Journal of Computational Physics 229 (2010) 4208–4224

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A multirate time integrator for regularized Stokeslets

Elizabeth L. Bouzarth [*,1,2], Michael L. Minion [3]

University of North Carolina, Department of Mathematics, Phillips Hall, CB #3250, Chapel Hill, NC 27599, United States

ARTICLE INFO

Article history:
Received 11 March 2009
Received in revised form 21 January 2010
Accepted 5 February 2010
Available online 12 February 2010

Keywords:
Stokes flow
Deferred corrections
Regularized Stokeslets

ABSTRACT

The method of regularized Stokeslets is a numerical approach to approximating solutions of fluid–structure interaction problems in the Stokes regime. Regularized Stokeslets are fundamental solutions to the Stokes equations with a regularized point-force term that are used to represent forces generated by a rigid or elastic object interacting with the fluid. Due to the linearity of the Stokes equations, the velocity at any point in the fluid can be computed by summing the contributions of regularized Stokeslets, and the time evolution of positions can be computed using standard methods for ordinary differential equations. Rigid or elastic objects in the flow are usually treated as immersed boundaries represented by a collection of regularized Stokeslets coupled together by virtual springs which determine the forces exerted by the boundary in the fluid. For problems with boundaries modeled by springs with large spring constants, the resulting ordinary differential equations become stiff, and hence the time step for explicit time integration methods is severely constrained. Unfortunately, the use of standard implicit time integration methods for the method of regularized Stokeslets requires the solution of dense nonlinear systems of equations for many relevant problems. Here, an alternate strategy using an explicit multirate time integration scheme based on spectral deferred corrections is incorporated that in many cases can significantly decrease the computational cost of the method. The multirate methods are higher-order methods that treat different portions of the ODE explicitly with different time steps depending on the stiffness of each component. Numerical examples on two nontrivial three-dimensional problems demonstrate the increased efficiency of the multi-explicit approach with no significant increase in numerical error.

1. Introduction

This paper presents a new strategy for the temporal integration of ordinary differential equations that can be decomposed into multiple components with various levels of stiffness. Specifically, higher-order multirate methods are constructed from a slightly more general type of deferred correction scheme. The term *multirate* is used here to describe a method that uses different time steps to integrate different parts of the splitting of a system of ODEs (*e.g.* as in [1]). The multirate methods are based on a multi-explicit spectral deferred correction (MESDC) method designed for systems of ODEs that are computationally expensive to treat with a fully implicit method. The MESDC method is an explicit variation of the spectral deferred

* Corresponding author. Tel.: +919 660 2800; fax: +919 660 2821.
E-mail addresses: bouzarth@math.duke.edu (E.L. Bouzarth), minion@amath.unc.edu (M.L. Minion).
URLs: http://www.math.duke.edu/~bouzarth (E.L. Bouzarth), http://www.amath.unc.edu/Faculty/minion (M.L. Minion).
[1] Present address: Duke University, Department of Mathematics, Box #90320, Durham, NC 27708, Unites States.
[2] This work was supported by the National Science Foundation and the Department of Energy through grants NSF RTG DMS-0502266 and DOE SciDac DE-FC02-01ER25471.
[3] This work was supported by the Alexander von Humboldt Foundation and the Director, DOE Office of Science, Office of Advanced Scientific Computing Research, Office of Mathematics, Information, and Computational Sciences, Applied Mathematical Sciences Program, under contract DE-AC03-76SF00098.

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.02.006

correction (SDC) method developed by Dutt et al. [2] and is similar to the multi-implicit SDC methods presented by Bourlioux et al. and Layton et al. [3,4].

For equations for which the computation of the contributions from the stiffest components of the system is relatively inexpensive to compute, the use of a multirate approach can lead to a decrease in the overall computational cost. The most likely scenario for finding a splitting for which the stiff term is computationally less expensive to compute is when a splitting of the variables themselves is possible (as explained in Section 3.2). Other possibilities exist, including the case where stiff terms are more easily parallelizable than non-stiff terms, as is typically true for advection-diffusion–reaction type systems. We present a framework for constructing explicit, higher-order, multirate schemes for such situations.

The motivation for the development of the MESDC method comes from the use of the method of regularized Stokeslets for approximating fluid–structure interactions in Stokes flow. The method of regularized Stokeslets utilizes Lagrangian computational elements (Stokeslets) to represent an immersed boundary in the fluid domain. The use of a system of regularized singularities connected by virtual springs to model flexible structures interacting with a surrounding fluid was popularized by the seminal work of Peskin and McQueen on the immersed boundary method (*e.g.* see [5–8]). The original motivation for the immersed boundary method was to model the dynamics of simplified models of the human heart. The physical membranes of the heart are modeled by a collection of discrete delta functions which transfer force to the surrounding incompressible fluid (which is discretized with a standard projection method). Forces are generated at the singularity positions through a virtual spring law applied to neighboring singularities to mimic stretching forces or a similar local construction to create normal forces to mimic bending forces. When the membranes are stiff, the underlying system of ODEs governing the discretized fluid–structure dynamics is also stiff. In many immersed boundary applications, this stiffness restricts the size of the maximum stable time step to be much smaller than the CFL limit of the associated projection method (*e.g.* see [9–12]).

The immersed boundary method has since been widely applied in diverse applications ranging from the motion of molecular motors [13] to the unfolding of parachutes [10]. The ease of implementation of the immersed boundary method has also led to the use of stiff immersed boundaries to model rigid boundaries in incompressible fluids. The advantage of this technique is that arbitrarily complex boundaries can be easily superimposed over a uniform computational grid on which the Navier–Stokes equations are solved. The drawback is that in order to make a boundary virtually rigid, the spring forces connecting the boundary points that enforce the rigidity must be made very stiff, which leads to a correspondingly stiff set of ODEs to solve.

Although one could use an immersed boundary approach for Stokes flow [14,15], the method of regularized Stokeslets, developed by Cortez [16,17], is more attractive for many problems in the Stokes regime since it avoids the necessity of solving the full fluid equations on an underlying numerical grid. However, the use of spring forces to model stiff and rigid bodies interacting with the flow still introduces (at times severe) stiffness into the underlying temporal ODE. Due to the nonlinear and nonlocal coupling of the fluid velocity and singularity position in both the immersed boundary and regularized Stokeslets methods, constructing efficient implicit or semi-implicit temporal integration methods for immersed boundary problems is a challenging task (*e.g.* see [18–22]). Therefore, a different approach to reduce the computational cost of stiff problems using an explicit multirate approach based on MESDC is investigated here. The method is tested on two target applications involving moving objects in three-dimensional Stokes flow, although the general idea is applicable to a wider class of problems. An extension of the MESDC approach to immersed boundary problems in the Navier–Stokes regime is under development and will be reported in a future paper.

The discussion in Section 3 describes the MESDC method for a general ODE that can be separated into components with varying levels of stiffness, each to be treated with its own time step. In Section 4, two application examples are discussed, both of that are motivated by current experimental fluid dynamics research being conducted at the University of North Carolina [23–25]. The first example models a slender rigid spheroidal rod precessing in an incompressible fluid that also contains a rigid sphere, while the second models multiple flexible fibers inspired by an experiment to investigate the flow through the endothelial glycocalyx.

The examples differ significantly in how velocities are imposed and computed at the regularized Stokeslet positions. In the first case, the position of the rod is prescribed, and a dense linear system must be solved to compute the Stokeslet strengths that impose the prescribed velocity boundary conditions on the rod. On the other hand, the sphere moves with the fluid velocity, and its rigidity is imposed by the use of virtual spring forces between Stokeslets. The stiffness of these springs translates into an increased stiffness in the underlying ODE describing the motion of the Stokeslets and hence the need to use smaller time steps for an explicit ODE method. Thus, the aforementioned linear systems (which are computationally expensive for a large system of many regularized Stokeslets) need to be solved more frequently when the time step is smaller.

In the second example, only a background velocity is imposed at all Stokeslet locations and the forces arise only from virtual springs connecting the Stokeslets. The velocity at any evaluation point must hence be computed by a sum over all the Stokeslet contributions.

One goal in implementing the MESDC method is to use a coarse time step for the non-stiff component which requires expensive linear solves and a smaller time step to accommodate the stiff components of the system. While implementing an implicit method may seem like a logical choice since it allows for large time steps with stiff systems, doing so for the method of regularized Stokeslets is undesirable since it would require the solution of a large, dense, nonlinear system coupling both Stokeslet positions and forces each time step. It will be shown that the MESDC method is an accurate, stable explicit method utilizing two different time steps to accommodate the needs of different components of the physical system.

The remainder of this paper is organized as follows. Section 2 provides an overview of spectral deferred correction methods and Section 3 introduces the multi-explicit spectral deferred correction method including a discussion of efficiency in a general context. The specific examples of using regularized Stokeslets to model a precessing rigid rod and a rigid sphere interacting in an incompressible fluid and the flexible fiber model are discussed in Section 4. Numerical tests of the MESDC method on the rod/sphere and flexible fiber problems are presented in Section 5 followed by a discussion of the results and future research directions.

2. Overview of spectral deferred corrections

The basic strategy of spectral deferred correction (SDC) methods developed by Dutt et al. [2] is to use a simple numerical scheme to compute a provisional solution for a time step and then to iteratively solve correction equations during the time step to improve the accuracy of the provisional solution. One advantage of using SDC methods is that one can compute a solution to an arbitrarily high formal order of accuracy using a simple numerical method.

For a general overview of the SDC method, consider this governing ordinary differential equation:

$$x'(t) = u(t, x(t)), \tag{1}$$

$$x(a) = x_a, \tag{2}$$

for $t \in [a, b]$ where it is assumed $x_a, x(t) \in \mathbb{C}^n$, $u : \mathbb{R} \times \mathbb{C}^n \to \mathbb{C}^n$, and u is sufficiently smooth so that higher-order methods are appropriate. Consider the Picard integral equation for the solution:

$$x(t) = x_a + \int_a^t u(\tau, x(\tau))d\tau. \tag{3}$$

Let $\tilde{x}(t)$ represent a provisional solution to the integral (3) and define the residual, $E(t, \tilde{x})$, as

$$E(t, \tilde{x}) = x_a + \int_a^t u(\tau, \tilde{x}(\tau))d\tau - \tilde{x}(t). \tag{4}$$

Also define the error, $\delta(t)$, as the difference between the solution and the provisional solution:

$$\delta(t) = x(t) - \tilde{x}(t). \tag{5}$$

Substitute the error from (5) into the equation for the solution (3) to obtain

$$\delta(t) + \tilde{x}(t) = x_a + \int_a^t u(\tau, \tilde{x}(\tau) + \delta(\tau))d\tau. \tag{6}$$

Combining (6) with the residual (4) produces another form of the correction equation:

$$\delta(t) = \int_a^t [u(\tau, \tilde{x}(\tau) + \delta(\tau)) - u(\tau, \tilde{x}(\tau))]d\tau + E(t, \tilde{x}). \tag{7}$$

A discretized form of this equation will be used in conjunction with the provisional solution to update the solution.

The goal of SDC is to compute a solution in the ith time step, say $[t_i, t_{i+1}]$. First split the time step into N_m substeps such that $t_i = t_{i,0} < t_{i,1} < \cdots < t_{i,m} < \cdots < t_{i,N_m} = t_{i+1}$, as shown in Fig. 1. Since a spectral deferred correction method is being used, one chooses the nodes of the substeps to correspond to Gaussian quadrature rules. For the discussion and examples presented here, Lobatto nodes are used. Layton and Minion explore other choices of quadrature nodes in detail in [26]. Thus, the size of each substep, Δt_m, is not uniform, but is naturally defined as $\Delta t_m = t_{i,m+1} - t_{i,m}$. Now compute the provisional solution within the ith time step at each node of the substep level, $\tilde{x}^k(t_{i,m})$, which will be simplified notationally as $\tilde{x}^k_m$ (the superscript k denotes the iteration number that will be discussed shortly). In addition, to help simplify notation, the i subscript will be dropped when it is implied, so that $t_m = t_{i,m}$. Next, approximate the error $\delta^k_m = \delta^k(t_m)$ using a first-order explicit method analogous to the forward Euler scheme applied to Eq. (7):

$$\delta^k_{m+1} = \delta^k_m + \Delta t_m \left[u(t_m, \tilde{x}^k_m + \delta^k_m) - u(t_m, \tilde{x}^k_m)\right] + E_{m+1}(\tilde{x}^k) - E_m(\tilde{x}^k). \tag{8}$$

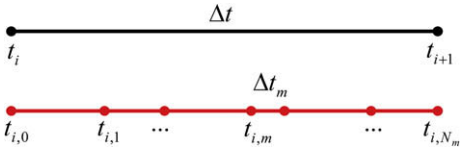

Fig. 1. The ith time step $[t_i, t_{i+1}]$ is split into N_m substeps $[t_{i,m}, t_{i,m} + \Delta t_m]$, $m = 0, 1, \ldots, N_m - 1$ for use with the SDC Method.

The final two terms in (8) must be approximated by numerical quadrature. Use the residual in (4) to write $E_{m+1}(\tilde{x}^k) - E_m(\tilde{x}^k)$ as

$$E_{m+1}(\tilde{x}^k) - E_m(\tilde{x}^k) = \int_{t_m}^{t_{m+1}} u(\tau, \tilde{x}^k(\tau))d\tau - \tilde{x}^k_{m+1} + \tilde{x}^k_m. \tag{9}$$

Approximate the integrand u in (9) with an interpolating polynomial

$$\hat{u}(t) = \sum_{j=0}^{N_m} u(t_j)l_j(t), \tag{10}$$

where $l_j(t)$ represents the jth Lagrange polynomial that satisfies $l_j(t_i) = \delta_{ij}$:

$$l_j(t) = \prod_{i=0, i\neq j}^{N_m} \frac{t - t_i}{t_j - t_i}. \tag{11}$$

Let $I_m^{m+1}(\tilde{x}^k)$ approximate the integral in (9) by

$$I_m^{m+1}(\tilde{x}^k) = \int_{t_m}^{t_{m+1}} \hat{u}(t)d\tau. \tag{12}$$

Note that the choice of quadrature in $I_m^{m+1}(\tilde{x}^k)$ depends on the choice of t_m.

Now (8) becomes

$$\delta^k_{m+1} = \delta^k_m + \Delta t_m\left[u(t_m, \tilde{x}^k_m + \delta^k_m) - u(t_m, \tilde{x}^k_m)\right] + I_m^{m+1}(\tilde{x}^k) - \tilde{x}^k_{m+1} + \tilde{x}^k_m. \tag{13}$$

Now use (13) and $\tilde{x}^{k+1} = \tilde{x}^k + \delta^k$ to update the provisional solution:

$$\tilde{x}^{k+1}_{m+1} = \tilde{x}^{k+1}_m + \Delta t_m\left[u(t_m, \tilde{x}^k_m + \delta^k_m) - u(t_m, \tilde{x}^k_m)\right] + I_m^{m+1}(\tilde{x}^k). \tag{14}$$

Each iteration of the correction equation increases the formal order of accuracy of the solution by one, provided the quadrature in (12) is accurate enough [2].

3. Multirate and multi-explicit spectral deferred corrections

This section outlines the construction of multirate integration schemes based on SDC methods. Variations of SDC methods exist that treat components of an ODE with different substeps and/or a mix of implicit and explicit treatments. In [27], Minion develops a *semi-implicit* SDC method allowing non-stiff components of an equation to be treated explicitly while the stiff components are treated implicitly. These semi-implicit methods are similar to IMEX Additive Runge–Kutta (*e.g.* [28–31]) or linear multistep schemes (*e.g.* [32–34]) in that both stiff and non-stiff (*i.e.* implicit and explicit) terms are treated with the same time step. The semi-implicit approach is taken a step further in [3,4], where the advective term of an advection-diffusion–reaction type equation is treated explicitly and the diffusive and reactive terms are treated implicitly but independently, and each component can have different time steps. These methods are termed *multi-implicit* to signify that the right hand side of the discretized equations is split into multiple parts, more than one of which is treated implicitly.

While one could consider implicit ODE methods for the regularized Stokeslet applications considered here, this leads to the need to solve large dense nonlinear systems which are computationally very expensive to solve. Hence we explore instead a multirate explicit approach which is computationally more efficient than a single time step explicit approach. We refer to methods which use different time steps for different terms in a splitting of the equations as *multirate* methods (*e.g.* see [1,35–37]). The multirate methods are based on a *multi-explicit* SDC method (MESDC), which signifies that the right hand side of the discretized equations is split into multiple parts that are each treated explicitly. Such methods are slightly more general than an explicit multirate scheme in that they also apply to the case where all components of the system are updated with the same overall time step but with the right hand side of the equation split into multiple pieces each with a different time step.

In the case of the rigid or elastic bodies moving within a Stokes flow, which will be discussed further in Section 4, the representation of the bodies by spring-connected regularized Stokeslets introduces stiffness into the system. A natural splitting of the discretized equations results in splitting the contribution to the velocity at a given position into those coming from nearby Stokeslets and those coming from well separated Stokeslets, which are presumably not connected by virtual springs.

3.1. MESDC overview

In this section the MESDC method will be described for a system of ODEs treated with two different levels of time discretization. Specifically, let $u = u_1 + u_2$ where u_1 and u_2 will be treated with a large and small time step, respectively. Extensions to a splitting into more than two pieces can be constructed as in [3]. The ODE is hence

$$x'(t) = u_1(t, x(t)) + u_2(t, x(t)), \tag{15}$$
$$x(a) = x_a, \tag{16}$$

for $t \in [a, b]$, assuming as in Section 2 that $x_a, x(t) \in \mathbb{C}^n, u_1, u_2 : \mathbb{R} \times \mathbb{C}^n \to \mathbb{C}^n$ and sufficiently smooth. Then the Picard integral equation for the solution is

$$x(t) = x_a + \int_a^t [u_1(\tau, x(\tau)) + u_2(\tau, x(\tau))]d\tau. \tag{17}$$

Given a provisional solution to the integral equation for the ith time step, $t \in [t_i, t_{i+1}] \in [a, b], \tilde{x}(t)$, define the residual $E(t, \tilde{x})$ by

$$E(t, \tilde{x}) = x(t_i) + \int_{t_i}^t [u_1(\tau, \tilde{x}(\tau)) + u_2(\tau, \tilde{x}(\tau))]d\tau - \tilde{x}(t). \tag{18}$$

Incorporating the error from (5) into (17) generates

$$\delta(t) + \tilde{x}(t) = x(t_i) + \int_{t_i}^t [u_1(\tau, \tilde{x}(\tau) + \delta(\tau)) + u_2(\tau, \tilde{x}(\tau) + \delta(\tau))]d\tau. \tag{19}$$

Moreover, incorporating the residual from (18) produces the correction equation:

$$\delta(t) = \int_{t_i}^t [u_1(\tau, \tilde{x}(\tau) + \delta(\tau)) - u_1(\tau, \tilde{x}(\tau)) + u_2(\tau, \tilde{x}(\tau) + \delta(\tau)) - u_2(\tau, \tilde{x}(\tau))]d\tau + E(t, \tilde{x}). \tag{20}$$

The integral Eq. (20) is approximated by coupled operator splitting formulation:

$$\delta_1(t) = \int_{t_i}^t [u_1(\tau, \tilde{x}(\tau) + \delta_1(\tau)) - u_1(\tau, \tilde{x}(\tau))]d\tau + E(t, \tilde{x}). \tag{21}$$
$$\delta(t) = \int_{t_i}^t [u_1(\tau, \tilde{x}(\tau) + \delta_1(\tau)) - u_1(\tau, \tilde{x}(\tau)) + u_2(\tau, \tilde{x}(\tau) + \delta(\tau)) - u_2(\tau, \tilde{x}(\tau))]d\tau + E(t, \tilde{x}). \tag{22}$$

Since only a first-order approximation of the correction equation is required in the SDC method, this operator splitting will not affect the overall order of accuracy of the method. The expression for δ_1 in (21) only contains terms relating to u_1, thus, a larger time step can be used to calculate δ_1. However, δ in (22) incorporates both u_1 and u_2 terms and hence should be treated with a smaller time step. As such, each integral in (21) and (22) can be treated with a different time step when discretized, as demonstrated in the following example.

As in the discussion of SDC methods in Section 2, the following example uses a first-order explicit discretization of the integral equations analogous to the forward Euler method. Again, the goal is to find a solution on the ith time step $[t_i, t_{i+1}]$, which is split into N_m substeps: $t_i = t_{i,0} < t_{i,1} < \cdots < t_{i,m} < \cdots < t_{i,N_m} = t_{i+1}$. Each substep is further split into N_p substeps: $t_{i,m} = t_{i,m,0} < t_{i,m,1} < \cdots < t_{i,m,p} < \cdots < t_{i,m,N_p} = t_{i,m+1}$. Fig. 2 shows the relationship between a time step and both levels of substeps. As discussed in Section 2 with regard to SDC, the size of each substep can vary depending on the quadrature method chosen. Again, $\Delta t_m = t_{i,m+1} - t_{i,m}$ and $\Delta t_p = t_{i,m,p+1} - t_{i,m,p}$. To simplify notation, let $t_m = t_{i,m}$ and $t_p = t_{i,m,p}$ when the deleted subscripts are understood. Now compute a provisional solution, $\tilde{x}_{p+1}$, as follows:

$$\tilde{x}_{p+1} = \tilde{x}_p + \Delta t_p \left[u_{1,m}(\tilde{x}_m) + u_{2,p}(\tilde{x}_p)\right]. \tag{23}$$

Note that this solution is computed on the fine substep level (indexed by p) but it uses u_1 velocity values from the coarse substep (indexed by m) and u_2 velocities on the fine substep. That is, u_1 is constant within each substep.

Now approximate $\delta^k_{1,m+1}$ and δ^k_{p+1} from the correction Eqs. (21) and (22), respectively, with forward Euler:

$$\delta^k_{1,m+1} = \delta^k_{1,m} + \Delta t_m \left[u_1(t_m, \tilde{x}^k_m + \delta^k_{1,m}) - u_1(t_m, \tilde{x}^k_m)\right] + E_{m+1}(\tilde{x}^k) - E_m(\tilde{x}^k). \tag{24}$$
$$\delta^k_{p+1} = \delta^k_p + \Delta t_p \left[u_1(t_m, \tilde{x}^k_m + \delta^k_{1,m}) - u_1(t_m, \tilde{x}^k_m) + u_2(t_p, \tilde{x}^k_p + \delta^k_p) - u_2(t_p, \tilde{x}^k_p)\right] + E_{p+1}(\tilde{x}^k) - E_p(\tilde{x}^k), \tag{25}$$

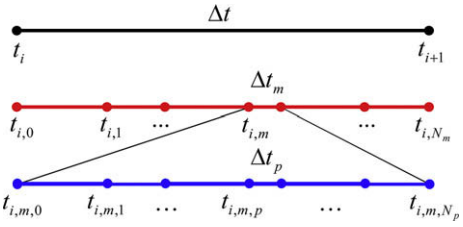

Fig. 2. Time step $[t_i, t_{i+1}]$ is split into substeps corresponding to the time step for $u_1, [t_{i,m}, t_{i,m} + \Delta t_m]$, and smaller substeps for the $u_2, [t_{i,m,p}, t_{i,m,p} + \Delta t_p]$, where $\Delta t_p \leqslant \Delta t_m \leqslant \Delta t$.

where $E_{m+1}(\tilde{x}^k) - E_m(\tilde{x}^k)$ and $E_{p+1}(\tilde{x}^k) - E_p(\tilde{x}^k)$ correspond to the residual defined in (18):

$$E_{m+1}(\tilde{x}^k) - E_m(\tilde{x}^k) = \int_{t_m}^{t_{m+1}} \left[u_1(\tau, \tilde{x}^k(\tau)) + u_2(\tau, \tilde{x}^k(\tau))\right] d\tau - \tilde{x}^k_{m+1} + \tilde{x}^k_m, \tag{26}$$

$$E_{p+1}(\tilde{x}^k) - E_p(\tilde{x}^k) = \int_{t_p}^{t_{p+1}} \left[u_1(\tau, \tilde{x}^k(\tau)) + u_2(\tau, \tilde{x}^k(\tau))\right] d\tau - \tilde{x}^k_{p+1} + \tilde{x}^k_p. \tag{27}$$

As before, define the approximation of (26):

$$I_m^{m+1}(u_1 + u_2) \approx \int_{t_m}^{t_{m+1}} \left[u_1(\tau, \tilde{x}^k(\tau)) + u_2(\tau, \tilde{x}^k(\tau))\right] d\tau. \tag{28}$$

The form of $I_m^{m+1}(u_1 + u_2)$ will be developed as follows, starting with the integral in (28) :

$$\int_{t_m}^{t_{m+1}} \left[u_1(\tau, \tilde{x}^k(\tau)) + u_2(\tau, \tilde{x}^k(\tau))\right] d\tau = \int_{t_m}^{t_{m+1}} u_1(\tau, \tilde{x}^k(\tau)) d\tau + \sum_{p=0}^{N_p-1} \int_{t_{m,p}}^{t_{m,p+1}} u_2(\tau, \tilde{x}^k(\tau)) d\tau. \tag{29}$$

Since u_1 is only known at the coarse nodes (t_m's) and the integration occurs from t_m to t_{m+1}, use an interpolating polynomial to approximate the function values at all $N_m + 1$ points on the coarse substep level. Integrating u_2 from t_p to t_{p+1} also requires an interpolating polynomial that matches the velocity value at each of the $N_p + 1$ nodes of the mth coarse substep. Use the known velocity values to find interpolating polynomials $\hat{u}_1 \approx u_1$ and $\hat{u}_2 \approx u_2$ that will be used shortly in quadrature:

$$\hat{u}_1(t) = \sum_{m'=0}^{N_m} u_{1,m'} l_{m'}(t), \tag{30}$$

$$\hat{u}_2(t) = \sum_{p'=0}^{N_p} u_{2,p'} l_{p'}(t), \tag{31}$$

where $l_j(t)$ represents basis polynomials that satisfy $l_j(t_i) = \delta_{ij}$ for $j = m', p'$. Namely,

$$l_{m'}(t) = \prod_{i=0, i \neq m'}^{N_m} \frac{t - t_i}{t_{m'} - t_i} \tag{32}$$

and

$$l_{p'}(t) = \prod_{i=0, i \neq p'}^{N_p} \frac{t - t_i}{t_{p'} - t_i}. \tag{33}$$

Hence

$$\int_{t_m}^{t_{m+1}} \left[u_1(\tau, \tilde{x}^k(\tau)) + u_2(\tau, \tilde{x}^k(\tau))\right] d\tau \approx \sum_{m'=0}^{N_m} u_{1,m'} q_m^{m'} + \sum_{p'=0}^{N_p} u_{2,p'} q^{p'}, \tag{34}$$

where

$$q_m^{m'} = \int_{t_m}^{t_{m+1}} l_{m'}(\tau) d\tau, \tag{35}$$

$$q^{p'} = \sum_{p=0}^{N_p-1} q_p^{p'}, \tag{36}$$

$$q_p^{p'} = \int_{t_{m,p}}^{t_{m,p+1}} l_{p'}(\tau) d\tau. \tag{37}$$

Now the form of I_m^{m+1} is defined as (34):

$$I_m^{m+1}(u_1 + u_2) = \sum_{m'=0}^{N_m} u_{1,m'} q_m^{m'} + \sum_{p'=0}^{N_p} u_{2,p'} q^{p'}. \tag{38}$$

As in Section 2, the choice of substep nodes t_m and t_p relate to the quadrature used in I_m^{m+1}.

Similarly, consider approximating the integral from (27):

$$I_p^{p+1}(u_1 + u_2) \approx \int_{t_p}^{t_{p+1}} \left[u_1(\tau, \tilde{x}^k(\tau)) + u_2(\tau, \tilde{x}^k(\tau))\right] d\tau. \tag{39}$$

Again, to develop the form of I_p^{p+1}, begin with the integral in question and utilize interpolating polynomials to discover

$$\int_{t_p}^{t_{p+1}} \left[u_1(\tau,\tilde{x}^k(\tau)) + u_2(\tau,\tilde{x}^k(\tau))\right] d\tau \approx \sum_{m'=0}^{N_m} u_{1,m'} q_{m,p}^{m'} + \sum_{p'=0}^{N_p} u_{2,p'} q_p^{p'}, \tag{40}$$

where

$$q_{m,p}^{m'} = \int_{t_p}^{t_{p+1}} l_{m'}(\tau) d\tau, \tag{41}$$

$$q_p^{p'} = \int_{t_p}^{t_{p+1}} l_{p'}(\tau) d\tau. \tag{42}$$

Thus,

$$I_p^{p+1}(u_1+u_2) = \sum_{m'=0}^{N_m} u_{1,m'} q_{m,p}^{m'} + \sum_{p'=0}^{N_p} u_{2,p'} q_p^{p'}. \tag{43}$$

Now that (38) and (43) define I_m^{m+1} and I_p^{p+1}, respectively, all of the components of the correction equations can be computed. Combining (26), (27), (38), and (43) into (24) and (25) produces a new form of the correction equations:

$$\delta_{1,m+1}^k = \delta_{1,m}^k + \Delta t_m \left[u_1(t_m,\tilde{x}_m^k + \delta_{1,m}^k) - u_1(t_m,\tilde{x}_m^k)\right] + I_m^{m+1}(u_1(\tilde{x}^k) + u_2(\tilde{x}^k)) - \tilde{x}_{m+1}^k + \tilde{x}_m^k, \tag{44}$$

$$\delta_{p+1}^k = \delta_p^k + \Delta t_p \left[u_1(t_m,\tilde{x}_m^k + \delta_{1,m}^k) - u_1(t_m,\tilde{x}_m^k) + u_2(t_p,\tilde{x}_p^k + \delta_p^k) - u_2(t_p,\tilde{x}_p^k)\right] + I_p^{p+1}(u_1(\tilde{x}^k) + u_2(\tilde{x}^k)) - \tilde{x}_{p+1}^k + \tilde{x}_p^k. \tag{45}$$

Finally, $\tilde{x}^{k+1} = \tilde{x}^k + \delta^k$ is used in addition to (44) and (45) to update the provisional solution:

$$\tilde{x}_{p+1}^{k+1} = \tilde{x}_p^{k+1} + \Delta t_p \left[u_1(t_m,\tilde{x}_m^k + \delta_{1,m}^k) - u_1(t_m,\tilde{x}_m^k) + u_2(t_p,\tilde{x}_p^k + \delta_p^k) - u_2(t_p,\tilde{x}_p^k)\right] + I_p^{p+1}(u_1(\tilde{x}^k) + u_2(\tilde{x}^k)). \tag{46}$$

Each iteration (in k) of the correction equation increases the order of accuracy by one, provided the quadrature rules used to calculate I_m^{m+1} and I_p^{p+1} are sufficiently accurate.

3.2. MESDC efficiency

We now illustrate particular scenarios where using a multirate explicit numerical method results in a substantial gain in numerical efficiency. Although the discussion is somewhat general, it is relevant to the time integration scheme in the method of regularized Stokeslets. This connection will be made clear in Section 4.

Consider the particular case of an ODE of the form

$$\mathbf{x}' = \mathbf{u}(\mathbf{x}) = \mathbf{A}(\mathbf{x})\mathbf{f}(\mathbf{x},t) \tag{47}$$

where $\mathbf{x}$ and $\mathbf{f}(\mathbf{x},t)$ are vectors of length N and $\mathbf{A}$ is an $N \times N$ matrix whose entries depend on $\mathbf{x}$. Furthermore, let $\mathbf{x}$ be naturally decomposed into two pieces $\mathbf{x}_1$ and $\mathbf{x}_2$ with lengths n_1 and n_2 respectively. Eq. (47) can be hence written

$$\begin{bmatrix}\mathbf{x}_1'\\ \mathbf{x}_2'\end{bmatrix} = \begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_2\end{bmatrix} = \begin{bmatrix}\mathbf{A}_{11}(\mathbf{x}_1,\mathbf{x}_2) & \mathbf{A}_{12}(\mathbf{x}_1,\mathbf{x}_2)\\ \mathbf{A}_{21}(\mathbf{x}_1,\mathbf{x}_2) & \mathbf{A}_{22}(\mathbf{x}_1,\mathbf{x}_2)\end{bmatrix}\begin{bmatrix}\mathbf{f}_1(\mathbf{x}_1,\mathbf{x}_2,t)\\ \mathbf{f}_2(\mathbf{x}_1,\mathbf{x}_2,t)\end{bmatrix}, \tag{48}$$

where each submatrix $\mathbf{A}_{ij}$ has size $n_i \times n_j$. Depending on the relative costs of computing $\mathbf{A}_{ij}$ and $\mathbf{f}_j$, a multirate time integration strategy based on a splitting of the matrix $\mathbf{A}$ can be computationally less expensive than a standard scheme.

In the simplest case, let $\mathbf{f}(\mathbf{x}) = \mathbf{x}$ and let $\mathbf{A}$ be a constant matrix. Then (47) is a constant coefficient linear system. If $\mathbf{A}$ is dense, evaluating the right hand side, $\mathbf{Ax}$, requires $O(N^2)$ operations. Given an explicit time integration scheme, the largest stable time step of the method will depend on the eigenvalues of $\mathbf{A}$. Now suppose that we can write $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 = \begin{bmatrix}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & \mathbf{A}_{22}\end{bmatrix}, \tag{49}$$

and furthermore that $\mathbf{A}_1$ has eigenvalues with smaller magnitude than $\mathbf{A}_2$ so that a multirate strategy is appropriate. For example, in a multirate strategy where $\mathbf{A}_1\mathbf{x}$ is evaluated P times less often than $\mathbf{A}_2\mathbf{x}$, then the relative cost of evaluating the matrices is reduced from $O(PN^2)$ to $O(Pn_2^2 + 2n_1n_2 + n_1^2)$ or $O(N^2 + (P-1)n_2^2)$. If n_2 is much smaller than N, this is a savings of approximately a factor of P.

Now consider a more involved scenario with a nontrivial dependence of $\mathbf{f}$ on $\mathbf{x}$. Let us assume that $\mathbf{A}$ can be computed in $O(N^2)$ operations with a small prefactor, so that the computation of $\mathbf{A}$ is not significantly more expensive than evaluating $\mathbf{Af}$. Let us assume as well that $\mathbf{f}_2(\mathbf{x},t)$ can be computed in $O(n_2^2)$ operations with a small prefactor. However, assume that the computation of $\mathbf{f}_1(\mathbf{x},t)$ requires more significant computation. For example, suppose for some given vector $\mathbf{v}(t)$

$$\mathbf{f}_1(\mathbf{x},t) = \mathbf{A}_{11}^{-1}(\mathbf{v}(t) - \mathbf{A}_{12}\mathbf{f}_2(\mathbf{x},t)). \tag{50}$$

In Section 4.2, the evolution equations for a particular regularized Stokeslet calculation are shown to have this form. Now the evaluation of $\mathbf{f}_1(\mathbf{x},t)$ requires $O(n_1^3)$ operations for a direct evaluation of $\mathbf{A}_{11}^{-1}$, or $O(Kn_1^2)$ operations for K iterations of a Krylov subspace method as is used in the numerical examples to follow. Hence the evaluation of $\mathbf{A}(\mathbf{x})\mathbf{f}(\mathbf{x},t)$ requires $O(N^2+Kn_1^2+N)$ operations. If the matrix $\mathbf{A}$ is split as in (49), then the evaluation of $\mathbf{A}_1(\mathbf{x})\mathbf{f}(\mathbf{x},t)$ has cost $O(N^2+Kn_1^2+N)$ since it requires the evaluation of $\mathbf{f}_1(\mathbf{x},t)$. On the other hand, the cost of evaluating $\mathbf{A}_2(\mathbf{x})\mathbf{f}(\mathbf{x},t)$ is the same as that of evaluating $\mathbf{A}_{22}(\mathbf{x})\mathbf{f}_2(\mathbf{x},t)$, namely $O(n_2^2)$. The leading order term in these costs are $O(N^2+Kn_1^2)$ and $O(n_2^2)$ respectively, hence an even greater numerical efficiency will result from using an operator splitting approach if n_2 is significantly less than N, particularly when K is not small. As an example, in the numerical results presented in Section 5, the number of GMRES iterations to solve the analog of (50) is in the range of 10–20.

4. A specific MESDC example

In this section we give two specific examples of how the MESDC routine outlined in Section 3 can be utilized for the particular case of the time integration method within the method of regularized Stokeslets. We will first give a brief overview of the method of regularized Stokeslets, then outline how MESDC is implemented for the case of a rotating rod and a rigid sphere interacting in Stokes flow as well as flexible fibers in shear flow inspired by modeling the endothelial glycocalyx. In both examples, the splitting of the ODEs governing the dynamics have clear physical counterparts which help illustrate the MESDC method. These physical scenarios will also be used for numerical tests in Section 5.

4.1. Method of regularized stokeslets

The method of regularized Stokeslets, developed by Cortez [16,17], calculates the fluid velocity in Stokes flow due to a collection of regularized forces. Cutoff functions are introduced as a way to regularize a point force. The regularization removes the singular nature from the velocity field, hence the velocity can be evaluated anywhere in the fluid, including at the location of a regularized Stokeslet. After regularizing the singular velocity, there exists a linear relationship between the force at a point in space and the velocity at another point in space. Utilizing the linearity of Stokes flow, one can superimpose the regularized Stokeslets solutions to build more complicated fluid velocity solutions.

Instead of representing a point force with a delta function, as is the case in deriving a Stokeslet (see [38]), consider regularizing the force using a radially symmetric cutoff function, ϕ_ϵ. It is assumed that the cutoff function is a smooth approximation to the delta function that satisfies

$$\int \phi_\epsilon(\mathbf{x})d\mathbf{x} = 1, \tag{51}$$

$$\lim_{\epsilon\to 0}\phi_\epsilon(\mathbf{x}) = \delta(\mathbf{x}), \tag{52}$$

where ϵ is the spreading parameter that controls the extent of the distribution. This discussion will use the cutoff function

$$\phi_\epsilon(r) = \frac{15\epsilon^4}{8\pi(r^2+\epsilon^2)^{7/2}}, \tag{53}$$

where $r = |\mathbf{x}-\mathbf{x}_0|$ [16]. Now consider solving the Stokes equations with a regularized forcing term at $\mathbf{x}_0$, with $\hat{\mathbf{x}} = \mathbf{x}-\mathbf{x}_0$:

$$\mu\nabla^2\mathbf{u} = \nabla p - \mathbf{f}\phi_\epsilon(\hat{\mathbf{x}}), \tag{54}$$

$$\nabla\cdot\mathbf{u} = 0. \tag{55}$$

One can represent a solution to (54) and (55) with a regularized Stokeslet constructed from the cutoff function ϕ_ϵ [17]:

$$S_{ij}^{\phi_\epsilon} = \delta_{ij}\frac{r^2+2\epsilon^2}{(r^2+\epsilon^2)^{3/2}} + \frac{\hat{x}_i\hat{x}_j}{(r^2+\epsilon^2)^{3/2}}. \tag{56}$$

Here, $i=1,2,3$ and $j=1,2,3$ correspond to the three components of an applied force vector (indexed by j) and the three components of the resulting velocity vector (indexed by i), as prescribed below in (58). Notice that in the limit as $\epsilon\to 0$, one recovers the expression for the singular Stokeslet in three dimensions [38]:

$$\lim_{\epsilon\to 0} S_{ij}^{\phi_\epsilon} = \frac{\delta_{ij}}{r} + \frac{\hat{x}_i\hat{x}_j}{r^3} = S_{ij}. \tag{57}$$

The velocity due to a regularized Stokeslet is

$$u_i^{S,\phi_\epsilon}(\mathbf{x}) = \frac{1}{8\pi\mu}\sum_{j=1}^{3} S_{ij}^{\phi_\epsilon} f_j, \tag{58}$$

where $S_{ij}^{\phi_\epsilon}$ is the regularized Stokeslet in (56).

Eq. (58) provides a way to calculate the velocity at a location $\mathbf{x}$ due to a regularized point force located at $\mathbf{x}_0$. Now, consider calculating the velocity at a point $\mathbf{x}$ due to a collection of N regularized Stokeslets, each located at $\mathbf{x}_n$:

$$u_i^{S,\phi_\epsilon}(\mathbf{x}) = \frac{1}{8\pi\mu}\sum_{n=1}^{N}\sum_{j=1}^{3} S_{ij}^{\phi_\epsilon}(\mathbf{x},\mathbf{x}_n) f_{j,n}, \tag{59}$$

where $f_{j,n}$ represents the jth component of the force exerted at $\mathbf{x}_n$. To calculate the fluid velocity at M locations due to N regularized Stokeslets, this relationship can be expressed in matrix form:

$$\mathbf{u}(\mathbf{x}) = \mathbf{S}^{\phi_\epsilon}(\mathbf{x},\mathbf{x}_0)\mathbf{f}, \tag{60}$$

where $\mathbf{u}$ is a $3M \times 1$ vector containing the velocity components at the M locations represented by the $3M \times 1$ vector $\mathbf{x}$, $\mathbf{x}_0$ is a $3N \times 1$ vector of regularized Stokeslet locations, $\mathbf{f}$ is a $3N \times 1$ vector of force coefficients, and $\mathbf{S}^{\phi_\epsilon}$ is a $3M \times 3N$ matrix incorporating the regularized Stokeslet information. When $M = N$, the matrix $\mathbf{S}^{\phi_\epsilon}$ often can be inverted to compute the forces necessary to satisfy a given velocity condition at a collection of points. This is the basis of the method of regularized Stokeslets which will be used to model a precessing rod and a rigid sphere in Stokes flow, as discussed in Section 4.2, and a collection of flexible fibers modeling the glycocalyx in Section 4.3.

4.2. Rod and sphere

The numerical examples in this paper are motivated by current experimental fluid dynamics research being conducted at the University of North Carolina [23–25]. First, consider the case of a rigid rod precessing about its center in free space with a prescribed angular velocity and a rigid sphere moving and interacting with the fluid, as shown in Fig. 3. The rigid sphere is constructed with regularized Stokeslets that are placed within the fluid domain and connected by virtual springs to neighboring regularized Stokeslets. Regularized Stokeslets connected by springs have been implemented in other works, for example in [17] to model motile spirochetes (an order of bacteria that have a helical shape). The findings in [24] can be used as a guide to choosing parameters for the regularized Stokeslets. Specifically, the regularized Stokeslets are inset from the desired spherical surface so that the effective radius of the collection of regularized Stokeslets matches that of the desired rigid sphere and minimizes the velocity error.

In this case, it is the spring forces that are responsible for adding stiffness to the system, requiring a reduction in time step for an increase in the spring constant. Requiring the entire system to take small time steps is inefficient, motivating the development of the MESDC method. In the context of this specific example, the MESDC method separates the system into two parts, one for the velocity due to Stokeslets defining the rod and one for the velocity due to those defining the sphere.

Recall from Section 4.1, the fluid velocity at $\mathbf{x}$ due to regularized Stokeslets located at $\mathbf{x}_0$ is given by $\mathbf{u}(\mathbf{x},\mathbf{x}_0) = \mathbf{S}^{\phi_\epsilon}(\mathbf{x},\mathbf{x}_0)\mathbf{f}(\mathbf{x}_0)$, where $\mathbf{S}^{\phi_\epsilon}$ is a matrix as defined in (56). Let $\mathbf{x}_s$ represent a location on the sphere and let $\mathbf{x}_r$ be a location on the rod. Similarly, let $\mathbf{f}_s$ and $\mathbf{f}_r$ be forces located at $\mathbf{x}_s$ and $\mathbf{x}_r$, respectively. For the sake of completeness in notation, define the following:

- $\mathbf{u}_s = \mathbf{u}(\mathbf{x}_s,\cdot)$: velocity at points on the sphere due to forces at any fluid location.
- $\mathbf{u}_r = \mathbf{u}(\mathbf{x}_r,\cdot)$: velocity at points on the rod due to forces at any fluid location.

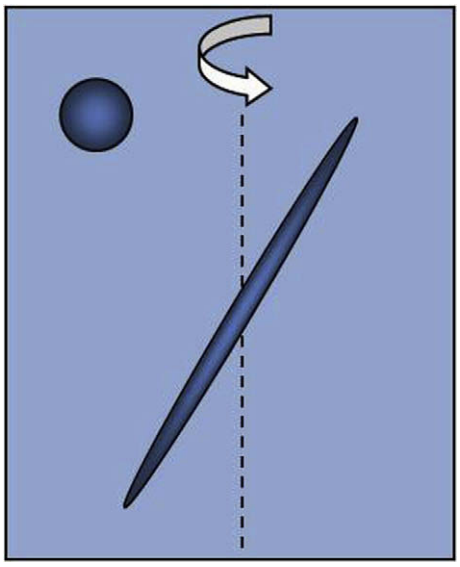

Fig. 3. A rigid rod precessing about its center with a prescribed angular velocity generates fluid flow moving a rigid sphere that also interacts with the fluid.

- $\mathbf{u}_{s,s} = \mathbf{u}(\mathbf{x}_s, \mathbf{x}_s) = \mathbf{S}_{ss}^{\phi_\epsilon}\mathbf{f}_s$: velocity at points on the sphere due to forces at the collective points on the sphere.
- $\mathbf{u}_{s,r} = \mathbf{u}(\mathbf{x}_s, \mathbf{x}_r) = \mathbf{S}_{sr}^{\phi_\epsilon}\mathbf{f}_r$: velocity at points on the sphere due to forces at the points on the rod.
- $\mathbf{u}_{r,r} = \mathbf{u}(\mathbf{x}_r, \mathbf{x}_r) = \mathbf{S}_{rr}^{\phi_\epsilon}\mathbf{f}_r$: velocity at points on the rod due to forces at the collective points on the rod.
- $\mathbf{u}_{r,s} = \mathbf{u}(\mathbf{x}_r, \mathbf{x}_s) = \mathbf{S}_{rs}^{\phi_\epsilon}\mathbf{f}_s$: velocity at points on the rod due to forces at the points on the sphere.

Consider the following steps which give a summary of how the velocity decomposition is implemented. A central idea here is that the fluid velocity at an arbitrary location can be decomposed into two components, the velocity due to regularized forces exerted at points on the rod and the velocity due to regularized forces exerted at points on the sphere:

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}(\mathbf{x}, \mathbf{x}_r) + \mathbf{u}(\mathbf{x}, \mathbf{x}_s). \tag{61}$$

Step 1. Prescribe the velocity of the rod at time t_i : $\mathbf{u}_{r,i} = \mathbf{u}_r(t_i)$.
Step 2. Use Hooke's Law to calculate the spring forces generated by the sphere locations. Let d_i^j be the length of the jth spring and d_0^j be its resting length. For an arbitrary regularized Stokeslet location on the sphere, $\mathbf{x}_s$, let m be the number of springs attached to $\mathbf{x}_s$ and let $\hat{\mathbf{e}}_j$ be a unit vector directed along the jth spring. Then the net force on the regularized Stokeslet located at $\mathbf{x}_s$ is given by:

$$\mathbf{f}_{s,i} = \sum_{j=1}^{m} -k(d_i^j - d_0^j)\hat{\mathbf{e}}_j. \tag{62}$$

Step 3. The rod and the sphere both exert forces on the fluid, so they will both contribute to the velocity on the rod at time t_i:

$$\mathbf{u}_{r,i} = \mathbf{u}_{r,r,i} + \mathbf{u}_{r,s,i}, \tag{63}$$

$$= \mathbf{S}^{\phi_{\epsilon rr,i}}\mathbf{f}_{r,i} + \mathbf{S}^{\phi_{\epsilon rs,i}}\mathbf{f}_{s,i}. \tag{64}$$

Step 4. Solve (64) for the forces exerted at the rod, $\mathbf{f}_{r,i}$, using GMRES:

$$\mathbf{f}_{r,i} = (\mathbf{S}^{\phi_{\epsilon rr,i}})^{-1}(\mathbf{u}_{r,i} - \mathbf{S}^{\phi_{\epsilon rs,i}}\mathbf{f}_{s,i}), \tag{65}$$

Notice that the form of this equation is exactly that of (50), also noting that $\mathbf{f}_{r,i}$ is the only unknown quantity. It should also be noted that the condition number of $\mathbf{S}_{rr,i}^{\phi_\epsilon}$ increases as $\frac{h}{\epsilon}$ decreases, where h represents spacing between regularized Stokeslets and ϵ is the spreading parameter. However, h and ϵ can be chosen easily to avoid ill-conditioned matrices (see [24]).
Step 5. Use the forces at the rod to calculate the fluid velocity at each location of interest on the sphere:

$$\mathbf{u}_{s,i} = \mathbf{u}_{s,r,i} + \mathbf{u}_{s,s,i}, \tag{66}$$

$$= \mathbf{S}_{sr,i}^{\phi_\epsilon}\mathbf{f}_{r,i} + \mathbf{S}_{ss,i}^{\phi_\epsilon}\mathbf{f}_{s,i}. \tag{67}$$

The MESDC method is used to temporally integrate the system utilizing this velocity decomposition. Notice that the decomposition in (66) is analogous to (15) in the discussion of MESDC in Section 3.

4.3. Flexible fibers

The motivation for a second numerical example comes from the work of Miller et al. in their study of the fluid dynamics surrounding the endothelial surface layer (ESL) [25]. The ESL includes the glycocalyx and attached proteins projecting out of endothelial cells. The glycocalyx is a brush-like structure protruding from the endothelial cells spaced in a hexagonal pattern [39,40].

The physical models studied by Miller et al. use an array of thin stationary rigid pins in Stokes flow to study the flow through glycocalyx [25]. Here, the experiment is modeled numerically as a hexagonal array of flexible structures referred to as fibers as depicted in Fig. 4. Each of the seven fibers consists of a collection of N regularized Stokeslets aligned as a triangular beam. Each regularized Stokeslet is connected to its eight nearest neighbors by Hookean springs. Background shear flow is added to this system with a no-slip plane implemented at the base of the fibers to represent a solid wall bottom.[4]

The velocity decomposition for using MESDC with the flexible fiber example distinguishes between the velocity at a particular fiber due to the forces generated by that fiber and the velocity collectively at the other fibers due to the forces generated by that particular fiber. That is, the velocity is decomposed into near field and far field contributions from each fiber, which will be treated in MESDC with small and large substeps, respectively. Let $\mathbf{x}_i$ represent the locations of the N regularized Stokeslets on the ith fiber and $\mathbf{x}_\tau$ represent tracer locations (any locations of interest in the fluid domain not occupied by a regularized Stokeslet). In a similar fashion as the rod and sphere example discussed in Section 4.2, let $\mathbf{u}_{i,j} = \mathbf{u}(\mathbf{x}_i, \mathbf{x}_j)$ represent the velocity at $\mathbf{x}_i$ due to $\mathbf{x}_j$, as given by (60). In this case, let $i = 1, \ldots, 7$ represent the indices of the fibers where

[4] Implementation of the no-slip plane requires a system of regularized image singularities in addition to regularized Stokeslets. See [24] for further information.

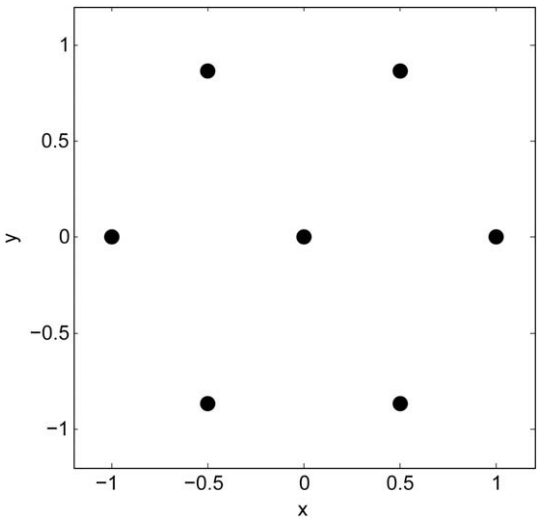

Fig. 4. Top view of seven fibers arranged in a hexagonal pattern.

the forces are exerted and let $j = 1, \ldots, 7, \tau$ represent both the fibers and the tracers where the velocity is calculated. The velocity is decomposed into near and far field contributions for use with MESDC:

$$\mathbf{u}(\mathbf{x}) = \mathbf{u}_{far}(\mathbf{x}) + \mathbf{u}_{near}(\mathbf{x}), \tag{68}$$

where $\mathbf{x}$ is the vector representing all regularized Stokeslet and tracer locations:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_7 \\ \mathbf{x}_t \end{bmatrix}. \tag{69}$$

Note that both $\mathbf{u}$ and $\mathbf{x}$ are of size $(7N + N_\tau) \times 3$, where N is the number of regularized Stokeslet on each fiber and N_τ is the number of tracer locations. The near field velocity component includes the background flow velocity, $\mathbf{u}_b$, and the velocity contributions from the ith fiber evaluated at the ith fiber, $\mathbf{u}_{i,i}$:

$$\mathbf{u}_{near} = \mathbf{u}_b + \begin{bmatrix} \mathbf{u}_{1,1} \\ \vdots \\ \mathbf{u}_{7,7} \\ 0 \end{bmatrix}. \tag{70}$$

The far field velocity component contains the velocity contributions from each fiber on the other fibers, not including itself. It also contains the velocity from all seven fibers at the tracer locations:

$$\mathbf{u}_{far} = \begin{bmatrix} \sum_{j=2}^{7} \mathbf{u}_{1,j} \\ \vdots \\ \sum_{j=1}^{6} \mathbf{u}_{7,j} \\ \sum_{j=1}^{7} \mathbf{u}_{\tau,j} \end{bmatrix}. \tag{71}$$

The velocity decomposition of $\mathbf{u}$ in (68) is analogous to the MESDC formulation in (15) where $\mathbf{u}_{far}$ is treated with a large time step and $\mathbf{u}_{near}$ is treated with a small time step. The flexible fiber model will be used as a numerical example in Section 5 along with the rod and sphere example discussed in Section 4.2.

5. Numerical tests

The goal in implementing a multirate MESDC method in place of a standard SDC method is to be able to handle stiff systems with a larger time step for the non-stiff components of the system than previously used with the SDC method. The stiff components will have similar small time steps in both MESDC and SDC treatments. If the relative cost of computing the non-stiff components is sufficiently more than those of stiff components, an overall increase in the efficiency of the method can be achieved. We demonstrate these issues on the two model problems discussed in Section 4. Much of the discussion in this section focuses on the rod and sphere example of Section 4.2, but the flexible fiber example of Section 4.3 is also discussed briefly Section 5.2.

5.1. Rod and sphere

To study the convergence and stability of MESDC in comparison to SDC, consider the previously mentioned system of a slender rigid rod precessing about its center in free space moving a fluid that contains a rigid sphere, as shown in Fig. 3 and discussed in Section 4.2. Recall, the rigid sphere is comprised of a collection of N_s regularized Stokeslets that are connected with a network of virtual springs. Each regularized Stokeslet is connected to its nearest neighbors by springs with spring constant k as well as the center of the sphere (with spring constant k_c). As the fluid moves these regularized Stokeslets, the displacement of the points generates a spring force that the singularities transfer back to the fluid. This additional collection of forces then creates an additional velocity at the rod, which no longer satisfies the velocity boundary condition we are trying to maintain on the rod. As such, a dense linear system needs to be solved at each time step to account for the change in the velocity at the rod due to the sphere.

The size of the matrix involved in recomputing the singularity strengths in this linear system is $3N_r \times 3N_r$, where N_r is the number of regularized Stokeslets comprising the rod. Thus, for large N_r, solving the linear system becomes increasingly expensive. For the numerical examples involving the rod and sphere in this section, the total number of Stokeslets $N = N_r + N_s$, where $N_r = 50$ is the number of regularized Stokeslets representing the rod and $N_s = 100$ is the number of regularized Stokeslets representing the sphere ($3N_r$ and $3N_s$ are analogous to n_1 and n_2, respectively, in the efficiency discussion in Section 3.2). While, $N_r = 50$ is not prohibitively large, it will suffice to demonstrate some properties of the MESDC method. We will also include another choice of N_r and N_s that will demonstrate MESDC efficiency in Section 5.3. The developed methods could be used to model experiments consisting of many rotating rods in future work, which would drastically increase the expense of solving the linear systems. By implementing MESDC, the possibility of being able to compute this linear solve less often while maintaining accuracy will be studied.

Unless otherwise noted, the reference solution is computed with $N_t = 512$ time steps per rod revolution for all of the numerical tests in this section. The position error is computed as the Euclidean norm of position error at mutual time steps averaged over the course of one rod revolution[5]. The rod parameters used are $N_r = 50$, rod length 2, and spreading parameter $\epsilon_r = 0.0276$. As discussed in [24], the spreading parameter varies along the length of the rod proportionally with the radius of the desired spheroid (with its maximum reaching the aforementioned value $\epsilon_r = 0.0276$). For the sphere, there are $N_s = 100$ regularized Stokeslets with spreading parameter $\epsilon_s = 0.1295$ set on a spherical surface of radius 0.4315, which is meant to correspond to a sphere of effective radius 0.5. (see [24]). The units of these length values are arbitrary, so they are adaptable to a physical scenario of interest, provided the motion can be approximated by Stokes flow. The time scale is such that the rod completes one revolution in N_t time steps.

First consider studying the convergence rate. With both MESDC and SDC, Sections 2 and 3 claim that the order of the error relates to the number of correction iterations used (n_{iter}), provided the quadrature choices are accurate enough. When $n_{iter} = 0$, the correction loop of the algorithms are not used, so the SDC and MESDC algorithms simplify to (multirate) forward Euler. As such, one would expect the error to decrease linearly with Δt. Fig. 5 shows the error versus time step (Δt) for various numbers of iterations, $n_{iter} = 0, 2, 3, 4, 5, 15$, for both SDC and MESDC. The increase in the order of accuracy with SDC iteration is apparent. Notationally, the iteration count n_{iter} begins with $n_{iter} = 2$ so that the order of the method will correspond with the n_{iter} value. For the data displayed in Fig. 5, the spring constants for the sphere are $k = 8, k_c = 0.8$.

Notice that for $n_{iter} = 15$, the convergence rate seems to approach 8 rather than 15 before leveling off as Δt approaches 0 (due to the limited precision of the quadrature weights). This is evidence of the quadrature convergence from choosing $j = 5$ Lobatto nodes in each substep where a convergence rate of $2j - 2$ limits the convergence rate from increasing beyond 8. By choosing more nodes, one would expect this convergence rate to increase. This phenomenon is also visible in Fig. 6 which shows similar data to Fig. 5, except that the spring constants are $k = k_c = 0$. Since the spring constants are 0, the results of MESDC and SDC are the same so only MESDC is displayed. Notice that for $n_{iter} = 4, 5, 15$, the convergence rate appears to be 8, rather than the anticipated 4, 5, or 15, respectively. The $n_{iter} = 15$ case is the same as discussed above, but the $n_{iter} = 4$ and $n_{iter} = 5$ cases are such that the convergence rate exceeds the expected rate. In this case the springs have no added effect to the system and the correction iterations within MESDC converge more quickly to the collocation solution [41]. Thus, the temporal error is subdominant to the quadrature error in this case.

[5] It should be noted that the error is not being compared to a system with an actual rigid sphere, rather the numerical solution with a given set of stiffness parameters.

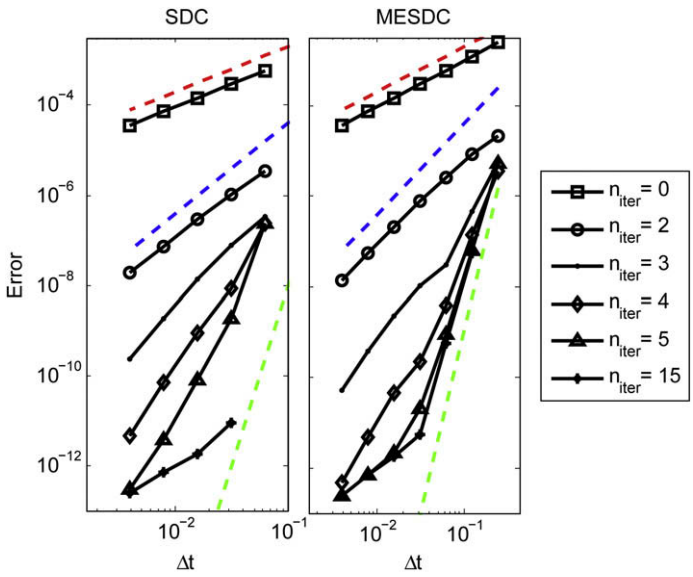

Fig. 5. Error in position versus time step using SDC and MESDC for a variety of n_{iter} values (number of correction iterations). Notice that the convergence rate increases as n_{iter} increases. The slopes of the dashed lines are 1,2, and 8. Here the spring constants are $k = 8$ and $k_c = 0.8$.

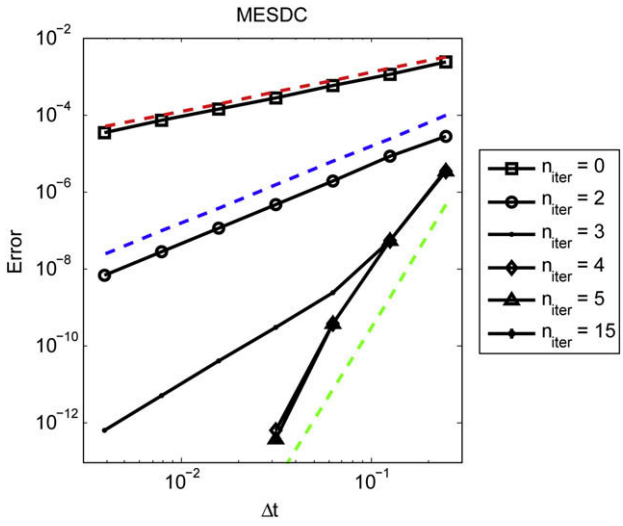

Fig. 6. Error in position versus time step using MESDC for a variety of n_{iter} values (number of correction iterations). Notice that the convergence rate increases as n_{iter} increases. The slopes of the dashed lines are 1,2, and 8. Here the spring constants are $k = k_c = 0$.

Now to discuss stability, consider the $n_{iter} = 5$ case, which should be fifth-order accurate. Consider sampling a variety of spring constants, k, allowing the spring constant of the center springs that connect the center and each point on the sphere's surface, k_c, to vary proportionally to the spring constant among nearest neighbors on the surface: $k_c = 0.1k$. Consider varying the number of time steps, N_t, and spring constants, k. For the substep discretization (see Fig. 2), $N_m = 5$ is fixed, but both $N_p = 5$ and $N_p = 10$ are considered. Fig. 7 shows the position error versus time step for four chosen spring constants, $k = 0, 8, 16, 64$, using both SDC and MESDC. Notice that for larger Δt, as k increases, the system becomes unstable. However, for fixed k, the instability occurs at a smaller Δt for the SDC case than MESDC. This indicates an increase in stability when using MESDC over SDC. For this application, the error achieved for the largest stable time step is quite small due to the high-order of accuracy of the method, hence increasing the largest stable time step allowable is of greatest importance. Also, notice that the MESDC error is consistently smaller than the SDC error for a given time step.

To view the same data in a slightly different context, consider plotting the error as a function of effective Δt rather than Δt, where the effective Δt takes into account the fact that MESDC adds another level of substeps to the time discretization. Referring to Fig. 2, letting $N_m = 5$ and $N_p = 5$ divides each time step into five substeps and each of those substeps into five

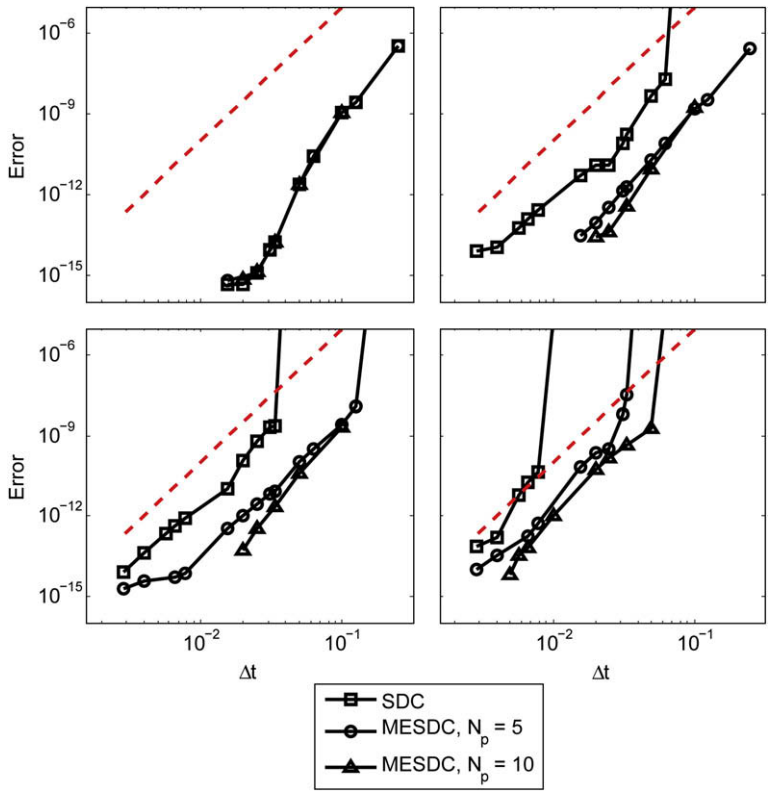

Fig. 7. Error in position versus time step using SDC and MESDC for a variety of spring constants with $n_{iter} = 5$ and $N_m = 5$ for the rod and sphere example presented in Section 4.2. From top left to bottom right plots, the spring constants are $k = 0, 8, 16, 64$ and $k_c = 0.1k$. The dashed lines have slope 5.

smaller substeps. Here, the calculation of the contribution from the sphere on the sphere is computed 25 times each SDC iteration with MESDC, as opposed to the remaining contributions which are computed 5 times per time step, where with SDC, all contributions are computed five times each iteration. Using the effective Δt compares the time step of the finest level of calculation whereas considering Δt compares the time step of the coarsest level of calculation. In this example when $N_m = N_p = 5, \Delta t_{eff} = \frac{1}{5}\Delta t$ for MESDC while $\Delta t_{eff} = \Delta t$ for SDC. Similarly, when $N_m = 5$ and $N_p = 10$ for the MESDC case, $\Delta t_{eff} = \frac{1}{10}\Delta t$. Fig. 8 shows the same data as Fig. 7, except the error is plotted against the effective time step rather than Δt. One noteworthy feature is that the instability occurs at approximately the same effective Δt value. This confirms that using a coarser time step for the non-stiff component of the system does not effect the stability of the overall method.

When shifting perspectives to consider effective Δt rather than Δt, the finest substeps are commensurate in size and the difference between SDC and MESDC is that the linear solves happen less frequently in the MESDC case than in the SDC case. Thus, this can either be viewed as a scenario where one might want to add a finer level of substeps for a particular portion of a calculation or, conversely, one might want to compute a part of a calculation on a coarser time scale gaining computational efficiency without sacrificing accuracy.

5.2. Flexible fiber model

Now consider the flexible fiber glycocalyx model as discussed in Section 4.3. To demonstrate the aforementioned relationships between MESDC and SDC, the model of seven flexible fibers was run with $N = 33$ regularized Stokeslets on each fiber, a spring constant of $k = 10$, and spreading parameter $\epsilon = 0.01$. Each fiber has length 1 and radius 0.037. The seven fibers were arranged in a hexagonal pattern where six fibers were spaced equally on a unit circle centered at the seventh fiber, as shown in Fig. 4. The background shear flow used is $\mathbf{u}(x, y, z) = [0.1z, 0, 0]$, ensuring a no-slip plane at $z = 0$. The reference solution is computed with SDC using 500 time steps per 1 time unit. Figs. 9 and 10 represent the analogous plots for the fiber case as Figs. 7 and 8 show for the rod and sphere scenario. Note again the dramatic increase in the size of the largest stable time step for MESDC methods.

5.3. Computational timings

To demonstrate the increased efficiency of using MESDC compared to SDC, timings for a specific example are included. Consider again a precessing rod and sphere, but allow the rod to be comprised of $N_r = 800$ regularized Stokeslets. This is

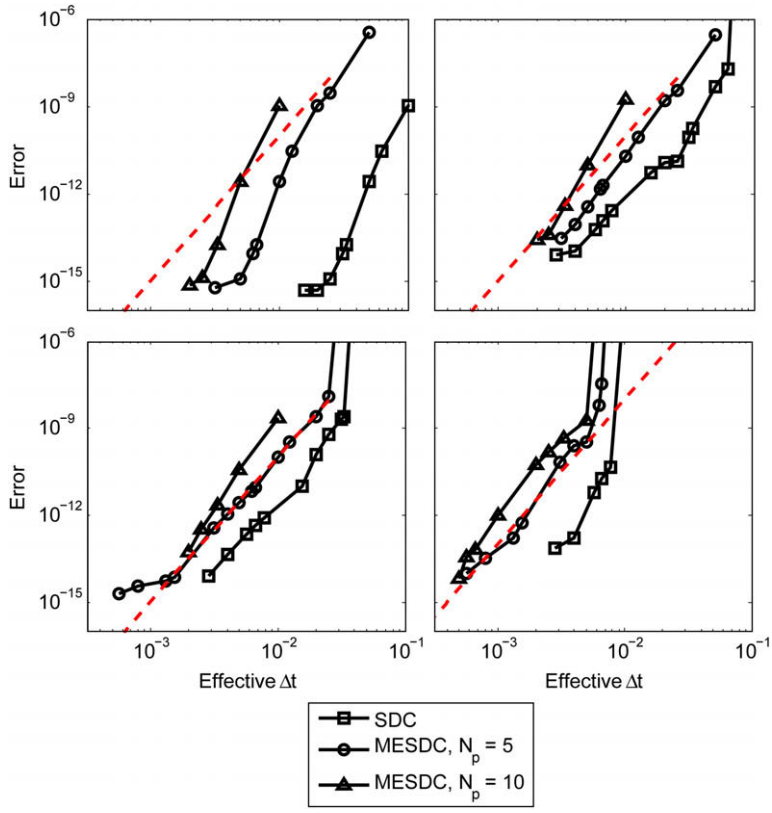

Fig. 8. Error in position versus effective time step using SDC and MESDC for a variety of spring constants with $n_{iter} = 5$ and $N_m = 5$ for the rod and sphere example presented in Section 4.2. From top left to bottom right plots, the spring constants are $k = 0, 8, 16, 64$ and $k_c = 0.1k$. The dashed lines have slope 5.

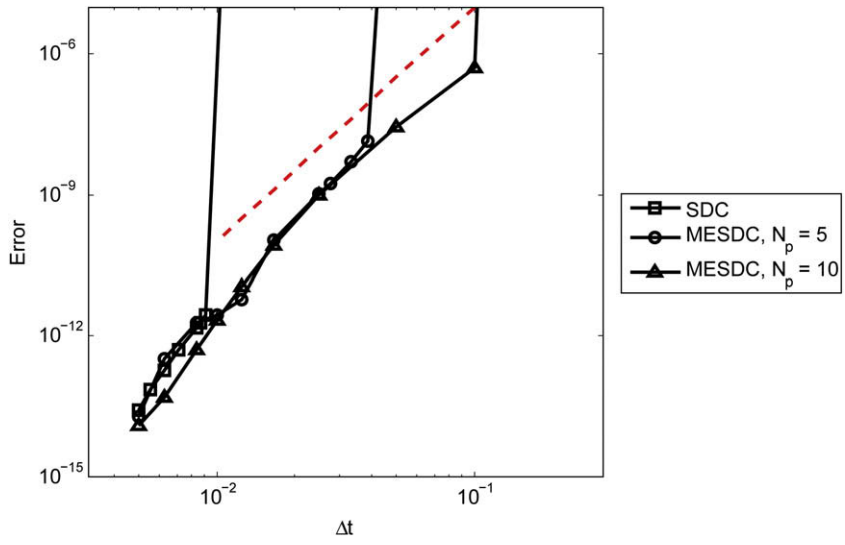

Fig. 9. Error in position versus time step using SDC and MESDC with $k = 10, n_{iter} = 5$, and $N_m = 5$ for the flexible fiber model presented in Section 4.3. The dashed line has slope 5.

significantly more Stokeslets than are used in Section 5.1, but would give an indication of the scaling of the problem when more rods are considered. Let the sphere consist of $N_s = 50$ regularized Stokeslets that are connected to their nearest neighbors with Hookean springs. Using a spring constant of $k = 64$, Table 1 shows the computational time (in seconds) for the largest stable time step for each method: SDC, MESDC with $N_p = 5$, and MESDC with $N_p = 10$. The computational time

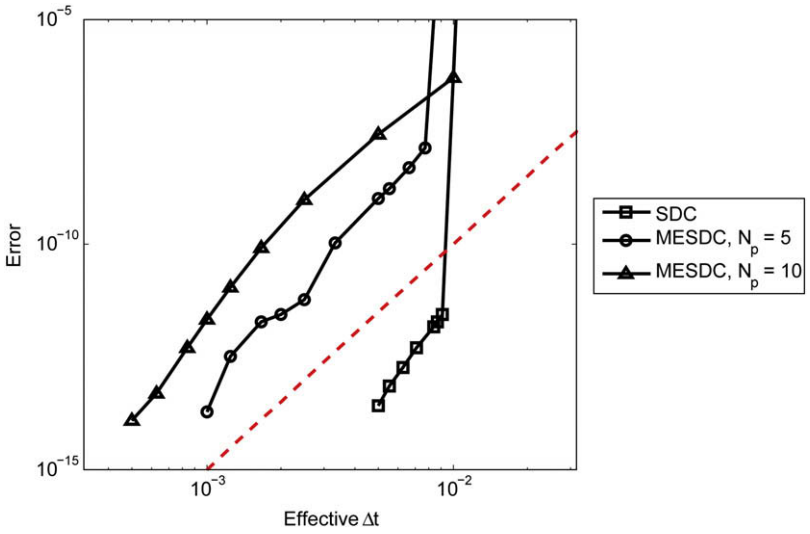

Fig. 10. Error in position versus effective time step using SDC and MESDC with $k = 10, n_{iter} = 5$, and $N_m = 5$ for the flexible fiber model presented in Section 4.3. The dashed line has slope 5.

Table 1
Computational timings for $N_r = 800$, $N_s = 50$, $N_m = 5$, and $k = 64$. The first data column shows the number of time steps per rod revolution for the largest stable time step. The last column shows the total computational time for one rod revolution using the largest stable time step. Time units are in seconds.

	Smallest number of stable time steps per rod revolution	Computational time per time step	Computational time per rod revolution
SDC	106	25.9	2743.4
MESDC, $N_p = 5$	25	26.7	667.5
MESDC, $N_p = 10$	12	28.7	344.2

per time step is an average over five time steps and the computational time per rod revolution is the product of the previous two columns. Note that the cost per time step is greater for MESDC methods, but this increase is more than compensated by the reduction in the number of time steps required.

The timings in Table 1 only give evidence for the effectiveness on a specific example, and the overall gain in efficiency for a multirate approach will depend strongly on the relative costs of computing the different pieces of the splitting of the equation. In this example, only two different time steps are used in the computation (*i.e.* rod and sphere), but splitting into more than two pieces is possible. For example, the spring forces on the ball could be further split into those from nearest neighbor connections and those well separated on the ball. Further research into the effectiveness of such a splitting is underway.

6. Summary

We have presented a new multi-explicit spectral deferred correction method that provides for increased efficiency in the explicit time integration of ordinary differential equations with multiple time scales. This method is designed for ODEs that can easily be decomposed into components with disparate levels of stiffness and would be computationally expensive to treat in a fully implicit manner. When the stiffest terms of the equation (*i.e.* those that are updated most frequently) are relatively inexpensive to evaluate, the use of coarser time steps for the less stiff components results in a substantial decrease in overall computational cost.

The motivation for investigating such temporal integration methods comes from the method of regularized Stokeslets for which the resulting ODE system has these characteristics. The MESDC method is demonstrated on two nontrivial three-dimensional Stokes flow problems modeled with regularized Stokeslets. Specifically, a slender rigid rod precessing about its center sweeping out a double cone in an incompressible fluid containing a rigid sphere and flexible fibers in shear Stokes flow are used as examples in this discussion. These examples are motivated by current experimental fluid dynamics research being conducted at the University of North Carolina [23–25]. In the rod and sphere case introduced in Section 4.2, the calculation of the Stokeslet strengths along the rod requires the solution of a large dense linear system while those on the sphere can be easily computed with a linear spring law. Hence, using the MESDC approach to apply a large time step to the evolution equation of the rod reduces the total computational cost since the expensive dense system is solved less frequently. The numerical experiments demonstrate that this reduction comes with no significant effect on the stability and accuracy of the overall temporal integration scheme.

Although the numerical experiments presented here use a decomposition of the ODE into two simple parts (corresponding to Stokeslets on the rod and those on the sphere or near and far field contributions from Stokeslets on neighboring fibers), the method is flexible enough to accommodate more complicated decompositions (*e.g.* see [3,4]). For example, in the present scenario, each Stokeslet on the sphere could move according to a force further decomposed into contributions from nearest neighbors, from the rest of the Stokeslets on the sphere, and from those on the rod. Similarly, for the glycocalyx example introduced in Section 4.3, the near field contribution could be divided up into nearest neighbors, which are treated separately from the rest of the points on the near fiber, adding another level of discretization. In some applications of immersed boundary methods, both tangential and normal forces (which are typically dependent on the local curvature) are considered, and the differing stiffness of these terms could be taken into account in designing an MESDC strategy. More elaborate adaptive decompositions are possible as long as one has at hand a reasonably efficient way of constructing the decomposition and estimating the appropriate relative size of the time step for each component. Such strategies will be investigated in future work.

References

[1] C. Gear, D. Wells, Multirate linear multistep methods, BIT Numer. Math. (4) (1984) 484–502.
[2] A. Dutt, L. Greengard, V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, BIT 40 (2) (2000) 241–266.
[3] A. Bourlioux, A.T. Layton, M.L. Minion, High-order multi-implicit spectral deferred correction methods for problems of reactive flow, J. Comput. Phys. 198 (2003) 351–376.
[4] A.T. Layton, M.L. Minion, Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics, J. Comput. Phys. 194 (2004) 697–715.
[5] C.S. Peskin, Flow patterns around heart valves: A numerical method, J. Comput. Phys. 10 (1972) 252–271.
[6] C.S. Peskin, Mathematical Aspects of Heart Physiology, Courant Institute Lecture Notes, 1975.
[7] C.S. Peskin, D.M. McQueen, A three-dimensional computational method for blood flow in the heart i. immersed elastic fibers in a viscous incompressible fluid, J. Comput. Phys. 81 (1989) 372–405.
[8] C.S. Peskin, D.M. McQueen, A general method for the computer simulation of biological systems interacting with fluids, in: C. Ellington, T. Pedley (Eds.), Biological Fluid Dynamics: Proceedings of a Meeting Held at the University of Leeds, UK, Cambridge, July 4–8, 1994.
[9] Y. Kim, C.S. Peskin, Penalty immersed boundary method for an elastic boundary with mass, Phys. Fluids 19 (5) (2007) 053103.
[10] Y. Kim, C.S. Peskin, 2-dimensional parachute simulation by the immersed boundary method, SIAM J. Sci. Comput. 28 (6) (2006) 2294–2312.
[11] L. Miller, C. Peskin, A computational fluid dynamics study of clap and fling in the smallest insects, J. Exp. Biol. 208 (2) (2005) 195–212.
[12] L. Miller, C. Peskin, A computational fluid dynamics study of clap and fling in the smallest insects, J. Exp. Biol. 208 (2005) 195–212.
[13] P.J. Atzberger, P.R. Kramer, C.S. Peskin, A stochastic immersed boundary method for fluid–structure dynamics at microscopic length scales, J. Comput. Phys. 224 (2) (2007) 1255–1292.
[14] T.T. Bringley, Analysis of the immersed boundary method for Stokes flow, Ph.D. Thesis, New York University (2008).
[15] T.T. Bringley, C.S. Peskin, Validation of a simple method for representing spheres and slender bodies in an immersed boundary method for Stokes flow on an unbounded domain, J. Comput. Phys. 227 (11) (2008) 5397–5425.
[16] R. Cortez, The method of regularized Stokeslets, SIAM J. Sci. Comput. 23 (4) (2001) 1204–1225.
[17] R. Cortez, L. Fauci, A. Medovikov, The method of regularized Stokeslets in three dimensions: Analysis, validation, and application to helical swimming, Phys. Fluids 17 (3) (2005) 031504.1–031504.14.
[18] A.A. Mayo, C.S. Peskin, An implicit numerical method for fluid dynamics problems with immersed elastic boundaries, Contemp. Math. 141 (1993) 261–277.
[19] E.P. Newren, A.L. Fogelson, R.D. Guy, R.M. Kirby, Unconditionally stable discretizations of the immersed boundary equations, J. Comput. Phys. 222 (2) (2007) 702–719.
[20] J.M. Stockie, B.R. Wetton, Analysis of stiffness in the immersed boundary method and implications for time-stepping schemes, J. Comput. Phys. 154 (1) (1999) 41–64.
[21] Y. Mori, C.S. Peskin, Implicit second-order immersed boundary methods with boundary mass, Comput. Methods Appl. Mech. Eng. 197 (2008) 2049–2067.
[22] 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. Stat. Comput. 13 (6) (1992) 1361–1376.
[23] E.L. Bouzarth, A. Brooks, R. Camassa, H. Jing, T.J. Leiterman, R.M. McLaughlin, R. Superfine, J. Toledo, L. Vicci, Epicyclic orbits in a viscous fluid about a precessing rod: Theory and experiments at the micro and macro scales, Phys. Rev. E 76 (2007) 016313.
[24] E.L. Bouzarth, Regularized singularities and spectral deferred correction methods: A mathematical study of numerically modeling stokes fluid flow, Ph.D. Thesis, The University of North Carolina at Chapel Hill 2008. URL http://dc.lib.unc.edu/u?/etd,1917.
[25] L. Miller, A. Santhanakrishnan, C. Hamlet, J.G. Cox, K.M. Leiderman, A mathematical model and physical experiments of flow through biological bristles over a range of scales, in preparation.
[26] A.T. Layton, M.L. Minion, Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations, BIT Numer. Math. 45 (2) (2005) 341–373.
[27] M.L. Minion, Semi-implicit spectral deferred correction methods for ordinary differential equations, Comm. Math. Sci. 1 (2) (2003) 471–500.
[28] U.M. Ascher, S.J. Ruuth, R.J. Spiteri, Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations, Appl. Numer. Math. 25 (1997) 151–167.
[29] S. Boscarino, Error analysis of IMEX Runge–Kutta methods derived from differential–algebraic systems, SIAM J. Numer. Anal. 45 (4) (2007) 1600–1621.
[30] C.A. Kennedy, M.H. Carpenter, Additive Runge–Kutta schemes for convection-diffusion–reaction equations, Appl. Numer. Math. 44 (2003) 139–181.
[31] M.P. Calvo, J. de Frutos, J. Novo, Linearly implicit Runge–Kutta methods for advection-reaction–diffusion equations, Appl. Numer. Math. 37 (2001) 535–549.
[32] U.M. Ascher, S.J. Ruuth, B.T.R. Wetton, Implicit–explicit methods for time-dependent PDE's, SIAM J. Numer. Anal. 32 (1995) 797–823.
[33] J. Frank, W.H. Hundsdorfer, J.G. Verwer, Stability of implicit–explicit linear multistep methods, Appl. Numer. Math. 25 (1997) 193–205.
[34] G. Akrivis, M. Crouzeix, C. Makridakis, Implicit–explicit multistep methods for Quasi linear parabolic equations, Numer. Math. 82 (1999) 521–541.
[35] J.F. Andrus, Stability of a multirate method for numerical integration of ODEs, Comput. Math. Appl. 25 (2) (1993) 3–14.
[36] M. Schlegel, O. Knoth, M. Arnold, R. Wolke, Multirate Runge–Kutta schemes for advection equations, J. Comput. Appl. Math. (2009) 345–357.
[37] A. Logg, Multi-adaptive time integration, Appl. Numer. Math. 48 (3–4) (2004) 339–354.
[38] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, Cambridge University Press, 1992.
[39] S. Weinbaum, X. Zhang, Y. Han, H. Vink, S.C. Cowin, Mechanotransduction and flow across the endothelial glycocalyx, Proc. Nat. Acad. Sci. 100 (13) (2003) 7988–7995.
[40] J.M. Squire, M. Chew, G. Nneji, C. Neal, J. Barry, C. Michel, Quasi-periodic substructure in the microvessel endothelial glycocalyx: A possible explanation for molecular filtering?, J Struct. Biol. 136 (2001) 239–255.
[41] J. Huang, J. Jia, M.L. Minion, Accelerating the convergence of spectral deferred correction methods, J. Comput. Phys. 214 (2) (2006) 633–656.