Journal of Computational Physics 228 (2009) 3358–3367

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

# A velocity decomposition approach for moving interfaces in viscous fluids

J. Thomas Beale *, Anita T. Layton

*Department of Mathematics, Duke University, Box 90320, Durham, NC 27708-0320, USA*

ARTICLE INFO

*Article history:*
Received 3 September 2008
Received in revised form 16 January 2009
Accepted 21 January 2009
Available online 31 January 2009

*MSC:*
76D05
65N06
35R05
74F10

*Keywords:*
Navier–Stokes flow
Stokes flow
Boundary integral
Stiff equations
Fractional stepping
Immersed interface
Immersed boundary

ABSTRACT

We present a second-order accurate method for computing the coupled motion of a viscous fluid and an elastic material interface with zero thickness. The fluid flow is described by the Navier–Stokes equations, with a singular force due to the stretching of the moving interface. We decompose the velocity into a "Stokes" part and a "regular" part. The first part is determined by the Stokes equations and the singular interfacial force. The Stokes solution is obtained using the immersed interface method, which gives second-order accurate values by incorporating known jumps for the solution and its derivatives into a finite difference method. The regular part of the velocity is given by the Navier–Stokes equations with a body force resulting from the Stokes part. The regular velocity is obtained using a time-stepping method that combines the semi-Lagrangian method with the backward difference formula. Because the body force is continuous, jump conditions are not necessary. For problems with stiff boundary forces, the decomposition approach can be combined with fractional time-stepping, using a smaller time step to advance the interface quickly by Stokes flow, with the velocity computed using boundary integrals. The small time steps maintain numerical stability, while the overall solution is updated on a larger time step to reduce computational cost.



## 1. Introduction

We consider the coupled motion of a viscous fluid and an immersed boundary in a two-dimensional computational domain $\Omega$. The fluid flow is described by the Navier–Stokes equations. The immersed boundary $\Gamma$ is assumed to be a membrane consisting of elastic material, so that, when it is distorted from its rest state by stretching or relaxing, it exerts a restoring force on the fluid. We assume that $\Gamma$ is a simple closed curve, separating $\Omega$ into two subdomains, $\Omega^+$ (exterior) and $\Omega^-$ (interior), so that $\Omega = \Omega^+ \cup \Gamma \cup \Omega^-$. (Generalization to multiple closed immersed boundaries is straightforward.) We assume here that the fluid properties are the same in $\Omega^+$ and $\Omega^-$. The interfacial force causes discontinuities in the fluid pressure and velocity gradient at $\Gamma$. We design a numerical method for this prototype problem which is second-order accurate in space and time. The distinctive feature of this approach is a decomposition of the velocity and pressure into two parts, one part determined by the (steady) Stokes equations and the interfacial force on $\Gamma$, and a second, more regular part which can be calculated on a regular grid without special treatment near the interface. The decomposition allows the discontinuities in fluid variables to be accounted for accurately in a relatively simple manner. Furthermore, a smaller time step can be used to calculate the force and advance the immersed boundary while a large time step is used for the overall solution.

* Corresponding author.
*E-mail addresses:* beale@math.duke.edu (J.T. Beale), alayton@math.duke.edu (A.T. Layton).
*URLs:* http://math.duke.edu/faculty/beale (J.T. Beale), http://math.duke.edu/faculty/alayton (A.T. Layton).

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.01.023

The Navier–Stokes equations for the fluid motion are

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{F}, \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{2}$$

where $\mathbf{u} = (u, v)$ denotes the fluid velocity; $p$ is the pressure; $\mu$ is the fluid viscosity, assumed to be constant; and $\mathbf{F} = (F_1, F_2)$ is the interfacial force, supported entirely along $\Gamma$. The fluid density is set to 1. Biperiodic boundary conditions are assumed.

The force $\mathbf{F}$ is the elastic tension force that arises from the stretching of $\Gamma$. The stretching is naturally expressed in terms of a material coordinate $\alpha$ on $\Gamma$, chosen to be arclength in the rest state. At time $t$ the material point with label $\alpha$ has current position $\mathbf{X}(\alpha, t)$. We denote the arclength on $\Gamma$ at the current time $t$ by $s$ and write the force $\mathbf{F} = (F_1, F_2)$ as

$$F_i(\mathbf{x}, t) = \int_0^L f_i(s,t)\delta(\mathbf{x} - \mathbf{X}(\alpha(s), t))ds \quad i = 1, 2, \tag{3}$$

where $f_i$ is the force strength at point $s$ and $\delta$ is the two-dimensional delta function.

The tension force $\mathbf{f}$, as derived from force balance arguments, is given by

$$\mathbf{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)), \tag{4}$$

where we assume the tension $T(s,t)$ is given by

$$T(s,t) = T_0\left(\left|\frac{\partial \mathbf{X}}{\partial \alpha}\right| - 1\right), \tag{5}$$

when the material is stretched. The tension coefficient $T_0$ depends on the elastic properties of the interface and is assumed to be a constant in this model. The unit tangent vector $\boldsymbol{\tau}(s,t)$ to $\Gamma$ is

$$\boldsymbol{\tau}(s,t) = \frac{\partial \mathbf{X}}{\partial s} = \frac{\partial \mathbf{X}/\partial\alpha}{|\partial \mathbf{X}/\partial\alpha|}. \tag{6}$$

Thus the force density can be computed directly from the location $\mathbf{X}(\alpha, t)$ of the boundary $\Gamma$. Note that in the relaxed state $|\partial\mathbf{X}/\partial\alpha| = 1$, and the tension vanishes.

Owing to the presence of a singular force $\mathbf{F}$, the solution of (1) and (2) is not smooth across $\Gamma$. Its effect can be expressed in terms of jump conditions in $p$ and $\mathbf{u}$. For any quantity $q$ with jump discontinuity at the boundary, the jump is

$$[q(\mathbf{X}, t)] = \lim_{\epsilon\to 0^+} q(\mathbf{X} + \epsilon\mathbf{n}, t) - \lim_{\epsilon\to 0^+} q(\mathbf{X} - \epsilon\mathbf{n}, t), \tag{7}$$

where $\mathbf{n}$ denotes the vector normal to $\Gamma$.

The jump conditions for pressure (see [16,19,25]) are

$$[p] = \mathbf{f}\cdot\mathbf{n}, \quad \left[\frac{\partial p}{\partial n}\right] = \frac{\partial}{\partial s}(\mathbf{f}\cdot\boldsymbol{\tau}). \tag{8}$$

The velocity $\mathbf{u} \equiv (u, v)$ is continuous across $\Gamma$; the boundary moves with the viscous fluid. However, the normal derivatives of the velocity components have jump discontinuities:

$$[\mathbf{u}] = 0, \quad \mu\left[\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right] = -(\mathbf{f}\cdot\boldsymbol{\tau})\boldsymbol{\tau}. \tag{9}$$

Unless special care is taken, these discontinuities tend to introduce substantial inaccuracy into the computed solution obtained by means of a standard finite difference method.

The immersed boundary method of Peskin et al. [25,29,24] was designed for this prototype problem and its generalizations for applications in biomechanics; see the review [24] and references therein. The interface is represented by markers as above, and a carefully designed smooth version of a delta function is used to impart the force due to each marker point to the grid points for the fluid. (Note that our notation above is slightly different from Peskin's.) This method can be used for a wide variety of problems, more realistic than that considered here. For an interface of zero thickness, the immersed boundary method seems to be limited to first-order accuracy. However, second-order versions have been designed for the case with a layer of positive thickness rather than an interface ([10,22]). Much work has been done related to the accuracy of the method with interfaces, as well as efforts to deal with stiffness, including [25,29,21,5,9,12,23].

One approach for achieving better accuracy is the immersed interface method [16,18] (or similarly, Mayo's method [20]), which first expresses the jumps in the solution and its derivatives in terms of the boundary force and its derivatives, and then incorporates these jumps into a finite difference scheme as correction terms [16,19,32]. This method was first applied to the Stokes equations [16], the simplified model at zero Reynolds number in which acceleration and advection are neglected. The Stokes equations, (11) and (12) below, form an elliptic system at each time. The method can also be applied to the full Navier–Stokes equations (1) and (2). This was first done in [19,15]. In principle this method can be quite accurate, but it becomes rather involved because of the large number of cases and corrections needed. Thorough development of this

approach with elastic boundaries has been made in [14,32], with extensive applications to flow past fixed boundaries, and in [27] to moving boundaries with jump in viscosity. Apparently in [32] second-order accuracy was obtained in space but not in time; in [27] near second-order accuracy was verified for a test problem with a deforming elastic interface.

The motion of an interface in Stokes flow is easier to represent accurately than for the case of Navier–Stokes. If we use the immersed interface method, the necessary correction terms are significantly simpler in the Stokes case [16]. Also, the solution of the Stokes equations can be expressed in boundary integrals [26]. Based on these observations, we propose an alternative – a method of velocity decomposition – for computing the solution of problem (1)–(5).

The method of velocity decomposition takes advantage of the fact that the jump conditions in the fluid variables for the problem with Navier–Stokes flow are the same as those for Stokes flow. To compute the solution of (1)–(5), we express the fluid velocity and pressure as the sum of a *Stokes* part and a *regular* part, denoted by the subscripts '*s*' and '*r*,' respectively:

$$\mathbf{u} = \mathbf{u}_s + \mathbf{u}_r, \quad p = p_s + p_r. \tag{10}$$

The Stokes part of the solution is determined by the Stokes equations, including the boundary force:

$$\nabla p_s = \mu \nabla^2 \mathbf{u}_s + \mathbf{F}, \tag{11}$$

$$\nabla \cdot \mathbf{u}_s = 0. \tag{12}$$

The jump conditions for $\mathbf{u}_s, p_s$ are the same as those for $\mathbf{u}, p$ in (8) and (9). This follows from the fact that $\mathbf{u}$ is continuous at $\Gamma$, and therefore its material, or total, derivative is continuous as well. Taking the difference of (1) and (11), one obtains the equation for the regular part of the solution

$$\frac{\partial \mathbf{u}_r}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}_r = -\nabla p_r + \mu \nabla^2 \mathbf{u}_r + \mathbf{F}_b, \tag{13}$$

$$\nabla \cdot \mathbf{u}_r = 0, \tag{14}$$

where $\mathbf{F}_b$ is a body force given by the material derivative of the Stokes velocity:

$$\mathbf{F}_b = -\frac{\partial \mathbf{u}_s}{\partial t} - \mathbf{u} \cdot \nabla \mathbf{u}_s. \tag{15}$$

Notice that the transport of both $\mathbf{u}_r$ on the left side of (13) and $\mathbf{u}_s$ in $\mathbf{F}_b$ are with the full velocity $\mathbf{u}$. Unlike $\mathbf{F}$, the body force $\mathbf{F}_b$ is not singularly supported on $\Gamma$. It is a continuous function on $\Omega$, since $\mathbf{u}_s$, and thus its material derivative, are continuous across $\Gamma$. However, the gradient of $\mathbf{F}_b$ has a jump discontinuity across $\Gamma$. Because the jump conditions for $\mathbf{u}_s, p_s$ are the same as those for $\mathbf{u}, p$, the corresponding jumps for $\mathbf{u}_r, p_r$ are zero. This fact and the continuity of $\mathbf{F}_b$ suggest that we can solve for $\mathbf{u}_r, p_r$ on a regular grid accurately without jump terms at the interface.

We solve the Stokes equations (11) and (12) using the immersed interface method as in [16]. For the regular part, it appears that if we were to solve (13) and (14) by means of the immersed interface method, we should not have to impose corrections for jumps; see [19]. However, the computed solution will still be sensitive to the choice of numerical method. We wish to avoid discretizing the advection terms since $\nabla \mathbf{u}$ is discontinuous at $\Gamma$. Moreover, the mild nonsmoothness of $\mathbf{u}_r$ at $\Gamma$ might lead to inaccuracy unless we use a time-stepping method which smooths the high wavenumbers. For these reasons we solve (13) and (14) with the semi-Lagrangian or Courant, Isaacson, Rees (CIR) method (e.g., [31,4]) and discretize in time with a backward difference formula. Values of the unknowns at the current time are found at regular grid points using Lagrangian treatment of transport terms. That is, the material derivative is replaced by a difference quotient formed with earlier values found at a location reached by traveling backward in time; the earlier value at that location must be interpolated in space from grid values. The complete method is relatively simple but maintains full second-order accuracy.

If the boundary force is sufficiently stiff (i.e., the tension coefficient is sufficiently large), an explicit method requires smaller time steps to be stable. The decomposition approach provides the flexibility to use small time steps to advance the boundary while using a larger step for the complete motion. The small time step is used for the Stokes part only, since the interfacial force is associated with this part. Moreover, the velocity needs be computed on the small time step only at boundary locations. We compute the free-space Stokes velocity from the boundary integral representation on the small steps with corrections from the large steps.

In Section 2, we describe the numerical method in more detail. Section 3 presents examples which demonstrate the second-order convergence in space-time and illustrate the effectiveness of the fractional time steps. Some issues related to the accuracy and possible extensions of this approach are discussed in Section 4.

## 2. Numerical method

With time step $\Delta t > 0$ chosen, let $t_n \equiv n\Delta t$ be the $n$th time level, for $n = 0, 1, \ldots$. For any time-dependent quantity $q$, we write $q^n$ for $q(t_n)$. We use rectangular grids with grid interval $h_x$ and $h_y$ along the $x$- and $y$-axis. For notational simplicity, we assume $h_x = h_y \equiv h$. We compute values of fluid quantities at grid points $(ih, jh)$ where $i, j = 0, 1, 2, \ldots, N$ with periodicity imposed in $x$ and $y$. The position of the immersed boundary at time $t$ is represented by a set of boundary markers $\mathbf{X}_k(t) \equiv (X_k(t), Y_k(t))$, for $k = 0, 1, 2, \ldots, N_k$, where $\mathbf{X}_0(t) = \mathbf{X}_{N_k}(t)$, since the boundary is assumed to be a simple closed curve. The $k$th boundary marker approximates $\mathbf{X}(\alpha_k, t)$, where $\alpha_k = kL_0/N_k$, and $L_0$ is the length of $\Gamma$ in the unstretched state; that is,

the boundary markers are chosen so that they are equally spaced in the unstretched or relaxed state. We discuss the partial steps for updating the velocity and boundary markers in succession.

### 2.1. Computing the stokes solution

In the method of velocity decomposition, we first compute the Stokes pressure $p_s$ and velocity $\mathbf{u}_s \equiv (u_s, v_s)$ by solving (11) and (12). We follow the approach of LeVeque and Li [16] and reduce (11) and (12) to a sequence of Poisson problems, one for each unknown $p_s, u_s, v_s$; the resulting Poisson problems are solved together with the jump conditions for the unknowns. We incorporate the jumps in the solution and its derivatives using Mayo's technique [20], in which the jumps are found along the coordinate lines, an approach slightly different from that in [16]. Nonetheless, we use the term immersed interface method. An analysis of the accuracy of the method, with application to the Stokes equations as in [16], can be found in [2].

As noted before the Stokes pressure $p_s$ and velocity $\mathbf{u}_s$ have the same jump conditions as (8) and (9) for $p$ and $\mathbf{u}$. To compute the fluid motion given by (11) and (12), as in [16], we first solve

$$\nabla^2 p_s = 0 \quad \text{in } \Omega^+ \cup \Omega^- \tag{16}$$

for $p_s$ along with the jump conditions (8). Once $p_s$ is computed, $\mathbf{u_s} = (u_s, v_s)$ can be obtained by solving (11), with $\mathbf{F}$ set to 0 in $\Omega^+ \cup \Omega^-$, together with (9). Standard second-order finite difference operators are used, and the solutions are computed with the FFT using the periodicity condition.

### 2.2. Computing the regular solution

Once $\mathbf{u}_s$ is known, the body force $\mathbf{F}_b$, which is given by the material, or total, derivative of $\mathbf{u}_s$, can be found at the current time. As previously noted, since both $\mathbf{u}$ and $\mathbf{u}_s$ are continuous across the interface, their material derivatives are also continuous, and we seek to take advantage of this fact. We advance $\mathbf{u}_r$ on a square grid, with semi-Lagrangian treatment of the material derivatives. Such an approach has long been used in fluid problems (e.g., see [8], Chapter VI) and is now common in meteorology [6]; a recent use in a setting similar to ours is in [4]. A recent thorough treatment is in [31].

In the semi-Lagrangian discretization, the advection terms are incorporated into material derivatives, and (13) becomes

$$\frac{d\mathbf{u}_r}{dt} = -\nabla p_r + \mu \nabla^2 \mathbf{u}_r + \mathbf{F}_b, \tag{17}$$

where

$$\mathbf{F}_b = -\frac{d\mathbf{u}_s}{dt}. \tag{18}$$

To discretize (17) along trajectories, we use a second-order backward difference formula (BDF) as in [8,31,4], since it avoids complications inherent in methods such as Crank–Nicolson with backward characteristics (see [31]). The resulting discretized equation is

$$\frac{3\mathbf{u}_r^{n+1} - 4\tilde{\mathbf{u}}_r^{n} + \tilde{\mathbf{u}}_r^{n-1}}{2\Delta t} + \nabla p_r^n = \mu \nabla^2 \mathbf{u}_r^{n+1} + \mathbf{F}_b^{n+1}, \tag{19}$$

where $\tilde{\mathbf{u}}_r^n$ and $\tilde{\mathbf{u}}_r^{n-1}$ are the fluid velocities at upstream positions $\mathbf{x}^n$ and $\mathbf{x}^{n-1}$, respectively, i.e.,

$$\tilde{\mathbf{u}}_r^n = \mathbf{u}_r(\mathbf{x}^n, t_n), \tag{20}$$

$$\tilde{\mathbf{u}}_r^{n-1} = \mathbf{u}_r(\mathbf{x}^{n-1}, t_{n-1}). \tag{21}$$

The material derivative of $\mathbf{u}_s$ in $\mathbf{F}_b$ on the right side of (19) is discretized in the same way as for $\mathbf{u}_r$ on the left. For the first step, i.e., for $n = 0$, backward Euler is used to discretize (17).

To evaluate $\tilde{\mathbf{u}}_r^n$ and $\tilde{\mathbf{u}}_r^{n-1}$, we first estimate $\mathbf{x}^n$ and $\mathbf{x}^{n-1}$, given by the initial value problem

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{u}(\mathbf{x}(t), t), \quad \mathbf{x}(t_{n+1}) = \mathbf{x}_0, \tag{22}$$

where the initial point $\mathbf{x}_0$ is a regular mesh point $(ih, jh)$. We estimate the upstream particle positions $\mathbf{x}^n$ and $\mathbf{x}^{n-1}$ by integrating (22) backward in time over the interval $[t_{n+1}, t_n]$ and $[t_{n+1}, t_{n-1}]$, respectively, using the midpoint method, according to the formulas

$$\mathbf{x}^* = \mathbf{x}_0 - \frac{\Delta t}{2}\mathbf{u}\left(\mathbf{x}_0 - \frac{\Delta t}{2}\mathbf{u}^{n+\frac{1}{2}}, t_{n+\frac{1}{2}}\right), \quad \mathbf{x}^n = \mathbf{x}_0 - \Delta t\,\mathbf{u}\left(\mathbf{x}^*, t_{n+\frac{1}{2}}\right), \tag{23}$$

$$\mathbf{x}^* = \mathbf{x}_0 - \Delta t\,\mathbf{u}(\mathbf{x}_0 - \Delta t\,\mathbf{u}^n, t_n), \quad \mathbf{x}^{n-1} = \mathbf{x}_0 - 2\Delta t\,\mathbf{u}(\mathbf{x}^*, t_n) \tag{24}$$

with interpolation as explained below.

The fluid velocities $\mathbf{u}$ and $\mathbf{u}_r$ are known at time levels $t_n$ and $t_{n-1}$, but only at grid points, which likely do not coincide with the upstream positions $\mathbf{x}^n$ and $\mathbf{x}^{n-1}$ that are needed in (19), (23), and (24). Thus, cubic Lagrange interpolation in the spatial

variable is used to estimate $\mathbf{u}$ and $\mathbf{u}_r$ at other locations. Cubic Lagrange interpolation, a fourth-order method, is used in conjunction with a second-order temporal discretization because of the expected order reduction in a semi-Lagrangian discretization when the spatial and temporal meshes are scaled alike, i.e., when $\Delta t = \mathcal{O}(h)$ (see [31,7]), and because the damping and phase-shift effects introduced by cubic interpolation are usually less severe than from linear or quadratic interpolation.

In (24), $\mathbf{u}^n$ is evaluated at $\mathbf{x}_0$ while at other locations $\mathbf{u}(\cdot, t_n)$ is found by cubic Lagrange interpolation. In (23), $\mathbf{u}^{n+\frac{1}{2}}$, the value at location $\mathbf{x}_0$, is approximated using the time extrapolation $\frac{3}{2}\mathbf{u}^n - \frac{1}{2}\mathbf{u}^{n-1}$, and for $\mathbf{u}\left(\cdot, t_{n+\frac{1}{2}}\right)$ at other locations, the same extrapolation is used in time, as well as spatial interpolation. Once $\mathbf{x}^n$ and $\mathbf{x}^{n-1}$ are known, they are used to find $\tilde{\mathbf{u}}_r^n$ and $\tilde{\mathbf{u}}_r^{n-1}$ in (19), and the forcing term $\mathbf{F}_b^{n+1}$ is treated analogously.

We use the second-order projection method to compute the solution of (19) and (14). We first solve for the intermediate velocity $\mathbf{u}_r^*$ from

$$\frac{3\mathbf{u}_r^* - 4\tilde{\mathbf{u}}_r^n + \tilde{\mathbf{u}}_r^{n-1}}{2\Delta t} + \nabla p_r^n = \mu\nabla^2\mathbf{u}_r^* + \mathbf{F}_b^{n+1}. \tag{25}$$

Specifically, fast Fourier transforms are used to compute the solution of

$$\left(\frac{3}{2\Delta t} - \mu\nabla^2\right)\mathbf{u}_r^* = \mathbf{F}_b^{n+1} - \frac{1}{2\Delta t}\left(-4\tilde{\mathbf{u}}_r^n + \tilde{\mathbf{u}}_r^{n-1}\right) - \nabla p_r^n \tag{26}$$

again using standard second-order finite difference operators. Next $\mathbf{u}_r^{n+1}$ is found by approximately projecting $\mathbf{u}_r^*$ onto the subspace of divergence-free vector fields,

$$\mathbf{u}_r^{n+1} = \mathbb{P}\mathbf{u}_r^*, \quad \mathbb{P}\mathbf{v} = \mathbf{v} - \nabla(\nabla^2)^{-1}\nabla\cdot\mathbf{v}. \tag{27}$$

This is achieved by defining $\phi$ as

$$\mathbf{u}_r^{n+1} = \mathbf{u}_r^* - (\Delta t)\nabla\phi, \tag{28}$$

and solving the Poisson equation

$$(\Delta t)\nabla^2\phi = \nabla\cdot\mathbf{u}_r^* \tag{29}$$

using difference operators in the Fourier transform. Then, in grid space, the pressure is updated according to

$$\nabla p_r^{n+1} = \nabla p_r^n + \frac{3}{2}\nabla\phi - \mu\Delta t\nabla^3\phi \tag{30}$$

so that finally we obtain the solution of

$$\frac{3\mathbf{u}_r^{n+1} - 4\tilde{\mathbf{u}}_r^n + \tilde{\mathbf{u}}_r^{n-1}}{2\Delta t} + \nabla p_r^{n+1} = \mu\nabla^2\mathbf{u}_r^{n+1} + \mathbf{F}_b^{n+1}. \tag{31}$$

Note that $\mathbb{P}$ is not an exact projection because $\nabla^2 \neq \nabla\cdot\nabla$ with the usual second-order difference operators. See, e.g., [10] for further discussion of the approximate projection method.

### 2.3. Computing boundary motion

Besides simplifying the treatment of jump conditions, the method of velocity decomposition also offers an efficient approach for advancing the boundary when the boundary force is stiff. When an explicit time-stepping method is used to advance a stiff boundary, the time step must be sufficiently small to maintain numerical stability, leading to high computational costs. This difficulty may be overcome by means of an implicit or semi-implicit scheme [21,29,16,23,12]. By computing boundary forces implicitly, i.e., from the boundary configuration at the end of the time step, one may relax the time step restriction imposed by stiffness. However, this approach results in a non-local and nonlinear problem for the boundary forces, which involves the fluid–mediated interaction of each boundary point with every other boundary point and requires the solution of a dense system of equations. Motivated by these numerical challenges, we propose a fractional time-stepping algorithm – the boundary is advanced using a smaller time step to maintain numerical stability, and the overall solution is updated using a larger time step to reduce computational cost.

Suppose we take $N_m$ (small) forcing substeps $\Delta t_m$ per (large) advection time step $\Delta t$; thus, $\Delta t = N_m\Delta t_m$. Let $t_m$ and $t_n$ denote the forcing and advection time-levels, respectively, where $t_m = t_n + m\Delta t_m = t_n + m\Delta t/N_m$, for $m = 1, 2, \ldots, N_m$. Because the proposed method treats the boundary force explicitly, the size of $\Delta t_m$ may be severely restricted to maintain numerical stability. Thus, for the method to be efficient, the computational cost in advancing the boundary by one $\Delta t_m$ must be sufficiently low. To this end, we make the following observations:

(1) The decomposition places the stiff boundary force in the Stokes equations (11) and (12). Thus, while the Stokes velocity must be advanced using a sufficiently small time step, the regular solution may be computed using a larger time step without significant loss of accuracy.

(2) To update the boundary forces, one must update the boundary position; to update the boundary position under Stokes flow, one needs to update the fluid velocity at points on the boundary – but nowhere else.
(3) The free-space Stokes velocity can be computed using a boundary integral, and this is routine for boundary locations.

With the above observations, we propose a fractional time-stepping method that updates *only* the Stokes velocity at the boundary during each forcing step $\Delta t_m$ using boundary integrals; the full velocity is then computed less frequently in the entire domain at each advection time step $\Delta t$. Let $\mathbf{u}_\Gamma$ denote the boundary velocity, i.e., $\mathbf{u}_\Gamma \equiv \mathbf{u}(\mathbf{X}(t),t)$. It is found on the small time steps in a manner explained below. To advance the boundary configuration from $t_m$ to $t_{m+1}$, we update the boundary markers as follows:

$$\mathbf{X}^{m+1} = \mathbf{X}^m + \Delta t_m \left( \frac{3}{2}\mathbf{u}_\Gamma^m - \frac{1}{2}\mathbf{u}_\Gamma^{m-1} \right). \tag{32}$$

When $m = 0$ and $n > 0$, $\mathbf{u}_\Gamma^{m-1}$ is set to $\mathbf{u}_\Gamma^{N_m-1}$ from the previous time interval $[t_{n-1}, t_n]$. When $m = 0$ and $n = 0$, forward Euler is used to advance $\mathbf{X}$. For the special case $N_m = 1$, the boundary markers are advanced using velocity values at the previous two advection time-levels:

$$\mathbf{X}^{n+1} = \mathbf{X}^n + \Delta t \left( \frac{3}{2}\mathbf{u}_\Gamma^n - \frac{1}{2}\mathbf{u}_\Gamma^{n-1} \right). \tag{33}$$

As in the case of the overall solution $\mathbf{u}$, the boundary velocity $\mathbf{u}_\Gamma$ can be expressed as the sum of a Stokes part and a regular part:

$$\mathbf{u}_\Gamma^m = \mathbf{u}_{s\Gamma}^m + \mathbf{u}_{r\Gamma}^m. \tag{34}$$

The Stokes velocity $\mathbf{u}_{s\Gamma}^m$, which satisfies (11) and (12), containing the stiff boundary force, is updated using $\mathbf{f}^m$ at every $t_m$, but only at the boundary $\Gamma$. To update $\mathbf{u}_{s\Gamma}^m$, we use the representation of the Stokes velocity as a layer potential:

$$\mathbf{u}_{s\Gamma}^m = \int_\Gamma V(\mathbf{X}^m - \mathbf{y}) \mathbf{f}^m(\mathbf{y}) ds(\mathbf{y}), \tag{35}$$

where the kernel $V$ depends on the space dimension and the boundary conditions. For free-space boundary conditions in two dimensions,

$$V_{\mathrm{FS}}(\mathbf{x}) = \frac{1}{4\pi\mu} \begin{bmatrix} -\log|\mathbf{x}| + \frac{x_1^2}{|\mathbf{x}|^2} & \frac{x_1 x_2}{|\mathbf{x}|^2} \\ \frac{x_1 x_2}{|\mathbf{x}|^2} & -\log|\mathbf{x}| + \frac{x_2^2}{|\mathbf{x}|^2} \end{bmatrix}, \tag{36}$$

where $\mathbf{x} \equiv (x_1, x_2)$. (See [26].)

In this study, biperiodic boundary conditions are used. Thus, the free-space kernel $V_{\mathrm{FS}}(x)$ given in (36) is, in principle, not applicable; instead, we need the analogous function for biperiodic boundary conditions. However, because the biperiodic kernel $V_{\mathrm{P}}(x)$ cannot be written explicitly, we compute the free-space velocity $\mathbf{u}_{s\Gamma,\mathrm{FS}}^m$ using $V(x) = V_{\mathrm{FS}}(x)$ in (36), and then correct for the discrepancy in boundary conditions. That is, we compute a correction term $\mathbf{u}_{s\Gamma,\mathrm{BC}}^m$, which is defined as

$$\mathbf{u}_{s\Gamma,\mathrm{BC}}^m = \int_\Gamma V_{\mathrm{P}}(\mathbf{X}^m - \mathbf{y}) \mathbf{f}^m(\mathbf{y}) ds(\mathbf{y}) - \int_\Gamma V_{\mathrm{FS}}(\mathbf{X}^m - \mathbf{y}) \mathbf{f}^m(\mathbf{y}) ds(\mathbf{y}). \tag{37}$$

Now since $V_{\mathrm{P}}(x)$ is not known, $\mathbf{u}_{s\Gamma,\mathrm{BC}}^m$ cannot be computed directly using (37). Instead, the Stokes problem (11) and (12) is solved twice for each large time step on the grid points of $\Omega$ by means of the immersed interface method, once with the biperiodic boundary conditions, and separately with the free space boundary conditions. (In the free-space case we need values for the velocity and pressure on the boundary of the computational domain $\Omega$. We find these from their integral representations ([26,2]) and then use the immersed interface method as before but with Dirichlet boundary conditions.) The difference between the two sets of velocity values yields $\mathbf{u}_{\mathrm{sBC}}$ on grid points.

Assuming that $\Gamma$ is sufficiently distant from the computational domain boundary $\partial\Omega$, the boundary condition correction $\mathbf{u}_{\mathrm{sBC}}^m$ is likely to be smoothly varying, compared to the boundary velocity $\mathbf{u}_{s\Gamma}^m$. Thus, to reduce computational cost, $\mathbf{u}_{\mathrm{sBC}}$ is updated only at the advection time-level $t_n$. At each $\Delta t_m$, $\mathbf{u}_{\mathrm{sBC}}^m$ is first obtained on grid points by means of temporal extrapolations using values at $t_n$ and $t_{n-1}$. Then $\mathbf{u}_{s\Gamma,\mathrm{BC}}^m$ is obtained from $\mathbf{u}_{\mathrm{sBC}}^m$ values at grid points by means of spatial interpolations. Because the jump conditions of $\mathbf{u}_{\mathrm{sFS}}$ and $\mathbf{u}_{\mathrm{sP}}$ are the same, those jumps cancel when one takes the difference of the two solutions to generate $\mathbf{u}_{\mathrm{sBC}}$. Thus, no correction terms are needed when interpolating $\mathbf{u}_{s\Gamma,\mathrm{BC}}^m$. The Stokes boundary velocity is then given by

$$\mathbf{u}_{s\Gamma}^m = \mathbf{u}_{s\Gamma,\mathrm{FS}}^m + \mathbf{u}_{s\Gamma,\mathrm{BC}}^m. \tag{38}$$

The regular velocity $\mathbf{u}_r$ at the boundary is handled during the small time steps in a manner similar to $\mathbf{u}_{\mathrm{sBC}}$, since its evolution Eq. (13) does not contain the stiff force $\mathbf{F}$. That is, it is computed only at the advection time-level $t_n$. For each $\Delta t_m$, $\mathbf{u}_{r\Gamma}^m$ is computed by first extrapolating to time $t_m$ from grid-point values of $\mathbf{u}_r$ at $t_n$ and $t_{n-1}$, and then spatially interpolating those grid-point values to the boundary markers. The final boundary velocity is given by (34).

### 2.4. The algorithm

At $t_n$, approximations for $\mathbf{u}_s^n, \mathbf{u}_r^n, \mathbf{u}_{\mathrm{sBC}}^n, \nabla p_r^n, \mathbf{X}^n$, and $\mathbf{f}^n$ are known. The algorithm for advancing the solution from $t_n$ to $t_{n+1}$ is summarized below:

(1) For $m = 0, 1, \ldots, N_m - 1$,
   (a) Compute free-space boundary velocity $\mathbf{u}_{s\Gamma,\mathrm{FS}}^m$ along $\Gamma$ using the integral formula (35).
   (b) Update boundary conditions correction $\mathbf{u}_{s\Gamma,\mathrm{BC}}^m$ by extrapolating in time and interpolating in space, from grid-point values of $\mathbf{u}_{\mathrm{sBC}}^{n-1}$ and $\mathbf{u}_{\mathrm{sBC}}^n$.
   (c) Update regular boundary velocity $\mathbf{u}_{r\Gamma}^m$ by extrapolating in time and interpolating in space, from grid-point values of $\mathbf{u}_r^{n-1}$ and $\mathbf{u}_r^n$.
   (d) Compute boundary velocity using (34) and (38).
   (e) Update the boundary markers $\mathbf{X}^{m+1}$ using (32) or (33).
   (f) Update the boundary forces $\mathbf{f}^{m+1}$.

(2) Set $\mathbf{X}^{n+1} = \mathbf{X}^{N_m}$ and $\mathbf{f}^{n+1} = \mathbf{f}^{N_m}$. Express jumps in solution and derivatives across $\Gamma$ in terms of boundary force $\mathbf{f}^{n+1}$.
(3) By means of the immersed interface method, compute solution of Stokes problem (11) and (12) over $\Omega$, assuming free space boundary conditions.
(4) By means of the immersed interface method, compute solution of Stokes problem (11) and (12) over $\Omega$, assuming biperiodic boundary conditions, following the procedures described in Section 2.1. This yields $\mathbf{u}_s^{n+1}$ and $p_s^{n+1}$.
(5) Compute the difference $\mathbf{u}_{\mathrm{sBC}}^{n+1}$ between the two velocities obtained in steps (3) and (4).
(6) Compute the regular solution $\mathbf{u}_r^{n+1}$ and $\nabla p_r^{n+1}$ as described in Section 2.2. The overall velocity is given by $\mathbf{u}^{n+1} = \mathbf{u}_s^{n+1} + \mathbf{u}_r^{n+1}$.

In the special case $N_m = 1$, without smaller time steps, the algorithm could be simplified; (1)(a)–(d) could be replaced by interpolating the velocity from grid values of $\mathbf{u}^n$ to the boundary locations, with care taken to incorporate jumps in the velocity derivatives into the interpolation formulae.

## 3. Numerical results

In this section, two numerical examples are used to demonstrate the accuracy and efficiency of the proposed method. All calculations reported below were performed using Fortran programs, which were executed in double precision.

### 3.1. Example 1

We first simulate the motion of a relaxing or oscillating ellipse. This example is frequently used in testing numerical methods for immersed boundary problems [29,16,15,23]. The initial boundary is an ellipse with major and minor axes set to $a = 0.7$ and $b = 0.5$, respectively. The unstretched boundary was taken to be a circle with radius $r_0 = 0.5$. The tension coefficient $T_0$ was set to 0.1. We tested the method for a fluid that is relatively viscous, with the diffusion coefficient $\mu$ set to 0.1, and then for another fluid that is significantly less viscous with $\mu = 0.01$. In both cases, the computational domain was $[-1.2, 1.2] \times [-1.2, 1.2]$. The fluid was initialized to be at rest, i.e., $\mathbf{u} = 0$ and $p = 0$ at $t = 0$.

After a sufficiently long simulation time, the interface converges to a circle with radius $r_e = \sqrt{ab} \approx 0.6124$, which is larger than the unstretched boundary but which has the same area as the initial ellipse, owing to the incompressibility of the enclosed fluid. At steady state, the fluid velocity vanishes everywhere; $p$ attains constant values inside and outside the boundary, with a jump $[p] < 0$, because the boundary is initialized to a stretched state and because the limiting fluid velocity is zero.

The model Eqs. (1)–(5) were integrated using the method of velocity decomposition to nondimensional time $t = 10$. At the final time, in both cases, the boundary approaches its steady state, but in the second case it oscillates at earlier times. Because the boundary forces in this example are not particularly stiff, the fractional time-stepping approach was not used, i.e., we set $N_m = 1$ and $\Delta t_m = \Delta t$. Fig. 1 shows the Stokes velocity ($u_s$ and $v_s$), regular velocity ($u_r$ and $v_r$), and the overall fluid velocity ($u$ and $v$), at $t = 1.2$ along $x = 0.3$, for $\mu = 0.1$ (panel A) and for $\mu = 0.01$ (panel B). In both cases, the jump discontinuities in the normal derivatives of $u_s$ and $u$ were captured sharply by the method. In contrast, the normal derivative of $u_r$ is continuous across $\Gamma$. In the more viscous case ($\mu = 0.1$), the magnitude of the Stokes velocity $u_s$ is substantially larger than $u_r$. In contrast, when $\mu = 0.01$, both $u_s$ and $u_r$ are significantly larger in magnitude than $u$, with opposite signs. We observe this relationship to hold at early times and less so at later times. Since the initial velocity is zero, $u_r = -u_s$ exactly at time zero. We expect the Stokes velocity $\mathbf{u}_s$ to have less relevance to the Navier–Stokes velocity $\mathbf{u}$ as the viscosity becomes smaller (i.e., the Reynolds number is larger).

Table 1 shows velocity accuracy and area conservation results for $N = 80$, 160, 320, and 640. The number of boundary markers scales with $N$; specifically, we set $N_k = N/2$.

The time step $\Delta t$ was set to $h$. The errors in velocity at time $t = 1.2$ were obtained at the grid points in the computational domain and at boundary markers on the boundary $\Gamma$, in $L^2$ norm and $L^\infty$ norm, using the reference solution computed on the

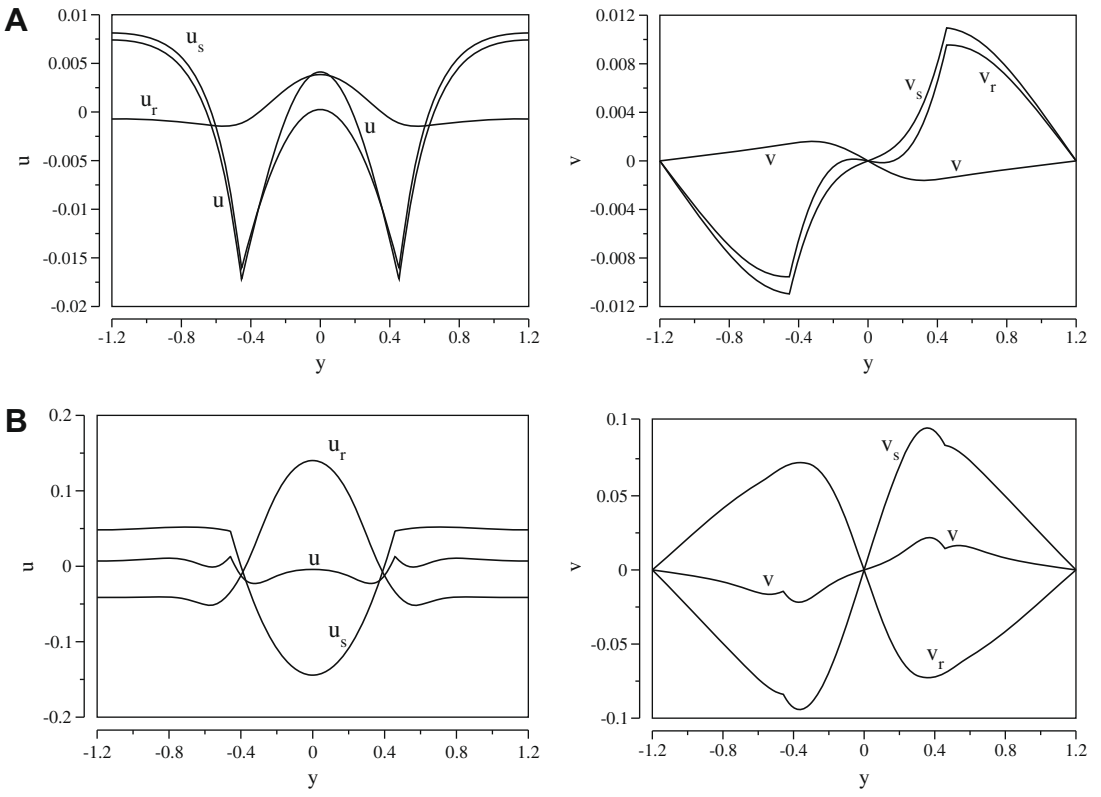


**Fig. 1.** The fluid velocity at $x = 0.3$ and $t = 1.2$. $u, v$, full velocity; $u_s, v_s$, Stokes part; $u_r, v_r$, regular part. $u, v, u_s$, and $v_s$ have a discontinuous normal derivative across the boundary, whereas $u_r$ and $v_r$ do not. A, $\mu = 0.1$; B, $\mu = 0.01$.

finest grid, with $N = 1280$. These results exhibit approximately second-order accuracy in space-time. The value of the area $A$ enclosed by $\Gamma$, computed at the approximate steady state at $t = 10$, is compared with the exact value $A = 0.7853981635$, which is known owing to incompressibility. Results for both $\mu = 0.1$ and for $\mu = 0.01$ exhibit second-order convergence at sufficiently large $N$ values.

### 3.2. Example 2

In the second test, we simulate the motion of a relaxing elastic boundary that is initialized to be the shape of a flower, given in polar coordinates by $r = 0.8 + 0.3 \sin 8\theta$. The relaxed boundary is the circle with radius $r_0 = 0.3$.

The diffusion coefficient $\mu$ was set to 0.1 and the tension coefficient to $T_0 = 10$. The computational domain was $[-1.5, 1.5] \times [-1.5, 1.5]$. The fluid was initialized to be at rest.

The larger curvature in the initial boundary configuration, compared to the first example, together with a larger tension coefficient $T_0$, renders this problem significantly stiffer. Thus, we use this example to assess the improvement in efficiency

**Table 1**
Results for Example 1. Relative errors in enclosed area $A$ and velocity $\mathbf{u}$ at grid points and on the interface $\Gamma$. The relative errors in $A$ were computed using the exact value; relative errors in $\mathbf{u}$ were computed using $N = 1280$ as the reference solution. Results show approximate second-order space-time convergence.

| $N$ | Area | $L^2$ (grid) | $L^\infty$ (grid) | $L^2$ ($\Gamma$) | $L^\infty$ ($\Gamma$) |
|---|---|---|---|---|---|
| $\mu = 0.1$ | | | | | |
| 80 | 5.291E–4 | 8.319E–4 | 1.036E–3 | 1.032E–3 | 1.176E–3 |
| 160 | 1.773E–4 | 3.026E–4 | 3.599E–4 | 2.366E–4 | 4.180E–4 |
| 320 | 5.299E–5 | 8.007E–5 | 1.160E–4 | 6.969E–5 | 1.237E–4 |
| 640 | 1.286E–5 | 1.300E–5 | 2.561E–5 | 1.462E–5 | 2.425E–5 |
| $\mu = 0.01$ | | | | | |
| 80 | 2.057E–3 | 3.734E–3 | 6.850E–3 | 1.995E–3 | 4.903E–3 |
| 160 | 6.903E–4 | 1.311E–3 | 2.822E–3 | 6.395E–4 | 1.560E–3 |
| 320 | 1.662E–4 | 3.156E–4 | 6.600E–4 | 1.674E–4 | 4.123E–4 |
| 640 | 4.053E–5 | 8.659E–5 | 1.548E–4 | 6.081E–5 | 1.025E–4 |

**Table 2**
Results for Example 2. Relative difference in velocity **u**, obtained using fractional time-stepping with $\Delta t = h$ and obtained using no fractional time-stepping but with a small time step of $\Delta t = h/10$.

| t | 1.0 | 2.0 | 3.0 |
|---|---|---|---|
| $L^2$ | 3.102E–4 | 5.953E–4 | 8.527E–4 |

that may be introduced by the fractional time-stepping approach described in Section 2.3. Before reviewing the numerical results, we first estimate the computational costs associated with the following procedures: (a) computing the boundary motion (i.e., step 1 in the algorithm presented in Section 2.4), and (b) computing the solutions over the entire domain (i.e., steps 2–5 in the algorithm). For (a), the most computationally expensive procedure in updating the boundary configuration is in computing the boundary velocity $\mathbf{u}^m_{sI,\mathrm{FS}}$ using a boundary integral; those computations take $\mathcal{O}(N^2)$ time. Thus, (a) has a computational complexity of $\mathcal{O}(N^2)$. For the overall solution in (b), the largest computational cost is associated with the FFT or fast cosine transforms, used in solving the Poisson and Laplace problems in steps 3, 4, and 6. Those computations require $\mathcal{O}(N^2 \log N)$ time. Even though the $\log N$ factor grows slowly, for sufficiently large $N$, the computations in (b) will dominate. Thus, given a sufficiently stiff boundary, which requires the forcing step $\Delta t_m$ to be prohibitively small to maintain numerical stability, we expect that the fractional time-stepping approach to lower the overall computational costs by allowing many ($N_m$) relatively quick forcing steps, per advection step, which is relatively expensive to compute. This reduction in frequency at which the overall solution is computed, and thus in computational cost, will translate to an improvement in efficiency, provided that taking larger advection steps does not significantly reduce accuracy. A rapid summation method could be used for the boundary integrals, reducing the cost of one forcing step to roughly $\mathcal{O}(N)$, and thus further improving the efficiency.

In the numerical tests, we simulated the fluid and boundary motions using two implementations of the method of decomposition, and compare the time step restriction. In the first implementation, we set $N_m = 1$, i.e., no fractional time-stepping is used. In the second implementation, we set $N_m = 10$. A spatial grid of $N = 320$ was used. With these parameters, and with $\Delta t = h$, the second implementation was stable whereas the first was not. (A time integration is considered unstable when the predicted boundary configuration sufficiently deviates from its expected course of approaching a circle. In practice, when an integration becomes unstable, model variables frequently become unrealistically large or undefined.) The first implementation attained numerical stability when $\Delta t$ was reduced to $h/10$. These results suggest that, for a stiff problem, the fractional time-stepping approach can reduce computational cost by reducing the frequency at which the solution is computed over the entire domain, as previously noted.

To assess the loss of accuracy incurred by taking a larger $\Delta t$ compared to $\Delta t_m$, we compute the solution using a third implementation, which uses $N_m = 1$, i.e., no fractional time-stepping, but a small time step of $\Delta t = h/10$. We then compare the resulting solution with the solution obtained using the fraction-stepping method above. That relative difference, computed on grid points and measured in the $L^2$ norm, is obtained for $t = 1.0$, 2.0, and 3.0; the results are shown in Table 2. At all three times, the two solutions differ by <0.1%. Thus, for this problem, the loss in accuracy is likely out-weighed by the reduction in computational costs.

## 4. Discussion

The decomposition method introduced here leads to approximations that exhibit second-order accuracy in space and time, as suggested by the numerical examples, even though grid values of the fluid quantities are corrected near the interface only for the Stokes part of the solution. This is a great advantage, since these corrections are much simpler to make than those for the full problem. Nonetheless, we do not expect that the regular velocity $\mathbf{u}_r$ is completely smooth at the interface, and it is not obvious that no corrections are needed for this part. Formulas for jump conditions as in [19] lead to the expectation that $\mathbf{u}_r$ is at least $C^2$, although the right hand side of the Eq. (13) is continuous but not $C^1$. These properties imply that the truncation error for (13) should be $\mathrm{O}(h)$ near the interface while $\mathrm{O}(h^2)$ elsewhere. Current practice with the immersed interface method, as well as the refinement studies done here, lead to the expectation that the error in the solution is uniformly $\mathrm{O}(h^2)$. For elliptic (steady state) problems, analytical results ([17,2]) explain this gain in accuracy. The analysis in [2] shows that the $\mathrm{O}(h^2)$ accuracy follows from the facts that the $\mathrm{O}(h)$ truncation error is on a lower dimensional set and that the solution of an elliptic problem gains regularity. The analysis in [2] applies to the method for solving the Stokes equations introduced in [16] and used here. For a parabolic problem such as (13) and (14) there can be an analogous gain in regularity, and therefore in accuracy, but we expect that this property depends on choosing a time discretization with decay in high spatial modes; the BDF has this property. We have proved that such a gain is possible for the simpler problem of a linear convection–diffusion equation with an interface in [1]. We plan to investigate the accuracy for the Navier–Stokes equations with interfaces in future work.

We have seen that stiffness can be dealt with in this approach by the use of small partial time steps for free-space Stokes flow with the interfacial force. If the coefficient $T_0$ in (5) is large, or if (5) is replaced by a more general force which includes higher derivatives, such as a bending force (e.g., [28,30]), then the stiffness is severe. In these cases implicit treatment is needed for the dependence of the force on the interface location. Implicit versions of the immersed boundary method [29,21,23,12] and other interface methods [16,15,13] have already been introduced. With the present decomposition an

implicit step should be needed only for the Stokes part of the solution, since the interfacial force is incorporated in this part. For two-dimensional flow, with a closed curve tracked with markers, as we have done here, we expect a technique like that of [11] (cf. [13,12]) can be used at least in some cases.

The decomposition used here applies as well to three-dimensional flow; the immersed interface method for the Stokes part and the semi-Lagrangian method for the regular part both extend naturally to 3D. The motion of the interface would have to be represented differently in three dimensions. The level set method can be used in 2D or 3D [19,15]. In [3], 2D periodic Stokes flow with an interface was computed using Strain's semi-Lagrangian contouring method for the interface motion and Ewald summation for velocity integrals, an approach which extends to three dimensions.

It was assumed in this work that the fluid is the same on both sides of the interface. If instead the interface separates two Navier–Stokes fluids with different viscosities and densities, the analogue of $\mathbf{u}_r$ will no longer be regular, although the right hand side of its evolution equation will be less singular than the interfacial force. It remains to be seen whether the decomposition would reduce the difficulty of the two-fluid problem.

## Acknowledgments

Ming-Chih Lai participated in early work in this direction, and it is a pleasure to thank him for his contributions. This work was supported in part by the National Science Foundation under Grants DMS-0806482 (J.T. Beale) and DMS-0715021 (A.T. Layton).

## References

[1] J.T. Beale, Smoothing properties of implicit finite difference methods for a diffusion equation in maximum norm, SIAM J. Numer. Anal. submitted for publication.
[2] J.T. Beale, A.T. Layton, On the accuracy of finite difference methods for elliptic problems with interfaces, Commun. Appl. Math. Comput. Sci. 1 (2006) 91–119.
[3] J.T. Beale, J. Strain, Locally corrected semi-Lagrangian methods for Stokes flow with moving elastic interfaces, J. Comput. Phys. 227 (2008) 3896–3920.
[4] G. Biros, L. Ying, D. Zorin, An embedded boundary integral solver for the unsteady incompressible Navier–Stokes equations, SIAM J. Sci. Comput, submitted for publication.
[5] R. Cortez, M. Minion, The blob projection method for immersed boundary problems, J. Comput. Phys. 161 (2000) 428–453.
[6] D. Durran, Numerical Methods for Wave Equations in Geophysical Fluid Dynamics, Springer, 1999.
[7] M. Falcone, R. Ferretti, Convergence analysis for a class of high-order semi-Lagrangian advection schemes, SIAM J. Numer. Anal. 35 (1998) 909–940.
[8] R. Glowinski, Finite element methods for incompressible viscous flow, Handbook of Numerical Analysis, vol. IX, North-Holland, 2003.
[9] B. Griffith, R. Hornung, D. McQueen, C. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, J. Comput. Phys. 223 (2007) 10–49.
[10] B. 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.
[11] T.Y. Hou, J.S. Lowengrub, M.J. Shelley, Removing the stiffness from interfacial flows with surface tension, J. Comput. Phys. 114 (1994) 312–338.
[12] T.Y. Hou, Z. Shi, Removing the stiffness of elastic force from the immersed boundary method for the 2d stokes equations, J. Comput. Phys. 227 (2008) 9138–9169.
[13] M. Kropinski, An efficient numerical method for studying interfacial motion in two-dimensional creeping flows, J. Comput. Phys. 171 (2001) 479–508.
[14] D. Le, B. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, J. Comput. Phys. 220 (2006) 109–138.
[15] L. Lee, R.J. LeVeque, An immersed interface method for the incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (2003) 832–856.
[16] 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.
[17] Z. Li, K. Ito, Maximum principle preserving schemes for interface problems, SIAM J. Sci. Comput. 23 (2001) 339–361.
[18] Z. Li, K. Ito, The Immersed Interface Method: Numerical Solutions of PDEs Involving Interface and Irregular Domains, SIAM, Philadelphia, PA, 2006.
[19] Z. Li, M.-C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822–842.
[20] A. Mayo, The fast solution of Poisson's and the biharmonic equations on irregular regions, SIAM J. Numer. Anal. 21 (1984) 285–299.
[21] A. Mayo, C.S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, in: A.Y. Cheer, C.P. von Dam (Eds.), Fluid Dynamics in Biology, AMS, Providence, RI, 1993, pp. 261–278.
[22] Y. Mori, C.S. Peskin, Implicit second-order immersed boundary methods with boundary mass, Comput. Methods Appl. Mech. Eng. 197 (2008) 2049–2067.
[23] E.P. Newren, A.L. Fogelson, R.D. Guy, R.M. Kirby, Unconditionally stable discretizations of the immersed boundary equations, J. Comput. Phys. 222 (2007) 702–719.
[24] C.S. Peskin, The immersed boundary method, Acta Numer. (2002) 1–39.
[25] 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.
[26] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, Cambridge University Press, Cambridge, 1992.
[27] 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.
[28] A.-K. Tornberg, M. Shelley, Simulating the dynamics and interactions of flexible fibers in Stokes flows, J. Comput. Phys. 196 (2005) 8–40.
[29] C. Tu, C.S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries: a comparison of three methods, SIAM J. Sci. Statist. Comput. 13 (1992) 1361–1376.
[30] S.K. Veerapaneni, D. Geuyffier, D. Zorin, G. Biros, A boundary integral method for simulating the dynamics of inextensible vesicles suspended in a viscous fluid in 2d, J. Comput. Phys. 228 (2009) 2334–2353.
[31] D. Xiu, G. Karniadakis, A semi-Lagrangian high-order method for Navier–Stokes equations, J. Comput. Phys. 172 (2001) 658–684.
[32] 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.