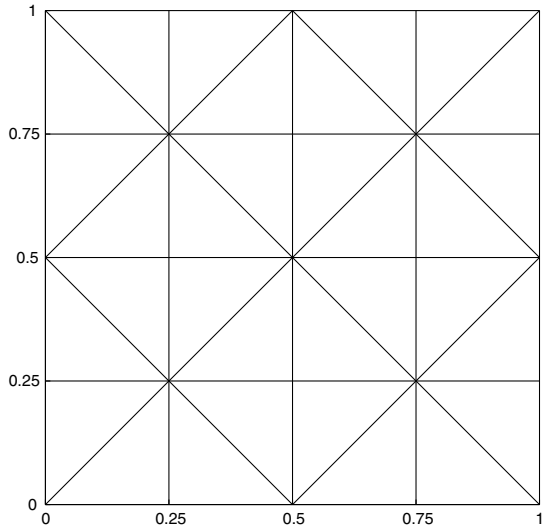


Fig. 4. Sample 4×4 grid.

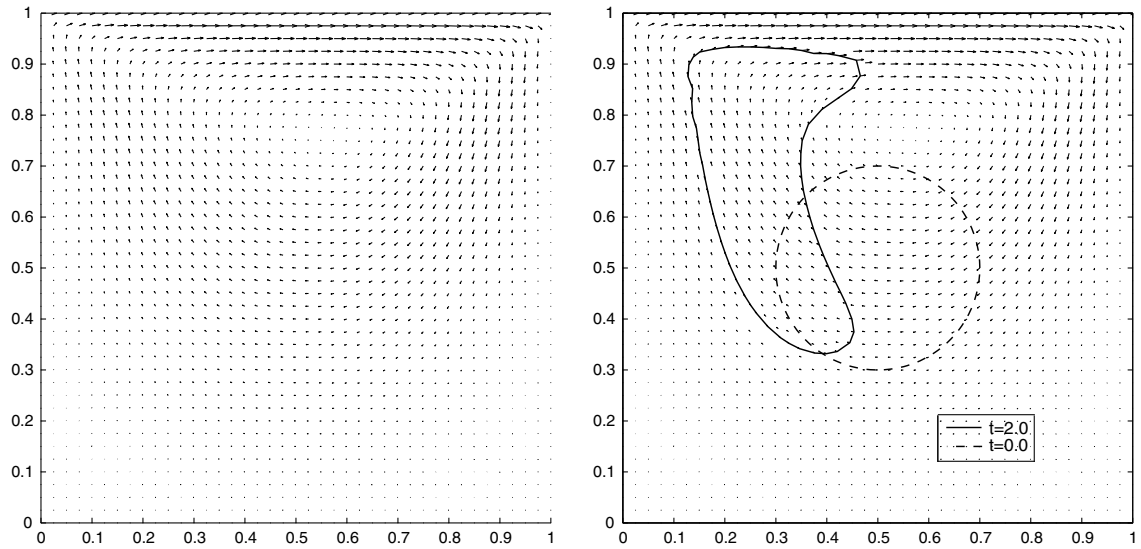


Fig. 5. Comparison of the results of the simulation of a lid-driven cavity flow with (right) and without (left) a free moving interface. $Re = 40$, $\Delta t = 0.005$, grid 40×40, no surface tension.

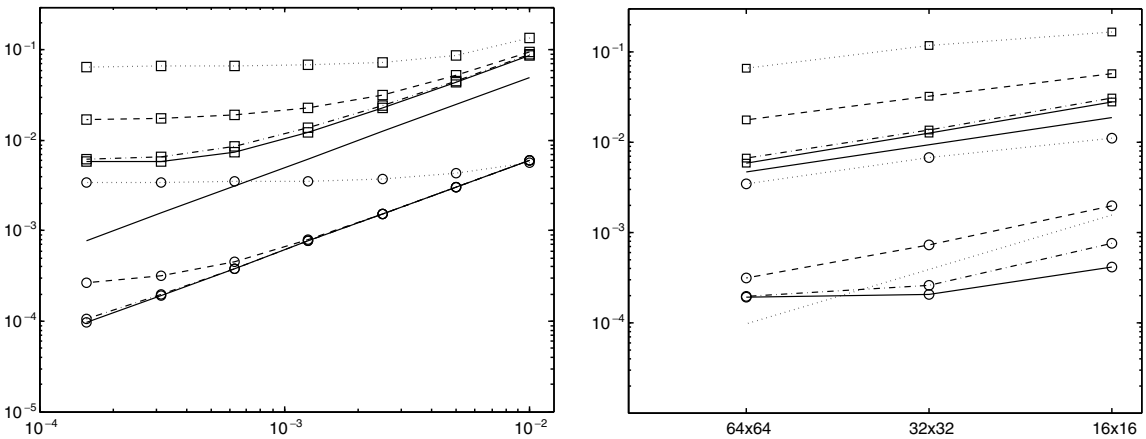


Fig. 6. Convergence of $\mathbf{u}$ in time (left, a uniform grid consisting of 64×64 rectangular cells subdivided into triangles) and space (right, $\Delta t = 0.0003125$). The test case is solved using ALE (dashed line), characteristics (dotted line), interpolation (dash-dotted line) and without moving interface (solid line). The error is measured in L^2 (circle marker) and H^1 (square marker) norms. Lines without markers represent slope 1 (solid line) and slope 2 (dotted line).

5.3. A circular drop

Consider a circular fluid particle immersed in another fluid and subject only to a non-zero surface tension (at zero gravity). If both fluids are at rest initially, the interface remains circular and the velocity field is equal to zero at all times. The pressure is also constant in each phase having a constant jump across the interface. This test case is quite challenging for many numerical techniques despite that the exact solution is so simple. The problem is that unless the curvature of the interface is approximated so that the contribution of the surface force to the right-hand side of (3.10) vanishes, the solution for the velocity would be non-zero and even if it is relatively small, provided that the surface tension is large enough, it can trigger oscillations on the interface. In addition, if the interface intersects some computational cells, the pressure cannot be properly approximated at the boundary (because it has a jump there) and as a result, after a sufficiently long integration in

time, a significant loss of mass may accumulate. The present method is exact (up to the machine accuracy) for this problem. It was tested in the square $[-1,1]\times[-1,1]$, for the following values of the dimensionless parameters: $\lambda_\rho = 1$, $\lambda_\mu = 1$, $Re = 10$, $We = 0.2, 1.0, 5.0$ on uniform grids of a grid size in each of the coordinate directions equal to 1/40 and 1/80.

The next test case considers the advection of a circular drop with a constant velocity field $\mathbf{u} = (1,0)$ at zero gravity. The boundary and initial conditions for the velocity are all set to (1,0) and the parameters are $\lambda_\rho = 1$, $\lambda_\mu = 1$, $Re = 10$ and $We = 0.2, 1.0, 5.0$. Theoretically, the shape and the velocity of the drop must not change as it moves. The domain is given by $[-1,1]\times[-0.5,0.5]$, and the radius of the circular particle is 0.25. The particle is initially located at $(-0.5,0)$. We use a grid of a grid size $1/24$ in each coordinate direction and a time step $\Delta t = 0.00125$. The final time is $T = 1.0$. Each of these cases is solved using the methods of interpolation, ALE and characteristics. All the methods produce the exact solution within the machine accuracy.

5.4. Relaxation of a drop of simple shape

This is a classical test case for free boundary methods where the initial shape of the particle slightly deviates from that of a disk and the gravity is set to zero. The problem is solved in the square $[0,1]\times[0,1]$, with parameters $\lambda_\rho \equiv 1$, $\lambda_\mu \equiv 1$, $Re = 10$, $We = 1.0$, on a uniform grid with a grid size $1/40$ in each direction, and a time step $\Delta t = 0.0025$. Initially the drop is elliptical with axes 5/16 and 1/5, i.e., it has the same area as a circle with radius 1/4. The drop becomes circular at time 0.3 (see Fig. 7) and remains so thereafter.

In order to test the stability properties of the three methods for data transfer between two subsequent grids this test case is run with a viscosity ratio $\lambda_\mu = 0.01$ on a uniform grid of 81×81 nodes. The results are summarized as follows. The interpolation of data on the grid $\mathcal{G}_{h,n}$ reaches an approximately circular static shape for $\Delta t = 0.00125$, however, spurious instabilities in the velocity around the interface are observed for some time after that. These oscillations are still presented if $\Delta t = 0.000833333$ is used, and become insignificant at $\Delta t = 0.000666667$ when the drop relaxes to a circle smoothly. The method of characteristics diverges (does not reach the circular steady shape) at $\Delta t = 0.002$ however it converges smoothly to a circle at $\Delta t = 0.001666667$. The ALE method diverges at $\Delta t = 0.00125$. At $\Delta t = 0.001111111$ it reaches a circular static shape, however, spurious instabilities around the interface can be observed for some time afterwards. It converges smoothly to a circle at $\Delta t = 0.001$. As expected, the method of characteristics has the most relaxed

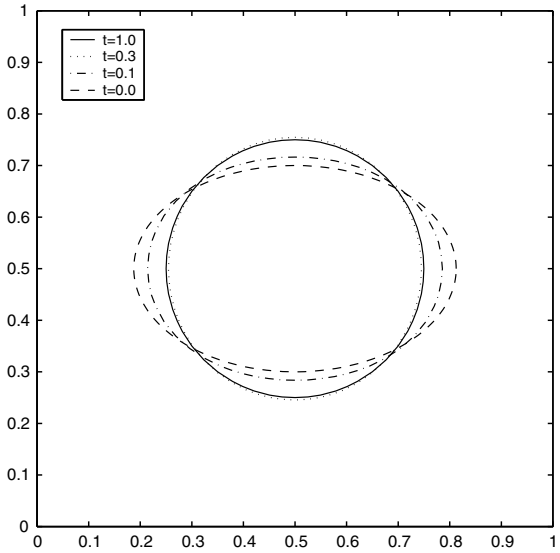


Fig. 7. Relaxation of elliptical particle. $We = 1.0$, $\Delta t = 0.0025$, grid 40×40.

stability restriction and therefore it is to be preferred at large viscosity ratios. If the viscosity ratio is close to one, however, all methods have a very similar behaviour.

5.5. Relaxation of drop of a complex shape

The next test case considers the relaxation of a drop of complex shape to a circle. The initial star-like curve given by

$$r = 0.2\sin(5\theta) - 0.5, \tag{5.1}$$

is proposed in [13]. The problem is solved in $[-1,1]\times[-1,1]$ on a uniform grid of 41x41 nodes with a time step $\Delta t = 0.005$ until time $T = 1.0$. The values of the parameters are the same as in the previous example. The drop reaches the static shape at approximately $T = 0.4$. In Fig. 8 we present the initial and final form of the drop together with the corresponding aligned grids, as well as the shape of the interface with the velocity field at times $T = 0.1, 0.25, 0.4$.

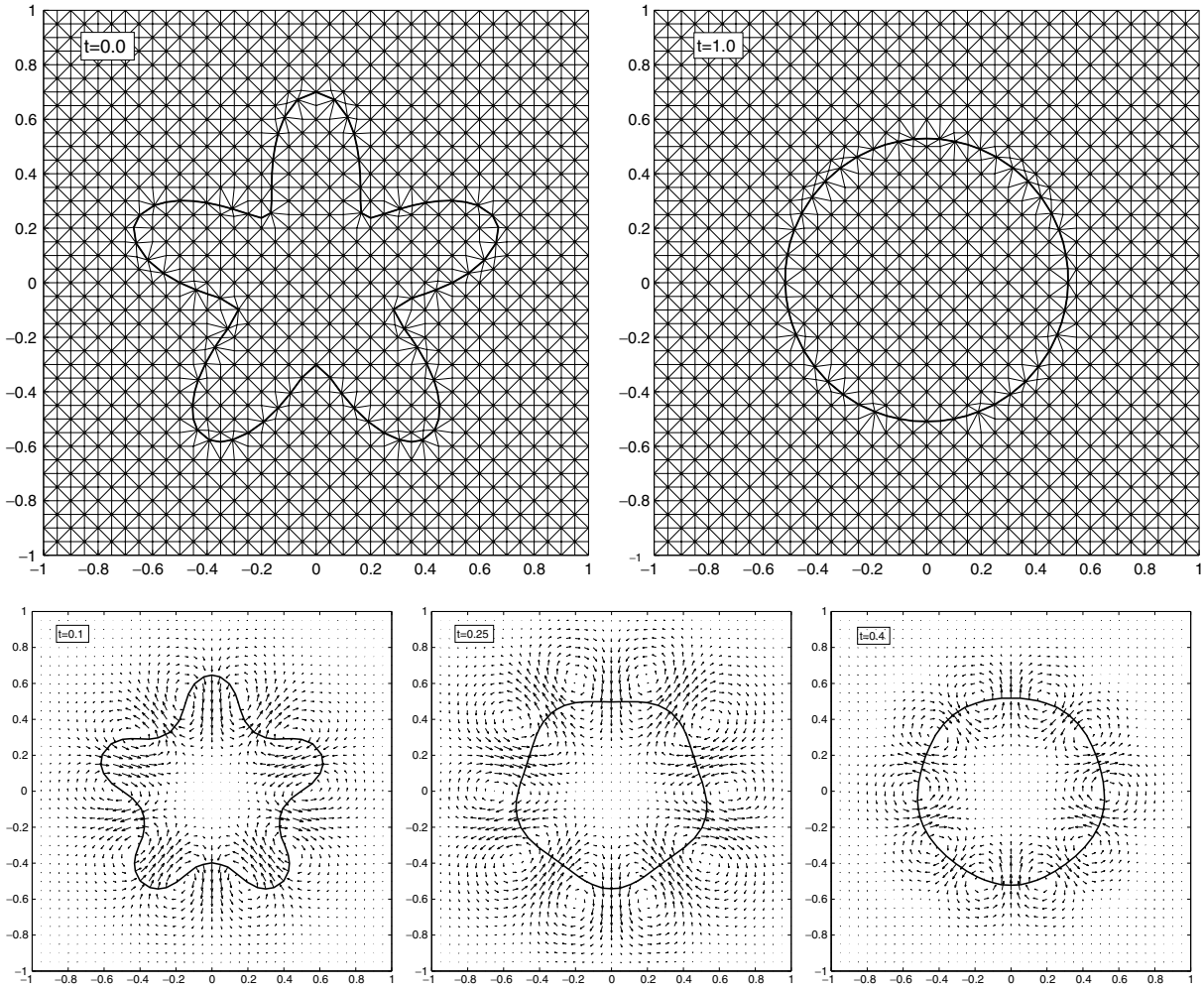
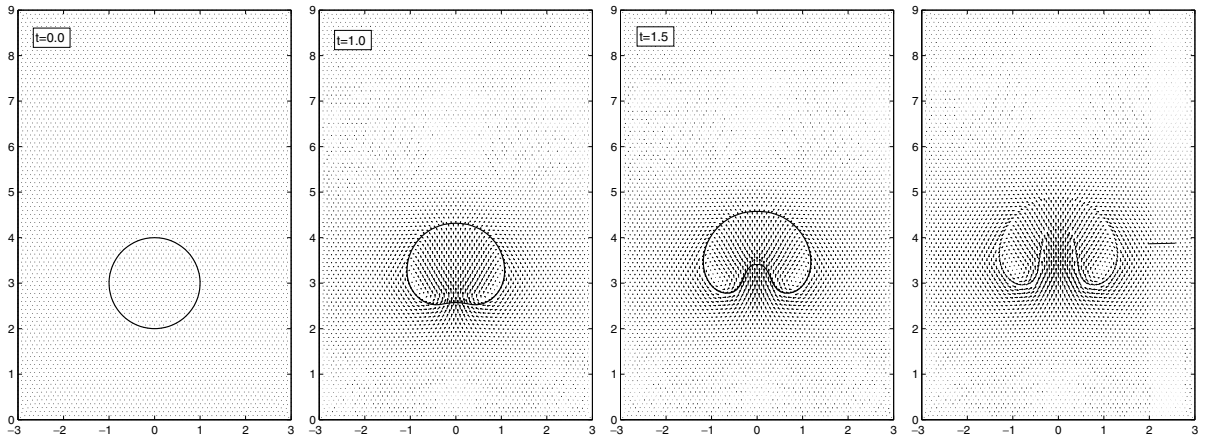


Fig. 8. Relaxation of star-shaped particle. $We = 1.0$, $\Delta t = 0.005$, uniform grid of 41×41 nodes.



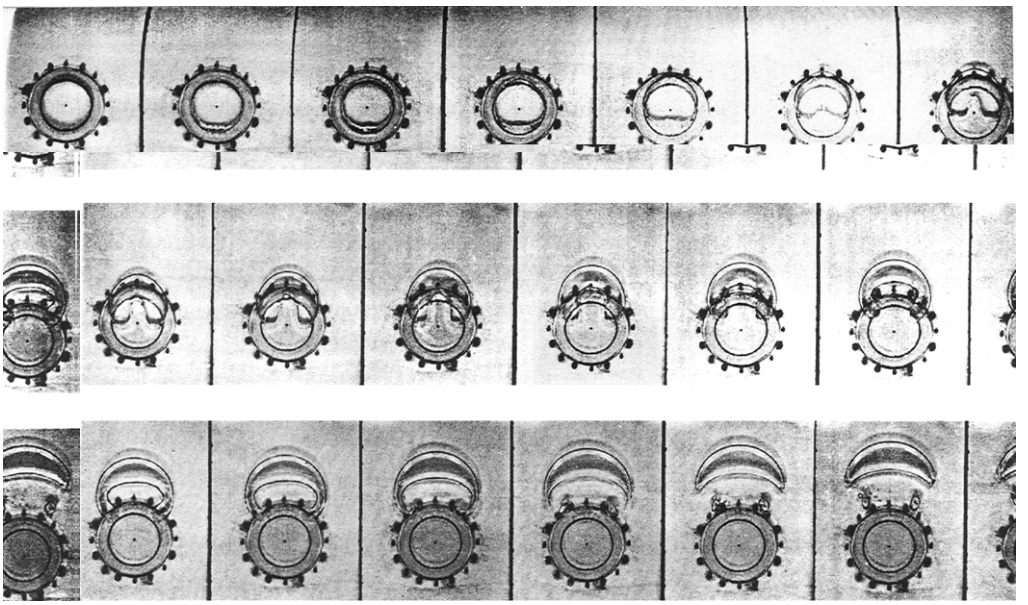


Fig. 10. An experimental result for a rising 2D bubble from [23]. The time between two subsequent frames is about 0.25.

5.6. Rising bubble

In the last test case we simulate the motion of a gas bubble immersed in a heavier liquid under the force of gravity. We simulate conditions similar to one of the cases in the experimental study of Walters and Davidson [23]. The parameters are $Re = 1000$, $Fr = 1$, $We = 200$, $\lambda_\rho = 1/816$ and $\lambda_\mu = 1/64$. The bubble has initially a circular shape with a radius equal to one, and is centered at $(0, 3)$. The computational domain is $[-3, 3] \times [0, 9]$, and the problem is discretized on a uniform grid with a grid size 0.1. The time step is $\Delta t = 0.000125$. The evolution of the bubble is presented in Fig. 9.

In the experiments of Walters and Davidson [23] a pinch off starts to develop at $t = 4.42$. Since the pinch off cannot be simulated with the present algorithm, we only compare the results before it occurs. The calculated shapes correlate well with the experimental results (see Fig. 10).

6. Conclusions

In conclusion let us summarize the advantages of the technique presented in this paper.

- The spatial approximation of the solution of the Navier–Stokes equations is accurate because the free boundaries are aligned with element edges on each grid.
- The grid connectivity does not change in time which makes the method easy to implement and parallelize.
- Strict mass conservation of the different phases is guaranteed.
- The approximation of the surface tension guarantees that the global contribution is equal to zero if the free boundary is circular, thus avoiding spurious currents.
- Three options for transfering data between subsequent grids are proposed. It seems that the Lagrangian and the ALE approaches yield better stability. However, this aspect of the method needs a further investigation.

Acknowledgements

The first and third authors would like to acknowledge the support, under a Discovery Grant, of the National Science and Engineering Research Council of Canada (NSERC). The work of the second author has been supported by CNRS and startup funds from Texas A&M.

References

[1] B. Bejanov, J.-L. Guermond, P.D. Minev, A locally div-free projection scheme for incompressible flows based on non-conforming elements, Int. J. Numer. Meth. Fluids 49 (2005) 549–568.
[2] T. Chen, P.D. Minev, K. Nandakumar, A projection scheme for incompressible miltiphase flow using adaptive Eulerian grid, Int. J. Numer. Meth. Fluids 45 (1) (2004) 1–19.
[3] M. Crouzeix, P.-A. Raviart, Conforming and nonconforming finite element methods for solving the stationnary Stokes equations, RAIRO Anal. Numér. 3 (1973) 33–75.
[4] M. Cruchaga, D. Celentano, T. Tezduyar, A moving Lagrangian interface technique for flow computations over fixed meshes, Comput. Meth. Appl. Mech. Eng. 191 (2001) 525–543.
[5] J. Donea, J-Ph. Ponthot, A. Huerta, A. Rodriquez-Ferran, Arbitrary Lagrangian–Eulerian methods, in: E. Stein, R. de Borst, T.J.R. Hughes (Eds.), Encyclopedia of Computational Mechanics, Fundamentals, vol. 1, John Wiley, Chichester, 2004 (Chapter 14).
[6] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, L. Wu, A simple package for front tracking, J. Comput. Phys. 213 (2) (2006) 613–628.
[7] M.M. Francois, S.J. Cummins, E.D. Dandy, D.B. Kothe, J.M. Sicilian, M.W. Williams, A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework, J. Comput. Phys. 213 (2006) 141–173.
[8] J. Glimm, J.W. Grove, X.L. Li, D.C. Tan, Robust computational algorithms for dynamic interface tracking in three dimensions, SIAM J Sci. Comput. 21 (6) (2000) 2240–2256.
[9] J. Glimm, L.I. Xiaolin, Y. Liu, X.U. Zhiliang, N. Zhao, Conservative front tracking with improved accuracy, SIAM J. Numer. Anal. 41 (5) (2003) 1926–1947.
[10] J.L. Guermond, J. Shen, Velocity-correction projection methods for incompressible flows, SIAM J. Numer. Anal. 41 (1) (2003) 112–134.
[11] R.J. LeVeque, Z. Li, Immersed interface methods for stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (3) (1997) 709–735.

[12] J. Li, Y. Renardy, M. Renardy, Numerical simulation of breakup of a viscous drop in simple shear flow through a volume-of-fluid method, Phys. Fluids. 12 (2) (2000) 269–282.
[13] Z. Li, M.C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comp. Phys. 171 (2001) 822–842.
[14] V. Maronnier, M. Picasso, J. Rappaz, Numerical simulation of three-dimensional free surface flows, Int. J. Numer. Meth. Fluids 42 (7) (2003) 697–716.
[15] S. Osher, R.P. Fedkiw, Level set methods an overview and some recent results, J. Comput. Phys. 169 (2001) 463–502.
[16] T. Chen, P.D. Minev, K. Nandakumar, A finite element technique for multifluid incompressible flow using Eulerian grids, J. Comput. Phys. 187 (2003) 255–273.
[17] J.A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annu. Rev. Fluid Mech. 35 (2003) 341–372.
[18] P. Shopov, P. Minev, I. Bazhlekov, Numerical method for unsteady viscous hydrodynamical problems with free boundaries, Int. J. Numer. Meth. Fluids 14 (6) (1992) 681–706.
[19] P.D. Minev, T. Chen, K. Nandakumar, A projection scheme for incompressible multiphase flow using adaptive Eulerian grids 3D validation, Int. J. Numer. Meth. Fluids 48 (2) (2005) 455–466.
[20] K. Takizawa, T. Yabe, Y. Tsugawa, T. Tezduyar, H. Mizoe, Computation of free-surface flows and fluid-object interaction with the CIP method based on adaptive meshless soroban grids, Comput. Mech. 40 (2007) 167–183.
[21] T.E. Tezduyar, Interface-tracking and interface-capturing techniques for finite element computation of moving boundaries and interfaces, Comput. Meth. Appl. Mech. Eng. 195 (23–24) (2006) 2983–3000.
[22] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, Y.-J. Jan, A front-tracking method for the computations of multiphase flow, J. Comput. Phys. 169 (2001) 708–759.
[23] J.L. Walters, J.F. Davidson, The initial motion of a gas bubble formed in an inviscid liquid, Part 1. The two-dimensional bubble, J. Fluid Mech. 12 (1962) 409–417.