Journal of Computational Physics 306 (2016) 478–499

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Adaptive time stepping for vesicle suspensions

Bryan Quaife [a,*], George Biros [b]

[a] Department of Scientific Computing, Florida State University, Tallahassee, FL 32306, United States
[b] Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX 78712, United States

ARTICLE INFO

Article history:
Received 26 May 2014
Received in revised form 19 September 2015
Accepted 20 November 2015
Available online 2 December 2015

Keywords:
Vesicle suspensions
Adaptive time stepping
Spectral deferred correction
Particulate flows
Stokesian flows

ABSTRACT

We present an adaptive high-order accurate time-stepping numerical scheme for the flow of vesicles suspended in Stokesian fluids. Our scheme can be summarized as an approximate implicit spectral deferred correction (SDC) method. Applying a fully-implicit SDC scheme to vesicle flows is prohibitively expensive. For this reason we introduce several approximations. Our scheme is based on a semi-implicit linearized low-order time stepping method. (Our discretization is spectrally accurate in space.) We expect that the accuracy can be arbitrary-order, but our examples suffer from order reduction which limits the observed accuracy to third-order. We also use invariant properties of vesicle flows, constant area and boundary length in two dimensions, to reduce the computational cost of error estimation for adaptive time stepping. We present results in two dimensions for single-vesicle flows, constricted geometry flows, converging flows, and flows in a Couette apparatus. We experimentally demonstrate that the proposed scheme enables automatic selection of the time step size and high-order accuracy.

1. Introduction

Vesicles are deformable capsules filled with a viscous fluid. Vesicle flows refer to flow of vesicles that are suspended in a Stokesian fluid. Vesicles are differentiated from other capsules in the balance of forces in their interface. In particular, their boundary is locally inextensible (in 3D is locally incompressible). Vesicle flows find applications in many biological applications such as the simulation of biomembranes [47] and red blood cells [23,32,38,39,41].

The dynamics of a vesicle is governed by bending, tension (enforces inextensibility), hydrodynamic forces from other vesicles in the suspension, and possibly hydrodynamic forces from flow confinement boundary walls. Vesicle suspensions are modeled using the Stokes equations, a jump in the stress across each vesicle to match the interfacial forces on the membrane of the vesicle, and a no-slip boundary condition on the vesicles and the confinement walls.

A significant challenge in simulating vesicle flows is that their governing equations are stiff. One stiffness source is associated with the interfacial forces, bending forces, and inextensibility related tension forces (these depend mainly on the curvature of the vesicle). Another stiffness source is the hydrodynamic interaction between vesicles (these depend on the minimum distance between vesicles). A third stiffness source is the hydrodynamic interaction between vesicles and external boundary walls (these depend on the minimum distance between the vesicles and the wall). Therefore, to resolve the dynamics of a vesicle, it may be necessary to take a small time step when the vesicle has regions of large curvature, or when a vesicle approaches a boundary wall, or approaches another vesicle. However, a large time step may be taken when

* Corresponding author.
E-mail addresses: bquaife@fsu.edu (B. Quaife), gbiros@acm.org (G. Biros).

http://dx.doi.org/10.1016/j.jcp.2015.11.050
0021-9991/

a vesicle has a smooth boundary and is separated from all other vesicles and the boundary walls. These considerations necessitate that we use an adaptive time-stepping scheme, mostly for robustness of the code and to remove the need to manually select a time step size.

Another challenge in simulating vesicle flows is that maintaining good accuracy for long horizons requires a great number of time steps when using a low-order time-stepping method. One remedy to accumulating error is to use artificial forces and algorithms that correct each vesicle's area and length (these quantities are conserved by the physics) [1,4,5,13]. However, these approaches may not successfully stabilize the dynamics if too large of a time step is taken. Alternatively, higher-order methods such as diagonally-implicit Runge–Kutta (DIRK) or implicit–explicit (IMEX) multistep can mitigate error accumulation over long simulation times. While DIRK is high-order, we are unaware of a formulation for a set of integro-differential equations, such as the ones that vesicle suspensions satisfy. Moreover, when applying many DIRK methods to stiff algebraic-differential equations, order reduction is often present [8]. While order reduction of DIRK methods can be reduced, we choose to develop high-order accurate methods with time-stepping methods that we have successfully used in our previous work: constant time step size IMEX multistep methods. Unfortunately, unless IMEX-Euler is used, these become unstable if the time step size is changed too quickly [22]. Therefore, we will be developing high-order solutions by iteratively applying an IMEX-Euler method.

In summary, the design goals for a time-stepping scheme for vesicle flows are to address stiffness with reasonable computational costs, allow for adaptivity, and enable high-order time marching (at least second-order). In the past (in other groups as well as our group), methodologies have addressed parts of the design goals above, but no method, to our knowledge, addresses all of them. In this paper, we propose a scheme that provides this capability.

Summary of the method and contributions. To address stiffness, we use our recent work in which wall–vesicle, vesicle–vesicle, and bending and tension self-interactions are treated implicitly [42]. There we used backward difference formulas (BDFs) with constant time step sizes. As we discussed, it is possible to extend BDF with adaptive time-stepping, but, since BDFs require the solution from multiple previous time step sizes, they become quite difficult to use in practice [22]. We develop a new high-order method that uses spectral deferred correction (SDC) to iteratively increase the order of a first-order time integrator. By introducing SDC, we are able to iteratively construct high-order solutions that only require the solution from the previous time step.

In addition, we propose a time-stepping error estimation specific to vesicle flows. The incompressibility and inextensibility conditions impose that two-dimensional vesicles preserve both their enclosed area and total length. Thus, errors in area enclosed by a vesicle and errors in its perimeter can be used to estimate the local truncation error. In this way we avoid forming multiple expensive solutions to estimate the local truncation error. We numerically verify that this estimate is valid by comparing it to the truncation error of the vesicle position.

Our contributions are summarized below.

- We propose an SDC formulation to construct high-order solutions of an integro-differential-algebraic equation that governs vesicle suspensions.
- We propose an adaptive time-stepping method that uses conserved quantities to estimate the local truncation error, therefore not requiring multiple numerical solutions to form this estimate.
- We conduct numerical experiments for several different flows involving multiple vesicles, confined flows, and long time horizons that demonstrate the behavior of our scheme.

Related work. There is a rich literature on numerical methods for Stokesian particulate flows. We only discuss some representative time-stepping methods for vesicle flows and we omit details on the spatial discretization. Most methods for vesicle flows are based on integral equation formulations [43,44,46,49,54–56,59,60], but other formulations based on stencil discretizations also exist [6,19,33,34].

The time-stepping schemes used for vesicle flows can be grouped in three classes of methods. The first class is fully explicit methods, which often include a penalty formulation for the inextensibility constraint [27,33,34,46,49,54]. These are relatively easy to implement but a small time step is necessitated by the bending stiffness. Another class of methods treats self-interactions using linear semi-implicit time-stepping [44,55,59,60]. This addresses one main source of stiffness but they are fragile. No adaptive or higher than second-order scheme have been reported for these schemes. Finally, a third class of methods treat all the vesicle interactions using linearization and semi-implicit time-stepping. We are aware of two papers in this direction. In [60] a first-order backward Euler scheme is used (with no adaptivity). As we mentioned in our own work [42,45], we used a BDF scheme for vesicle flows that treats all interactions implicitly, but again, it is not adaptive and is accurate only up to second-order.

Finally, we briefly review the literature on SDC methods. SDC was first introduced by Dutt et al. [20]. SDC has been applied to partial differential equations [9,18,21,31,36,50,51], differential algebraic equations [12,30], and integro-differential equations [29], but not to vesicle flows. In [37], Minion was the first to apply SDC to implicit–explicit (IMEX) methods by studying the problem $\dot{\mathbf{x}}(t) = F_E(\mathbf{x}, t) + F_I(\mathbf{x}, t)$ where F_E is non-stiff and F_I is stiff. Unfortunately, our governing equations do not exhibit an additive splitting as non-stiff and stiff terms. Moreover, IMEX SDC often suffers from order reduction, meaning that a very small time step is required before the asymptotic convergence rate is achieved. We will see similar behavior in our work.

Table 1
Index of frequently used notation.

Symbol	Definition
N	Number of points per vesicle
M	Number of vesicles
N_t	Number of time steps
γ_k	Boundary of vesicle k
$\mathbf{x}_k(s,t)$	Parameterization of γ_k at time t and parameterized in arclength s
$\sigma_k(s,t)$	Tension of vesicle k at time t and parameterized in arclength s
Γ	Boundary of the confined geometry
$\mathcal{B}(\mathbf{x}_k)$	Bending operator due to vesicle k
$\mathcal{T}(\mathbf{x}_k)$	Tension operator due to vesicle k
$\mathrm{Div}(\mathbf{x}_k)$	Surface divergence operator due to vesicle k
$\mathcal{S}(\mathbf{x}_j,\mathbf{x}_k)$	Single-layer potential due to vesicle k and evaluated on vesicle j
$\mathcal{D}(\mathbf{x}_j,\Gamma)$	Double-layer potential due to Γ and evaluated on vesicle j
$\mathbf{v}_\infty(\mathbf{x}_j)$	Background velocity due to a far-field condition
$\mathbf{v}(\mathbf{x}_j;\mathbf{x}_k)$	Velocity of vesicle j due to hydrodynamic forces from vesicle k
$\mathbf{r}(t;\tilde{\mathbf{x}})$	Residual of equation (8) due to the provisional solution $\tilde{\mathbf{x}}$
p	Number of Gauss–Lobatto points used to approximate $\mathbf{r}$
$\mathbf{e}_{\mathbf{x}_j}$	Error between the exact vesicle position and the provisional position $\tilde{\mathbf{x}}_j$
e_{σ_j}	Error between the exact vesicle tension and the provisional tension $\tilde{\sigma}_j$
$A(t), L(t)$	Area and length of the vesicles at time t
e_A, e_L	Relative error in area and length
ϵ	Desired final tolerance for the area and length of the vesicles
$\beta_{\mathrm{up}} \geq 1$	Maximum rate that the time step is increased per time step
$\beta_{\mathrm{down}} \leq 1$	Minimum rate that the time step is decreased per time step
$\alpha \leq 1$	Multiplicative safety factor for the new time step size

Algorithmically, SDC has been accelerated using several strategies. SDC can be parallelized by either assigning each iteration of SDC to its own processor [15,17,18,40], or by embedding SDC in a parareal or multigrid framework [21,36,50,51]. We anticipate that similar parallelization strategies are possible for vesicle suspensions, but we do not report results at this time. Next, high-order (greater than first-order) methods can be used at each SDC iteration, and this is analyzed in [16] for the initial value problem $y' = f(y,t)$. Finally, by interpreting SDC as an iterative method that converges to the collocation scheme (a fully-implicit Runge–Kutta discretization), several methods of accelerating the convergence have been investigated [11,12,21,28,30,31,36,50–52,57,58]. However, as a first step towards applying SDC to vesicle suspensions, we investigate the use of a first-order method applied to a predetermined number of SDC iterations.

Limitations. The novelty of this work is to introduce a high-order adaptive time stepper. This addresses one of the limitations form our previous work [42], but also introduces new limitations.

- We cannot provide a proof on the convergence order of our scheme. We will see in Section 5 that each SDC iteration reduces the error significantly. However, when large numbers of SDC iterations are used, the asymptotic convergence rates are unclear. Many other groups have observed similar order reduction, and several methods of reducing its effect include Krylov deferred corrections [11,12,28,30,31], multilevel SDC [21,36,50,51], and diagonally implicit Runge–Kutta sweeps [57]. Understanding the asymptotic rates of convergence, and resolving the observed order reduction, could be used to further optimize the method for choosing optimal time step sizes.
- Currently, each vesicle must use the same time step size. Therefore, in flows with multiple vesicles, the time step size is controlled by the vesicle requiring the smallest time step. This can result in the actual global error being much less than the desired error (see *Couette* example). Simulations could be accelerated if each vesicle had its own time step size, resulting in a multirate time integrator.

Let us remark that we do not use adaptivity in space.

Outline of the paper. In Section 2, we briefly discuss SDC for initial value problems (IVPs), and then discuss how to extend SDC to vesicle suspensions. In Section 3, we discretize the differential and integral equations arising in the SDC framework, discuss preconditioning, and provide complexity estimates. Section 4 discusses our adaptive time-stepping strategy, and numerical results are presented in Section 5. Finally, we make concluding remarks in Section 6, and we reserve a more in depth formulation of our governing equations in the SDC framework in Appendix A.

Notation. In Table 1, we summarize the main notation used in this paper.

2. Formulation

In this section, we briefly summarize the SDC formulation for IVPs and then extend SDC to an integro-differential equation that governs vesicle dynamics. We first discuss the theory of SDC and then describe its numerical implementation in Section 3.

2.1. Spectral deferred correction

In its original development [20], SDC iteratively constructed a high-order solution of the IVP

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, t), \quad t \in [0, T], \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{1}$$

While classical deferred correction methods discretize the time derivative in (1), SDC uses a Picard integral to avoid unstable numerical differentiation. Equation (1) is reformulated as

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t f(\mathbf{x}, \tau) d\tau, \quad t \in [0, T]. \tag{2}$$

A time stepping scheme is applied to compute a numerical solution at p quadrature points in $[t_m, t_{m+1}]$. By interpolating at these quadrature points, a provisional solution $\tilde{\mathbf{x}}(t)$ is generated. Given this provisional solution $\tilde{\mathbf{x}}(t)$ of (2), the residual is defined as

$$\mathbf{r}(t; \tilde{\mathbf{x}}) = \mathbf{x}(t_m) - \tilde{\mathbf{x}}(t) + \int_{t_m}^t f(\tilde{\mathbf{x}}, \tau) d\tau, \quad t \in [t_m, t_{m+1}]. \tag{3}$$

The provisional solution at the p quadrature points is then used to approximate the integral in (3) with pth-order accuracy. The error $\mathbf{e} = \mathbf{x} - \tilde{\mathbf{x}}$ satisfies

$$\mathbf{e}(t) = \mathbf{r}(t; \tilde{\mathbf{x}}) + \int_{t_m}^t (f(\tilde{\mathbf{x}} + \mathbf{e}, \tau) - f(\tilde{\mathbf{x}}, \tau)) d\tau, \quad t \in [t_m, t_{m+1}], \tag{4}$$

and an approximation $\tilde{\mathbf{e}}$ of $\mathbf{e}$ is computed, generally using the same numerical method used to compute $\tilde{\mathbf{x}}$. Finally, the provisional solution is updated to $\tilde{\mathbf{x}} + \tilde{\mathbf{e}}$ and the procedure is repeated as many times as desired. The process of forming the residual $\mathbf{r}$, approximating the error $\mathbf{e}$, and updating the provisional solution is referred to as an SDC iteration or sweep. We note that a time stepping scheme forms a discrete provisional solution which does not have to be interpolated; instead, one forms discrete values at quadrature points which are then used to approximate the residual $\mathbf{r}$.

The accuracy of SDC depends on the discretization of (2) and (4) and the accuracy of the quadrature rule used to estimate $\mathbf{r}(t; \tilde{\mathbf{x}})$. An abstract error analysis for applying deferred correction methods to the operator equation $F(y) = 0$ has been performed in [7,35,48,53]. In [26], this abstract framework was applied to (1) and the main result is stated in Theorem 4.2. Here we formulate the result in terms of first-order corrections, which we will be using.

Proposition 2.1. *Let* $\mathbf{x}$ *be the unique solution of* (1)*, and* $\tilde{\mathbf{x}}$ *be an order* k *approximation of* $\mathbf{x}$ *formed with a stable time integrator meaning that*

$$\|\tilde{\mathbf{x}}(T) - \mathbf{x}(T)\| = \mathcal{O}(\Delta t^k),$$

where the constant depends only on the derivatives of f *and on* T*, but not on* Δt*. Assume the residual* (3) *is computed exactly. If the exact error* $\mathbf{e}$ *satisfying* (4) *is approximated with a first-order solution* $\tilde{\mathbf{e}}$*, then*

$$\|\tilde{\mathbf{e}}(T) + \tilde{\mathbf{x}}(T) - \mathbf{x}(T)\| = \mathcal{O}(\Delta t^{k+1}).$$

However, since $\mathbf{r}$ *is approximated with a quadrature rule, the asymptotic error is*

$$\|\tilde{\mathbf{e}}(T) + \tilde{\mathbf{x}}(T) - \mathbf{x}(T)\| = \mathcal{O}(\Delta t^{\min(k+1, \ell)}),$$

where ℓ *is the quadrature's order of accuracy.*

Proposition 2.1 tells us that by estimating the error $\mathbf{e}$ with a first-order method, which is the only order we will consider, the order of accuracy is increased by one, with the constraint that this convergence is limited by the accuracy of the quadrature rule for approximating (3). However, the theorem states nothing about the stability of SDC. In [20], the authors consider three discretizations of (2) and (4): fully explicit, fully implicit, and linear combinations of the two. We do not consider explicit methods since the governing equations are stiff. The other two methods could be used for vesicle suspensions, but both would require solving non-linear equations.

We have successfully used a variant of IMEX methods for vesicle suspensions [42,44,55], and we couple these methods with SDC in this work. IMEX methods [3] are a family of time integrators that treat some terms (generally, linear) implicitly and other terms explicitly. IMEX methods for additive splittings $\dot{\mathbf{x}}(t) = F_{\text{EXPLICIT}}(\mathbf{x}, t) + F_{\text{IMPLICIT}}(\mathbf{x}, t) = F_E(\mathbf{x}, t) + F_I(\mathbf{x}, t)$ of (1) were first applied to SDC by Minion [37]. We summarize the numerical results from [37] since their behavior resembles results that we will observe. First, the Van der Pol oscillator is considered in a non-stiff, mildly stiff, and stiff regime, and

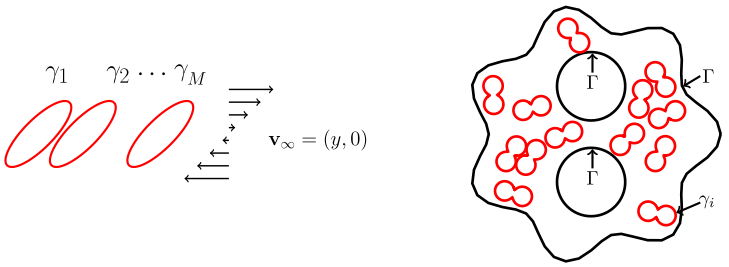

Fig. 1. Two typical vesicle suspensions. Left: M vesicles are submerged in an unbounded shear flow. Right: 13 vesicles are in a bounded domain. In the right configuration, the flow is driven by Dirichlet boundary conditions on Γ.

the number of SDC iterations ranges from two to six. In the non-stiff regime, the error behaves according to Proposition 2.1. In the mildly stiff case, the correct asymptotic result is observed, but not until Δt is much smaller. In the stiff regime, the convergence behavior differs considerably from the formal error. This behavior is attributed to order reduction which is further analyzed. The author proceeds to claim that "the correct asymptotic convergence rates would be observed given sufficiently small Δt; however, this is not the relevant issue in most applications".

An alternative interpretation of SDC which is often useful is as an iterative method that is converging to the solution of the fully-implicit Gaussian collocation scheme of (2). By using this interpretation, Krylov deferred correction methods [11,12,28,30,31] can be applied, the stability and convergence properties of SDC can be analyzed and improved [57], and inexact and multilevel approximations can be made [21,36,50–52]. These methods have been shown to accelerate SDC convergence. However, in this first work on applying SDC to vesicle suspensions, we focus on using the simplest algorithm: we apply a preselected number of SDC iterations to a set of quadrature points which remain fixed at all the SDC iterations.

2.2. SDC for vesicle suspensions

Let $\{\gamma_j\}_{j=1}^M$ be a collection of vesicles parameterized by $\mathbf{x}_j$ and with tension σ_j (Fig. 1). We use the integro-differential equation derived in [55]. It expresses the balance of the bending and tension forces of the vesicle with the stress jump of the fluid across the vesicle membrane and is augmented by a constraint to enforce the inextensibility condition. In detail, the equations that govern the evolution of the vesicle suspensions is an equation for the velocity of the interface (7) and the vesicle surface inextensibility condition (6). The key term in these two equations is $\mathbf{v}(\mathbf{x}_j;\mathbf{x}_k)$—the velocity of vesicle j due to the hydrodynamic forces from vesicle k. This term is given by

$$\mathbf{v}(\mathbf{x}_j;\mathbf{x}_k) = \mathbf{v}_\infty(\mathbf{x}_j)\delta_{j,k} + \mathcal{S}(\mathbf{x}_j,\mathbf{x}_k)(-\mathcal{B}(\mathbf{x}_k)(\mathbf{x}_k) + \mathcal{T}(\mathbf{x}_k)(\sigma_k)), \tag{5}$$

where $\delta_{j,k}$ is the Kronecker delta function,

$$\mathcal{S}(\mathbf{x}_j,\mathbf{x}_k)(\mathbf{f}) = \frac{1}{4\pi}\int_{\gamma_k}\left(-\log\rho + \frac{(\mathbf{x}_j-\mathbf{x}_k)\otimes(\mathbf{x}_j-\mathbf{x}_k)}{\rho^2}\right)\mathbf{f}(\mathbf{x}_k)ds_k,$$

$$\rho = \|\mathbf{x}_j - \mathbf{x}_k\|,$$

$$\mathcal{B}(\mathbf{x}_k)(\mathbf{f}) = \frac{d^4\mathbf{f}}{ds_k^4},$$

$$\mathcal{T}(\mathbf{x}_k)(\sigma_k) = \frac{d}{ds_k}\left(\sigma_k\frac{d\mathbf{x}_k}{ds_k}\right),$$

s_k is the arclength of γ_k, and $\mathbf{v}_\infty$ is the background velocity (unconfined flows) or the velocity due to solid walls (confined flows). In the case of confined flows, we use the double-layer potential of an unknown density function $\boldsymbol{\eta}$ defined on the boundary of the solid walls. The extra equation comes from a non-slip boundary condition on the solid walls and the details are presented in [44]. We point out that $\mathbf{v}$ is not symmetric, meaning that $\mathbf{v}(\mathbf{x}_j;\mathbf{x}_k) \neq \mathbf{v}(\mathbf{x}_k;\mathbf{x}_j)$ for all $j \neq k$, and $\mathcal{S}$, $\mathcal{B}$, $\mathcal{T}$ are all linear in their second argument.

The notation we are using is chosen so that terms such as $\mathcal{B}(\mathbf{x}_k^{m+1})(\mathbf{x}_k^{m+1})$ can be approximated as

$$\mathcal{B}(\mathbf{x}_k^m)(\mathbf{x}_k^{m+1}) = \frac{d^4}{ds_k^4}\mathbf{x}_k^{m+1}.$$

The tension σ_j acts as a Lagrange multiplier to satisfy the inextensibility constraint

$$\mathrm{Div}(\mathbf{x}_j)\left(\sum_{k=1}^M \mathbf{v}(\mathbf{x}_j;\mathbf{x}_k)\right) = 0, \tag{6}$$

where

$$\mathrm{Div}(\mathbf{x}_j)(\mathbf{f}) = \frac{d\mathbf{x}_j}{ds_j} \cdot \frac{d\mathbf{f}}{ds_j},$$

which is also linear in its second argument. Equation (6) can be eliminated using the Schur complement of the tension to write σ_j in terms of the positions $\mathbf{x}_k$, $k = 1, \ldots, M$. Then, the vesicle velocity (5) can be written entirely in terms of $\mathbf{x}_j$ and $\mathbf{x}_k$, and the no-slip boundary condition of vesicle j gives

$$\frac{d\mathbf{x}_j}{dt} = \sum_{k=1}^{M} \mathbf{v}(\mathbf{x}_j; \mathbf{x}_k), \quad j = 1, \ldots, M. \tag{7}$$

The formulation (7) is easiest to present our numerical methods, but we in fact do not eliminate the tension in our implementation. The resulting changes to the SDC formulation are presented in Appendix A.

Following the SDC method, given the solution of (7) at time t_m, the solution at some later time t satisfies

$$\mathbf{x}_j(t) = \mathbf{x}_j(t_m) + \int_{t_m}^{t} \sum_{k=1}^{M} \mathbf{v}(\mathbf{x}_j; \mathbf{x}_k) d\tau, \quad t \in [t_m, t_{m+1}]. \tag{8}$$

A provisional solution $\tilde{\mathbf{x}}$ is formed at a set of quadrature nodes in $[t_m, t_{m+1}]$ by using a time stepping method that we describe in Section 3.4. By interpolating this discrete solution (which is not done in practice), we can define the residual

$$\mathbf{r}_j(t; \tilde{\mathbf{x}}) = \mathbf{x}_j(t_m) - \tilde{\mathbf{x}}_j(t) + \int_{t_m}^{t} \sum_{k=1}^{M} \mathbf{v}(\tilde{\mathbf{x}}_j; \tilde{\mathbf{x}}_k) d\tau, \quad t \in [t_m, t_{m+1}], \tag{9}$$

which is approximated by a quadrature formula defined in Section 3.3. Then, the error $\mathbf{e}_{\mathbf{x}_j} = \mathbf{x}_j - \tilde{\mathbf{x}}_j$ satisfies

$$\mathbf{e}_{\mathbf{x}_j}(t) = \mathbf{x}_j(t_m) - \tilde{\mathbf{x}}_j(t) + \int_{t_m}^{t} \sum_{k=1}^{M} \mathbf{v}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}; \tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) d\tau, \quad t \in [t_m, t_{m+1}],$$

which we write using the residual $\mathbf{r}$ as

$$\mathbf{e}_{\mathbf{x}_j}(t) = \mathbf{r}_j(t; \tilde{\mathbf{x}}) + \int_{t_m}^{t} \sum_{k=1}^{M} (\mathbf{v}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}; \tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) - \mathbf{v}(\tilde{\mathbf{x}}_j; \tilde{\mathbf{x}}_k)) d\tau, \quad t \in [t_m, t_{m+1}]. \tag{10}$$

Then, the solution of (10) is approximated by a function $\tilde{\mathbf{e}}_{\mathbf{x}}$ using the same time stepping method and interpolation scheme used to form $\tilde{\mathbf{x}}$. Finally, we update the new provisional solution as $\tilde{\mathbf{x}} + \tilde{\mathbf{e}}_{\mathbf{x}}$. Again, this procedure of computing the residual (9), numerically solving (10) for the error, and updating the provisional solution is referred to as an SDC iteration. Assuming that we take a prediction step followed by n_{sdc} first-order SDC iterations, and ℓ is the quadrature error for computing the residual $\mathbf{r}$, from Proposition 2.1, we expect that the asymptotic rate of convergence is $\min(n_{\mathrm{sdc}} + 1, \ell)$. However, we will see that order reduction is present and this asymptotic error is difficult to achieve unless $n_{\mathrm{sdc}} \leq 2$. This has been analyzed in [57]; in particular, it is shown that order reduction for SDC is present in differential-algebraic problems and problems with a scale separation between non-stiff and highly stiff components. Both of these characteristics are present in our governing equations. We plan to introduce more sophisticated algorithms to reduce order reduction in future work.

3. Numerical scheme

In this section, we present our numerical scheme for solving (8), (9), and (10). Evaluating $\mathbf{v}$ involves layer potentials and Fourier differentiation, and for concentrated suspensions, this can be costly, especially when this work is extended to three dimensions. Moreover, high-order derivatives in the governing equations introduce stiffness, and this can result in order reduction. In this first implementation of SDC for vesicle suspensions, we opt to use the following strategy: we fix the number of SDC iterations and apply our previous first-order semi-implicit time integrator [42] to (8) and (10), and use standard quadrature to compute the residual defined in (9). In addition, there are several approximations when comparing our discretization of (10) with standard methods of discretizing (4). More expensive discretizations are possible, but we found that they did not improve the overall accuracy of the method. We also discuss how the block-diagonal preconditioner outlined in [42] is applied. We conclude with estimates of the overall work per time step as a function of the number of vesicles and the number of points per vesicle.

3.1. Spatial discretization

Following our previous work [42,44,55], we use a Lagrangian formulation by marking each vesicle with N tracker points. Since vesicles are modeled as smooth closed curves, all the derivatives are computed spectrally using the fast Fourier transform (FFT). Near-singular integrals are handled using the near-singular integration scheme outlined in [42]. Finally, we use Alpert's high-order Gauss-trapezoid quadrature rule [2] with accuracy $\mathcal{O}(h^8 \log h)$ to evaluate the single-layer potential, and the trapezoid rule for the double-layer potential which has spectral accuracy since its kernel is smooth.

3.2. Temporal discretization

A fully explicit discretization of (7) results in a stiff system for multiple reasons. A stiffness analysis in [55] reveals that the leading sources of stiffness, and the corresponding time step restrictions, are:

- $\mathcal{S}(\mathbf{x}_j, \mathbf{x}_j)\mathcal{B}(\mathbf{x}_j)(\mathbf{x}_j)$ (self-hydrodynamic bending force) – $\Delta t \sim \Delta s^3$;
- $\mathcal{S}(\mathbf{x}_j, \mathbf{x}_j)\mathcal{T}(\mathbf{x}_j)(\mathbf{x}_j)$ (self-hydrodynamic tension force) – $\Delta t \sim \Delta s$;
- $\mathrm{Div}(\mathbf{x}_j)(\mathbf{x}_j)$ (self-inextensibility force) – $\Delta t \sim \Delta s$;
- $\mathcal{S}(\mathbf{x}_j, \mathbf{x}_k)(\mathcal{B}(\mathbf{x}_k)(\mathbf{x}_k) + \mathcal{T}(\mathbf{x}_k)(\sigma_k))$, $j \neq k$ (inter-vesicle hydrodynamic forces) – depends on the inter-vesicle distance.

The leading sources of stiffness result from the intra-vesicle interactions; but, for concentrated suspensions, inter-vesicle interactions become significant and introduce stiffness. In future work, we plan to use spectral analysis to analyze the stiffness due to inter-vesicle interactions. In particular, we hope to find a relationship between the stable time step size and the distance between vesicles.

To address these multiple sources of stiffness, we use a variant of IMEX [3] time integrators. IMEX methods were developed for the problem $\dot{\mathbf{x}}(t) = F_E(\mathbf{x}) + F_I(\mathbf{x})$, where F_E is non-stiff and is treated explicitly, and F_I is stiff and is treated implicitly. For problems with this additive splitting, the family of time integrators is

$$\frac{\beta \mathbf{x}^{m+1} - \mathbf{x}^0}{\Delta t} = F_E(\mathbf{x}^e) + F_I(\mathbf{x}^{m+1}),$$

where $\mathbf{x}^0$ and $\mathbf{x}^e$ are linear combinations of previous time steps and $\beta > 0$. Unfortunately, (7) does not have an additive splitting between stiff and non-stiff terms. However, we have observed first- and second-order convergence [42] for the time integrator

$$\frac{\beta \mathbf{x}_j^{m+1} - \mathbf{x}_j^0}{\Delta t} = \sum_{k=1}^{M} \mathbf{v}(\mathbf{x}_j^e; \mathbf{x}_k^{m+1}), \quad j = 1, \ldots, M,$$

where

$$\mathbf{v}(\mathbf{x}_j^e; \mathbf{x}_k^{m+1}) = \mathcal{S}(\mathbf{x}_j^e, \mathbf{x}_k^e)(-\mathcal{B}(\mathbf{x}_k^e)(\mathbf{x}_k^{m+1}) + \mathcal{T}(\mathbf{x}_k^e)(\sigma_k^{m+1})).$$

Note that in this formulation, it is the operators involved in $\mathbf{v}$, such as the bending $\mathcal{B}(\mathbf{x}_k)$ and the single-layer potential $\mathcal{S}(\mathbf{x}_j, \mathbf{x}_k)$, that are discretized explicitly at $\mathbf{x}^e$. In line with our previous work, we use the first-order method given by $\beta = 1$, $\mathbf{x}^0 = \mathbf{x}^m$, and $\mathbf{x}^e = \mathbf{x}^m$, and the second-order backward difference formula (BDF) given by $\beta = 3/2$, $\mathbf{x}^0 = 2\mathbf{x}^m - 1/2\mathbf{x}^{m-1}$, and $\mathbf{x}^e = 2\mathbf{x}^m - \mathbf{x}^{m-1}$.

3.3. Quadrature formula

SDC requires a quadrature formula to approximate (9), the residual of the Picard integral. The quadrature rule affects the accuracy of SDC (Proposition 2.1). However, the stability of SDC is also effected by the quadrature rule, and this has been investigated in depth in [9]. In order to achieve the highest possible accuracy, we could use Gaussian nodes; however, in order to avoid extrapolation, we would like both endpoints to be quadrature nodes. Therefore, we use Gauss–Lobatto points, $t_m = t_1 < \cdots < t_p = t_{m+1}$, since they include both endpoints and have successfully been used by other groups [9,10,37], and have an order of accuracy of $2p - 2$. Alternatively, we could use Radau quadrature formula which include the right endpoint, but not the left endpoint.

For computational efficiency, the accuracy of the quadrature formula should not exceed the expected rate of convergence. This will become especially important in three dimensions since multiple variables must be stored at each of the quadrature nodes for future SDC iterations. In our numerical examples, we take p to be the smallest value such that the quadrature order of accuracy (in our case, $2p - 2$) is greater than or equal to $n_{\mathrm{sdc}} + 1$ (the expected order of convergence of the time integrator). Once a quadrature formula for (8) is chosen, the result is a fully-implicit collocation scheme whose solution the SDC iteration attempts to converge towards.

3.4. Picard integral discretization

Equations (8) and (10) have similar structure, and we take advantage of this structure in their discretizations. For adaptivity, we want a scheme that easily allows for variable time step sizes. For this reason, we use only first-order methods for (8) and (10). When desired, we use SDC iterations to increase the accuracy.

The first-order provisional solution is found at the substeps $t_1, \ldots, t_p$ by discretizing (8) as

$$\mathbf{x}_j^{m+1} = \mathbf{x}_j^m + \Delta t_m \sum_{k=1}^{M} \mathbf{v}(\mathbf{x}_j^m; \mathbf{x}_k^{m+1}), \quad m = 1, \ldots, p-1, \tag{11}$$

where $\Delta t_m = t_{m+1} - t_m$. This is exactly the first-order time integrator we introduced in [42]. As we march in time with (11), we save the variables required for near-singular integration for future SDC iterations. Then, we evaluate the residual (9) using the Gauss–Lobatto quadrature rule.

In line with [37], which considers semi-implicit SDC methods, we would like to discretize (10) as

$$\mathbf{e}_{\mathbf{x}_j}^{m+1} = \mathbf{e}_{\mathbf{x}_j}^m + \mathbf{r}_j^{m+1} - \mathbf{r}_j^m + \Delta t_m \sum_{k=1}^{M} (\mathbf{v}(\tilde{\mathbf{x}}_j^m + \mathbf{e}_{\mathbf{x}_j}^m; \tilde{\mathbf{x}}_k^{m+1} + \mathbf{e}_{\mathbf{x}_k}^{m+1}) - \mathbf{v}(\tilde{\mathbf{x}}_j^m, \tilde{\mathbf{x}}_k^{m+1})).$$

The issue with this formulation is that it requires additional storage and computations to find the velocity due to the vesicle parameterized by $\tilde{\mathbf{x}}_j^m + \mathbf{e}_{\mathbf{x}_j}^m$. Again, in three dimensions this restriction is even more prohibitive. We have experimented with other discretizations of (10). The simplest such one is

$$\mathbf{e}_{\mathbf{x}_j}^{m+1} = \mathbf{e}_{\mathbf{x}_j}^m + \mathbf{r}_j^{m+1} - \mathbf{r}_j^m.$$

This discretization is consistent with the governing equations,[1] but, experimentally, SDC converges only if a very small time step is used. To allow for larger time steps, we can include the implicit term in the discretization

$$\mathbf{e}_{\mathbf{x}_j}^{m+1} = \mathbf{e}_{\mathbf{x}_j}^m + \mathbf{r}_j^{m+1} - \mathbf{r}_j^m + \Delta t_m \sum_{k=1}^{M} (\mathbf{v}(\tilde{\mathbf{x}}_j^m; \tilde{\mathbf{x}}_k^{m+1} + \mathbf{e}_{\mathbf{x}_k}^{m+1}) - \mathbf{v}(\tilde{\mathbf{x}}_j^m; \tilde{\mathbf{x}}_k^{m+1})), \tag{12}$$

where we use a slight abuse of notation by defining

$$\mathbf{v}(\tilde{\mathbf{x}}_j^m; \tilde{\mathbf{x}}_k^{m+1} + \mathbf{e}_{\mathbf{x}_k}^{m+1}) = \mathcal{S}(\tilde{\mathbf{x}}_j^m, \tilde{\mathbf{x}}_k^m)(-\mathcal{B}(\tilde{\mathbf{x}}_k^m)(\tilde{\mathbf{x}}_k^{m+1} + \mathbf{e}_{\mathbf{x}_k}^{m+1}) + \mathcal{T}(\tilde{\mathbf{x}}_k^m)(\sigma_k^{m+1} + e_{\sigma_k}^{m+1})).$$

(Note how none of the operators depend on the error $\mathbf{e}$.) Appendix A presents this same discretization without the abuse of notation. While this discretization allows larger time steps, it does not converge to the solution of the fully-implicit discretization of (8) and is incompatible with the inextensibility constraint (see Appendix A). As an alternative to (12), an appropriate discretization of the error equation is

$$\mathbf{e}_{\mathbf{x}_j}^{m+1} = \mathbf{e}_{\mathbf{x}_j}^m + \mathbf{r}_j^{m+1} - \mathbf{r}_j^m + \Delta t_m \sum_{k=1}^{M} (\mathbf{v}(\tilde{\mathbf{x}}_j^{m+1}; \tilde{\mathbf{x}}_k^{m+1} + \mathbf{e}_{\mathbf{x}_k}^{m+1}) - \mathbf{v}(\tilde{\mathbf{x}}_j^{m+1}; \tilde{\mathbf{x}}_k^{m+1})), \tag{13}$$

where we are using the same abuse of notation. Note that (13) only requires evaluating the velocity field due to the vesicle configuration given by $\tilde{\mathbf{x}}_j^{m+1}$. Since these velocity fields are required to form residual $\mathbf{r}$, no additional velocity fields need to be formed.

To summarize, the main steps for using SDC to solve (8) are

1. Find a first-order provisional solution $\tilde{\mathbf{x}}$ at the Gauss–Lobatto quadrature points using (11).
2. Compute the residual $\mathbf{r}$ by approximating the integral in (9) with the Gauss–Lobatto quadrature rule.
3. Use (13) to approximate the error $\tilde{\mathbf{e}}$.
4. Define the new provisional solution to be $\tilde{\mathbf{x}} + \tilde{\mathbf{e}}$.
5. Repeat steps 2–5 n_{sdc} times.

3.5. Preconditioning

Equations (11) and (13) are ill-conditioned and require a large number of GMRES iterations [55]. To reduce the cost of the linear solves, we used a block-diagonal preconditioner in [42] which is formed and factorized in matrix form at each time step. Using this preconditioner, the number of preconditioned GMRES iterations depends only on the magnitude of the inter-vesicle interactions, which in turn is a function of the proximity of the vesicles. For further savings, we freeze

[1] If $\mathbf{e}_{\mathbf{x}_j}^m$ converges to 0, then $\mathbf{r}_j^{m+1} = \mathbf{r}_j^m$, and by (9), SDC has converged to the solution of the fully-implicit collocation discretization of (8).

and factorize the preconditioner at the first Gauss–Lobatto point, and this preconditioner is used for all the subsequent Gauss–Lobatto points and SDC iterates. By freezing the preconditioner, there is a slight increase in the number of GMRES iterations as the solution is formed at the each subsequent Gauss–Lobatto substep. However, the savings is significant when compared to constructing and factorizing the preconditioner at each substep. Moreover, we only require one matrix factorization per time step, which is the number of factorizations required in the preconditioner we introduced in [42].

3.6. Complexity estimates

Here we summarize the cost of the most expensive algorithms required in our formulation.

- *Matrix-vector multiplication:* For unbounded flows, if M vesicles are each discretized with N points, the bending and tension calculations require $\mathcal{O}(MN \log N)$ operations using the FFT, and the single-layer potential requires $\mathcal{O}(MN)$ operations using the fast multipole method (FMM). If the solid wall is discretized with N_{wall} points, using the FMM, the matrix-vector multiplication requires $\mathcal{O}(MN \log N + N_{\text{wall}})$ operations.
- *Computing the residual:* Given a provisional solution $\tilde{\mathbf{x}}$, its velocity $\mathbf{v}$ satisfies (7) and can be formed with a single matrix-vector multiplication. Therefore, computing the residual $\mathbf{r}$ requires p matrix-vector multiplications, one for each Gauss–Lobatto substep, and computing the residual requires $\mathcal{O}(p(MN \log N + N_{\text{wall}}))$ operations.
- *Forming the provisional solution and SDC iterations:* Equations (11) and (13) require solving the same linear system (only the right-hand sides are different), and our preconditioner results in a mesh-independent number of GMRES iterations. Therefore, if n_{gmres} total iterations are required to find the provisional solution, then n_{sdc} SDC iterations require $\mathcal{O}(n_{\text{gmres}} p(n_{\text{sdc}} + 1)(MN \log N + N_{\text{wall}}))$ operations.
- *Forming the preconditioner:* The preconditioner is computed and stored in matrix form and requires $\mathcal{O}(MN^2 \log N)$ operations per time step by using Fourier differentiation and the FFT. The preconditioner must be factorized which requires $\mathcal{O}(MN^3)$ operations. This is computed only once per time step and is reused at all the additional Gauss–Lobatto quadrature points. In two dimensions, this cost is acceptable since the number of unknowns on each vesicle is sufficiently small. However, in three dimensions, this cost is unacceptable and different preconditioners will need to be constructed.

4. Adaptive time stepping

During the course of a simulation, vesicles come close to each other and to confining walls. For instance, as a vesicle passes through a constriction (see Fig. 6), it can come very close to the solid wall and small time steps should be taken. Resolving multiple time scales is an important step towards a robust solver for vesicle suspensions for two reasons. First, the simulation is sped up since the largest possible time step is always taken, and second, we eliminate a trial-and-error procedure for finding a time step size that results in the desired tolerance.

A common strategy of an adaptive time step method is to control an estimate of the local truncation error [14,24,25]. One estimate is the difference of two numerical solutions $\mathbf{x}_1$ and $\mathbf{x}_2$. One choice for $\mathbf{x}_1$ and $\mathbf{x}_2$ is the numerical solution formed from a single time step of size Δt and one formed from two time steps of size $\Delta t/2$ (step-doubling). Another choice is to use solutions formed by two different numerical methods of different orders. For vesicle suspensions, we can instead use two invariants to estimate the local truncation error. We use the errors in area and length, which are invariant by the incompressibility and inextensibility conditions, to estimate the local truncation error. This estimate does not require forming multiple numerical solutions, and it can be computed with spectral accuracy since we use a Fourier representation for the vesicle boundary. Since a small error in area and length does not imply that the true error is small, we also perform a convergence study for the vesicle position in one of the numerical examples in Section 5. We will see that the error in position decays at the same rate as the error in area and length, and, therefore, they are justified as an estimate for the local truncation error.

We now outline how this estimate is used to accept or reject a time step, and how it selects a new time step size. Here we assume that Δt is sufficiently small that we are in the asymptotic regime described in Proposition 2.1. We will see that this is not always the case, and for this reason, we build safety factors, which we define at the end of this section, into the algorithm. Suppose we have a single vesicle[2] at time t with area $A(t)$, length $L(t)$, and the desired tolerance for the global error is ϵ at the time horizon $T = 1$. We compute the solution at time $t + \Delta t$ using a kth-order time stepping scheme which results in a new area $A(t + \Delta t)$ and length $L(t + \Delta t)$. First, we check if the solution at time $t + \Delta t$ satisfies

$$|A(t + \Delta t) - A(t)| \leq A(t)\Delta t\epsilon, \quad \text{and} \quad |L(t + \Delta t) - L(t)| \leq L(t)\Delta t\epsilon. \tag{14}$$

If condition (14) is satisfied, we accept this solution and we increase the time step size so that we are taking the largest possible Δt such that (14) is satisfied for future time steps. If condition (14) is not satisfied, we reject this solution and we compute a new solution using a smaller time step size. Given a rejected time step size, one option would be to form a new provisional solution by interpolating the rejected solution to the new nodes. However, in our current implementation, we simply form the new provisional solution using (11).

[2] If there are multiple vesicles, we choose the minimum requested time step size over all of the vesicles.

Regardless of the acceptance or rejection of the solution at time $t + \Delta t$, a new time step size must be chosen. We first require estimates of the asymptotic constants of proportionality for the errors in area and length

$$C_A = \frac{|A(t+\Delta t) - A(t)|}{A(t)\Delta t^{k+1}}, \quad \text{and} \quad C_L = \frac{|L(t+\Delta t) - L(t)|}{L(t)\Delta t^{k+1}}.$$

We only consider the error in the area since the same argument can be applied to the error in length. The optimal time step size Δt_{opt} satisfies

$$|A(t+\Delta t_{\text{opt}}) - A(t)| = A(t)\Delta t_{\text{opt}}\epsilon, \tag{15}$$

and we also have the estimate

$$|A(t+\Delta t_{\text{opt}}) - A(t)| = C_A A(t)\Delta t_{\text{opt}}^{k+1}. \tag{16}$$

Equating (15) and (16), Δt_{opt} satisfies

$$C_A \Delta t_{\text{opt}}^{k+1} = \Delta t_{\text{opt}}\epsilon.$$

Finally, using our estimate for C_A, we have

$$\frac{|A(t+\Delta t) - A(t)|}{A(t)\Delta t^{k+1}}\Delta t_{\text{opt}}^{k+1} = \Delta t_{\text{opt}}\epsilon,$$

which implies that our next time step size should be

$$\Delta t_{\text{opt}} = \left(\frac{A(t)\epsilon\Delta t}{|A(t+\Delta t) - A(t)|}\right)^{1/k}\Delta t. \tag{17}$$

A similar optimal time step size is computed based on the length and the smaller of these two time steps is selected for the next time step. We have also experimented with using the residual (9) to choose the optimal time step size. Unfortunately, due to the ill-conditioning of the governing equations, we found that the residual is an unreliable estimate of the truncation error. However, the residual can be used to observe if the SDC iteration is converging, and we observe this convergence in the reported examples.

When computing the optimal time step size (17), we assumed that asymptotic estimates could be used. However, we will see that the desired order of accuracy is often not achieved and the time step size (17) will not provide tight bounds for (14). Therefore, we place restrictions on the new time step size to increase the probability of the next time step being accepted. Using this strategy, we will see that for most simulations, the number of rejected time steps is acceptable and the desired tolerance is always achieved. First, we restrict the new time step size scaling to the interval $[\beta_{\text{down}}, \beta_{\text{up}}]$, where $\beta_{\text{down}} < 1$ and $\beta_{\text{up}} > 1$. Next, we multiply the new time step size by a safety factor $\alpha^{1/k} < 1$ to increase the likelihood of the next time step size being accepted. Finally, we never increase the time step size if the previous time step size is rejected [25]. In summary, if the previous time step is accepted, the new time step size is

$$\Delta t_{\text{new}} = \alpha^{1/k}\min(\beta_{\text{up}}\Delta t, \max(\Delta t_{\text{opt}}, \beta_{\text{down}}\Delta t)),$$

and if the previous time step is rejected, the new time step size is

$$\Delta t_{\text{new}} = \alpha^{1/k}\min(\Delta t, \max(\Delta t_{\text{opt}}, \beta_{\text{down}}\Delta t)).$$

The scaling factor $\alpha^{1/k}$ is chosen so that

$$\max\left(\frac{|A(t+\Delta t) - A(t)|}{A(t)}, \frac{|L(t+\Delta t) - L(t)|}{L(t)}\right) \approx \alpha\Delta t\epsilon,$$

regardless of the order k. That is, with this scaling of α, the method tries to commit the same amount of error per time step, independent of the time stepping order.

The parameters α, β_{up}, and β_{down} affect the overall efficiency of the method. For instance, if α is too large, then Δt_{new} may be too large and the time steps will be rejected too often. However, if it is too small, the bounds in (14) will not be tight which will increase the total number of time steps. We have experimented with different values and have had success with $\alpha = 0.9$, $\beta_{\text{up}} = 1.5$ and $\beta_{\text{down}} = 0.6$. These values are used for all the numerical examples. We point out that if a sufficiently small tolerance ϵ is requested, the estimate (16) will be much more accurate, and β_{up} and β_{down} will not play a role since only small changes to the time step size will be made. However, when larger tolerances are requested, (16) is less accurate, and without β_{up} and β_{down}, the algorithm may choose unreasonable time step sizes.

5. Results

We discuss the behavior of our different constant and adaptive time step size integrators for confined and unconfined flows. We are particularly interested in comparing our two second-order time integrators: BDF and $n_{\text{sdc}} = 1$. However, it is important to note that only $n_{\text{sdc}} = 1$ is compatible with an adaptive time step size. We are interested in problems whose dynamics exhibit multiple time scales, and problems with long time horizons. For such problems, it is beneficial that the largest possible time step is taken and that no trial-and-error procedure to achieve a desired tolerance is required. The main parameters are the number of points per vesicle, N, the number of points on the solid walls, N_{wall}, the number of constant size time steps, N_t, the number of SDC iterations, n_{sdc}, the number of Gauss–Lobatto quadrature points, p, and the safety parameters, β_{up}, β_{down}, and α defined in Section 4.

The simulations are performed in Matlab on a six-core 2.67 GHz Intel Xeon processor with 24 GB of memory. We use the following four numerical examples of increasing complexity.

- **Relaxation flow** (Tables 2–4 and Figs. 2–3): We consider stiffness due to self-interactions by simulating a single vesicle in a Stokes fluid with no background velocity. Motivated by Proposition 2.1, we would like each additional SDC iteration to result in an additional order of accuracy. The errors in area, length, and position are all reported. This example highlights the behavior of the scheme in the case of single-vesicle dynamics.
- **Extensional flow** (Tables 5–9 and Figs. 4–5): We consider stiffness due to vesicle–vesicle interactions by simulating two vesicles approaching each other in an extensional flow. We check the convergence of different time integrators and compare constant time step sizes and adaptive time step sizes. This example highlights the behavior of the scheme in the presence of vesicle–vesicle interactions.
- **Stenosis flow** (Table 10 and Figs. 6–8): We consider stiffness due to vesicle–wall interactions by simulating a single vesicle in a confined tube with a parabolic-profile flow at the intake and outtake. The vesicle passes through a narrow region where a smaller time step should be taken. We compare BDF with our new adaptive second-order time integrator. This example highlights the effect of different flow regimes on the time-stepper.
- **Couette** (Tables 11–14 and Figs. 9–12): We consider two different vesicle suspensions in a Couette apparatus, one with 42 vesicles and the other with 150 vesicles. These examples involve all the stiff terms in the evolution equations.

We again emphasize that when using adaptive time stepping, no trial-and-error procedure is required to guarantee that the desired tolerance is achieved. Moreover, by using the area and length as estimates for the local truncation error, multiple numerical solutions do not need to be formed. We report results using our previously introduced time integrators [42], $n_{\text{sdc}} = 0$ (first-order) and BDF (second-order), as well as results from our new time integrators, $n_{\text{sdc}} > 0$ (order $n_{\text{sdc}} + 1$). In our new time integrators, we use the fewest number of substeps so that the accuracy of the quadrature is at least as large as the expected accuracy of the time integrator. The BDF time-stepping scheme is incompatible with an adaptive time step size since it can become unstable if the time step size changes too rapidly [22]. Therefore, all the BDF runs are done with a constant time step size, which has been selected by trial-and-error.

In our results we report the number of calls to the FMM (the #fmm column in the tables), which are used in the computation of the hydrodynamic interactions (vesicle–vesicle and vesicle–wall) and are the most expensive part of our calculation. The second most expensive part is the factorization of the block-diagonal preconditioner. This part is amortized over the SDC iteration as it is computed exactly once per time step and used at all Gauss–Lobatto substeps.

All linear systems are solved with GMRES without restarts to a tolerance of 1E−10. We have experimented with smaller tolerances, but found that the condition number of the linear system is too large to justify using a smaller tolerance. We choose N and N_{wall} large enough so that the aliasing errors and quadrature error from the layer potentials are negligible. In this manner, as long as the error is not less than 1E−10, the error due to the time integrator dominates, which is the focus of this work. Finally, as described in Section 4, we set $\beta_{\text{up}} = 1.5$, $\beta_{\text{down}} = 0.6$, and $\alpha = 0.9$ in all of the experiments. Following [25], we never increase the time step size right after a rejected time step.

5.1. Relaxation flow

We consider a single vesicle, initialized as a three-to-one ellipse, in a Stokes fluid with no background velocity. We discretize the vesicle with $N = 96$ points and the time horizon is $T = 2$ which is large enough that the vesicle comes within 10% of its steady state solution. In order to do a convergence study of the vesicle shape, we also ran the simulation with four SDC iterations and 4000 time steps. The difference in position at the time horizon of each run with this "exact" solution is reported as e_p. For this example, there are no calls to the FMM because the self-interactions are evaluated directly. We report the errors and CPU timings in Tables 2–4.

We see that SDC quickly converges for a fixed Δt. However, the convergence stagnates after two SDC iterations, and this is due to other sources of error, such as the GMRES tolerance. As mentioned earlier, reducing the GMRES tolerance does not improve the results because the linear system that needs to be solved has a condition number around 1E+6. We have run the same simulation, but with a larger time horizon, larger time step sizes, and we used the LU decomposition of the system matrix rather than GMRES. In Fig. 2, we observe first-, second-, and third-order convergence. We first attempt to achieve fourth-order convergence using $p = 3$ Gauss–Lobatto points and 3 SDC iterations, but we cannot conclusively

Table 2
The errors in position, area, and length and the CPU time for a single vesicle in a **relaxation** flow with a **constant** time step size using $\boldsymbol{n_{sdc} = 0}$ (left) and **BDF** (right). In both runs, $p = 2$ meaning that there are no substeps. The CPU times for Tables 2–4 are relative to the cheapest simulation ($N_t = 125$ and $n_{sdc} = 0$) which took approximately 9.3 seconds. Both methods converge at the expected rate.

N_t	$\boldsymbol{n_{sdc} = 0}$				**BDF**			
	e_P	e_A	e_L	CPU	e_P	e_A	e_L	CPU
125	1.05E−4	3.43E−6	3.40E−5	1.0	1.11E−6	2.52E−8	9.49E−7	1.1
250	5.29E−5	1.74E−6	1.74E−5	1.9	2.10E−7	3.17E−9	1.74E−7	2.2
500	2.67E−5	8.75E−7	8.83E−6	3.6	4.18E−8	3.52E−10	2.84E−8	4.1
1000	1.34E−5	4.39E−7	4.44E−6	7.3	9.13E−9	8.59E−10	4.18E−9	8.3

Table 3
The errors in position, area, and length and the CPU time for a single vesicle in a **relaxation** flow with a **constant** time step size using $\boldsymbol{n_{sdc} = 1}$, $p = 2$ (left) and $\boldsymbol{n_{sdc} = 2}$, $p = 3$ (right). We achieve second-order convergence with one SDC iteration, but third-order convergence is only observed for the error in length with two SDC iterations. However, the errors in area and position have plateaued close to the GMRES tolerance. We observe that with $n_{sdc} = 2$ and $N_t = 125$, we get 6E−9 error with 4.2 CPU cost whereas the BDF scheme (Table 2) with $N_t = 1000$ time steps gets a slightly larger error (9E−9) at 8.3 CPU cost, about two times more expensive.

N_t	$\boldsymbol{n_{sdc} = 1}$				$\boldsymbol{n_{sdc} = 2}$			
	e_P	e_A	e_L	CPU	e_P	e_A	e_L	CPU
125	9.55E−7	1.61E−7	1.27E−7	1.7	6.26E−9	8.02E−10	2.91E−10	4.2
250	2.40E−7	4.21E−8	2.64E−8	2.7	9.44E−10	1.22E−9	3.89E−11	7.7
500	5.89E−8	1.02E−8	4.83E−9	5.7	1.41E−10	1.11E−9	4.12E−12	14.3
1000	1.43E−8	1.86E−9	7.93E−10	11.2	2.49E−11	1.04E−9	3.42E−13	29.2

Table 4
The errors in position, area, and length and the CPU time for a single vesicle in a **relaxation** flow with a **constant** time step size using $\boldsymbol{n_{sdc} = 3}$, $p = 3$ (left) and $\boldsymbol{n_{sdc} = 4}$, $p = 4$ (right). For this example, additional SDC iterations do not offer significant advantage as we have reached accuracy limits related to the conditioning of the system.

N_t	$\boldsymbol{n_{sdc} = 3}$				$\boldsymbol{n_{sdc} = 4}$			
	e_P	e_A	e_L	CPU	e_P	e_A	e_L	CPU
125	5.66E−10	1.71E−9	1.22E−11	5.1	1.36E−10	7.82E−10	1.05E−13	9.1
250	1.59E−10	1.25E−9	6.22E−13	10.0	3.39E−11	9.52E−10	4.89E−15	17.8
500	4.18E−11	1.08E−9	2.93E−13	19.4	8.43E−12	9.95E−10	4.11E−15	39.2
1000	1.08E−11	1.03E−9	1.11E−16	37.6	2.02E−12	1.01E−9	9.99E−15	71.6

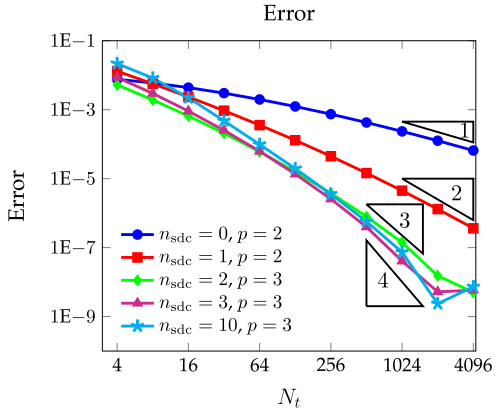

Fig. 2. The maximum error in area and length for a single vesicle in a **relaxation** flow. We use $p = 2$ Gauss–Lobatto points for $\boldsymbol{n_{sdc} = 0}$ (first-order) and $\boldsymbol{n_{sdc} = 1}$ (second-order), and $p = 3$ Gauss–Lobatto points for $\boldsymbol{n_{sdc} = 2}$ (third-order), $\boldsymbol{n_{sdc} = 3}$ (fourth-order), and $\boldsymbol{n_{sdc} = 10}$ (fourth-order). We see that first-, second-, and third-order convergence are achieved.

observe fourth-order convergence. Therefore, we also try using $p = 3$ Gauss–Lobatto points and 10 SDC iterations. In this manner, SDC has converged to the fully-implicit Gauss–Lobatto fourth-order collocation scheme of (8). We see that there is no improvement compared to the 3 SDC iterations, and, therefore, we expect that order reduction is limiting the convergence rate.

Achieving high-order convergence for stiff systems, such as those that govern vesicle suspensions, can be quite challenging. Several groups have developed methods to reduce the effect of order reduction [31,37,51,52,57]. These methods may result in asymptotic convergence rates for larger time step sizes, but they come at an expense of a larger memory footprint.

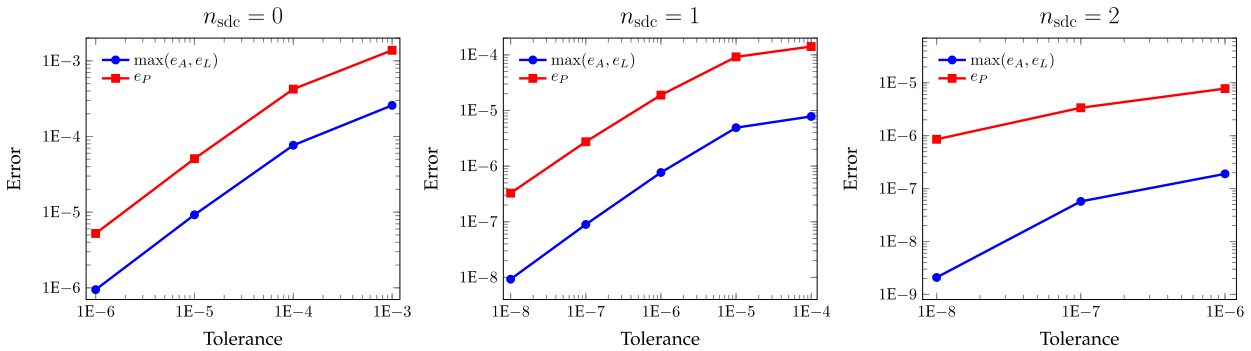

Fig. 3. The maximum error in area and length, and in position for a single vesicle in a **relaxation** flow with an **adaptive** time step size. In all the runs, the maximum error in area and length are achieved, and the error in position decays at the same rate.

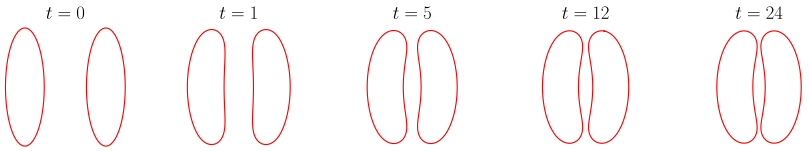

Fig. 4. Two vesicles discretized with $N = 96$ points. The vesicles are initially placed symmetrically around the origin and the background velocity is $\mathbf{v}_\infty = (-x, y)$.

Therefore, at this point, we only use one or two SDC iterations, and, therefore, will not achieve more than third-order convergence. However, we are able to compare our two second-order methods: BDF and $n_{\text{sdc}} = 1$ with and without a variable time step size. For this particular example, we see that our new time integrator ($n_{\text{sdc}} = 1$) is slightly faster than our old time integrator (BDF). In addition, we see that the error in position converges at the same asymptotic rate as the error in area and length. This is also tested for an adaptive time step size in Fig. 3. This justifies our use of the error in area and length to choose an adaptive time step size.

5.2. Extensional flow

We consider two vesicles placed symmetrically around the origin with the background velocity $\mathbf{v}_\infty = (-x, y)$ (Fig. 4). We discretize both vesicles with $N = 96$ points and the time horizon is $T = 24$ which is long enough that the distance between the vesicles at the time horizon is $0.4\sqrt{\Delta s}$, where Δs is the arclength spacing. We report results using zero, one, and two SDC iterations, and BDF in Tables 5 and 6. Similar to the previous example, for some of the results, the expected convergence rates are not observed. Other groups (see, for example, [37,57]) have observed that for stiff systems, very small time steps must be taken before the asymptotic convergence rates are achieved. As before, we expect that other sources of error will dominate once the temporal asymptotic regime is achieved.

As the vesicles approach one another, their shape is nearly static since their velocities decrease, and we expect that larger time step sizes can be taken. Therefore, we test our adaptive time stepping strategy using zero, one, and two SDC iterations. The errors in area and length, the number of accepted and rejected time steps, the number of FMM calls, and the CPU times are reported in Table 7 ($n_{\text{sdc}} = 0$), Table 8 ($n_{\text{sdc}} = 1$), and Table 9 ($n_{\text{sdc}} = 2$). The adaptive time stepping strategy does a good job of attaining the desired tolerance while not having too many rejected time steps. In Fig. 5, we plot the time step size with the location of the rejected time steps (top left), and the errors in area and length for a constant time step size and for an adaptive time step size (top right). While the constant time step size commits a negligible amount of error shortly after the initial condition, too much error is accumulated at the beginning of the simulation. This indicates that a smaller time step should be taken near the start of the simulation and then it can be increased later in the simulation, which is exactly what we observe.

In Tables 5–9 and the bottom plot of Fig. 5, we compare the results of the different constant and adaptive time step size integrators. We see that our new second-order method, $n_{\text{sdc}} = 1$, slightly outperforms our previous second-order method, BDF. Moreover, by using an adaptive time step size, there is a further reduction in the required CPU time. For example, to achieve a tolerance of 1E−6 with $n_{\text{sdc}} = 1$, a constant time step size requires about 10 times more CPU time than if an adaptive time step size is used. Finally, we see that with our current implementation, taking two SDC iterations does not result in a computational speedup compared to $n_{\text{sdc}} = 1$. Therefore, we only use one SDC iteration from this point onwards.

Table 5
The errors in area and length and the CPU time for two vesicles in an **extensional** flow with a **constant** time step size using $\boldsymbol{n_{sdc} = 0}$ (left) and **BDF** (right). The CPU times for Tables 5–9 are relative to the cheapest simulation ($N_t = 300$ and $n_{sdc} = 0$) which took approximately 453 seconds. We achieve the desired first-order results, but second-order results are not yet achieved. With additional time steps, second-order convergence is achieved (see Table 7 in [42]).

N_t	$\boldsymbol{n_{sdc} = 0}$				**BDF**			
	e_A	e_L	# fmm	CPU	e_A	e_L	# fmm	CPU
300	2.47E−4	1.27E−3	4.21E3	1.0	3.80E−5	3.90E−4	4.53E3	1.2
600	1.24E−4	6.64E−4	8.31E3	2.1	9.84E−6	1.37E−4	8.90E3	2.0
1200	6.29E−5	3.46E−4	1.63E4	3.8	2.80E−6	4.45E−5	1.75E4	4.0
2400	3.20E−5	1.79E−4	3.17E4	7.8	1.04E−6	1.27E−5	3.45E4	7.6

Table 6
The errors in area and length and the CPU time for two vesicles in an **extensional** flow with a **constant** time step size using $\boldsymbol{n_{sdc} = 1}$, $p = 2$ (left) and $\boldsymbol{n_{sdc} = 2}$, $p = 3$ (right). While each SDC iteration reduces the error, the desired asymptotic rates of convergence are not achieved. As the ratios of successive errors are approaching the expected values, we expect that we have not taken enough time steps to observe the asymptotic rate of convergence. Also, we observe that our new time integrator, $n_{sdc} = 1$, slightly outperforms BDF, even with a constant time step size (see Fig. 5).

N_t	$\boldsymbol{n_{sdc} = 1}$				$\boldsymbol{n_{sdc} = 2}$			
	e_A	e_L	# fmm	CPU	e_A	e_L	# fmm	CPU
300	4.99E−5	2.91E−5	1.04E4	2.5	2.01E−6	4.84E−7	2.93E4	7.2
600	1.63E−5	1.04E−5	1.96E4	4.9	7.35E−7	8.14E−8	5.58E4	13.3
1200	5.35E−6	3.61E−6	3.76E4	9.2	5.24E−7	1.45E−8	1.07E5	26.0
2400	1.94E−6	1.13E−6	7.22E4	16.0	5.04E−7	2.89E−9	2.10E5	51.2
4800	9.15E−7	3.07E−7	1.40E5	34.9	5.08E−7	6.11E−10	4.12E5	99.1

Table 7
The errors in area and length, the CPU time, and the number of accepted and rejected time steps for two vesicles in an **extensional** flow with an **adaptive** time step size using $\boldsymbol{n_{sdc} = 0}$. For the larger tolerances, the desired tolerance is achieved in an acceptable CPU time. However, first-order methods require too small of time steps to achieve four digits of accuracy. With these smaller tolerances, higher-order methods should be used.

Tolerance	e_A	e_L	Accepts	Rejects	# fmm	CPU
1E−2	1.56E−4	9.97E−4	79	16	1.06E3	0.3
1E−3	3.95E−5	2.46E−4	549	25	5.11E3	1.3
1E−4	6.50E−6	3.60E−5	5068	25	4.19E4	11.2

Table 8
The errors in area and length, the CPU time, and the number of accepted and rejected time steps for two vesicles in an **extensional** flow with an **adaptive** time step size and $\boldsymbol{n_{sdc} = 1}$, $p = 2$. This second-order method is able to achieve much smaller tolerances than $n_{sdc} = 0$.

Tolerance	e_A	e_L	Accepts	Rejects	# fmm	CPU
1E−2	3.63E−4	3.81E−3	53	21	2.67E3	0.6
1E−3	2.93E−5	4.81E−4	60	26	2.97E3	0.7
1E−4	1.70E−5	4.83E−5	94	25	3.66E3	0.9
1E−5	3.91E−6	3.89E−6	210	36	6.20E3	1.4
1E−6	5.69E−7	2.95E−7	562	73	1.43E4	3.4

Table 9
The errors in area and length, the CPU time, and the number of accepted and rejected time steps for two vesicles in an **extensional** flow with an **adaptive** time step size using $\boldsymbol{n_{sdc} = 2}$, $p = 3$. We see that there is no use to use more than one SDC iteration with our current implementation.

Tolerance	e_A	e_L	Accepts	Rejects	# fmm	CPU
1E−4	1.92E−5	2.36E−5	59	19	8.72E3	2.0
1E−5	4.23E−6	1.92E−6	87	13	9.77E3	2.3
1E−6	4.82E−7	1.12E−7	194	11	1.60E4	4.0

5.3. Stenosis flow

We consider a single vesicle discretized with $N = 128$ points in a constricted tube discretized with $N_{wall} = 768$ points (Fig. 6). The time horizon, $T = 80$, is chosen so that the vesicle is forced to pass through the constriction. In Fig. 7, we plot the errors in area and length using BDF with a constant time step size. Unsurprisingly, the errors increase when the vesicle passes through the constriction. In this interval, a smaller time step should be taken. Therefore, we test our second-order adaptive time stepper, $n_{sdc} = 1$. The results are summarized in Table 10. In Fig. 8, we plot the time step size and compare the two second-order time integrators. As expected, we see that a small time step is taken as the vesicle passes through the constriction. In fact, the time step size varies over more than three orders of magnitude which indicates that this example

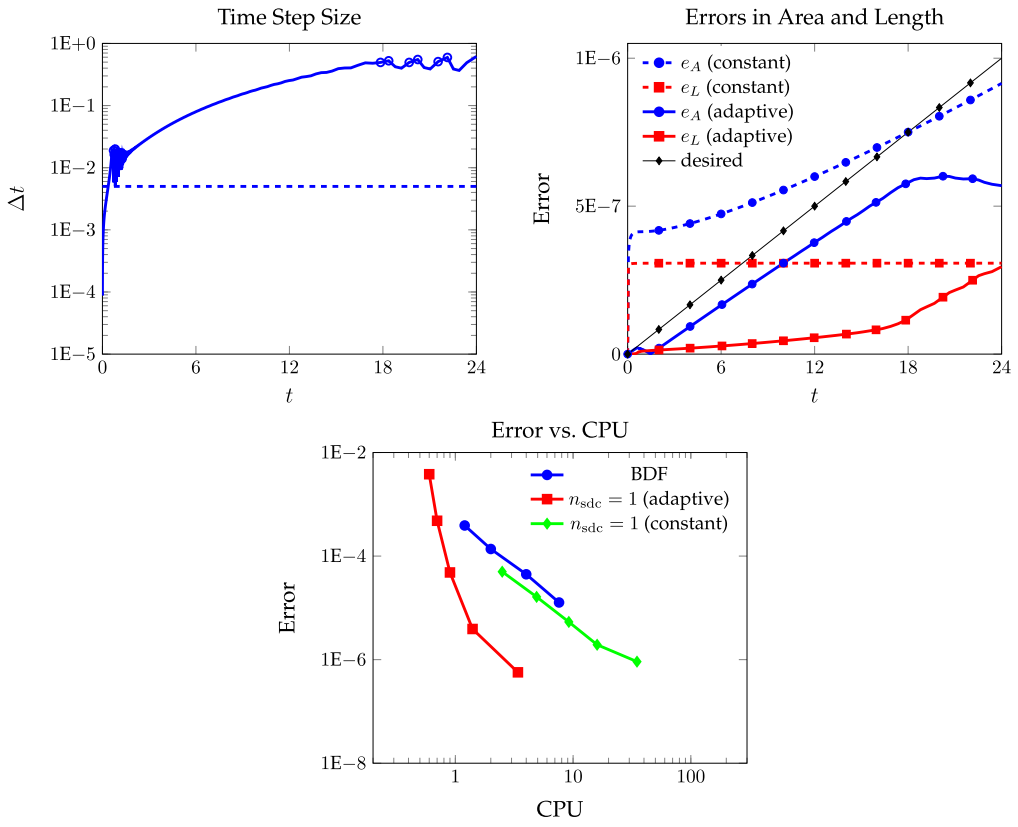

Fig. 5. Results for the **extensional** flow. Top left: The time step size using 4800 **constant** time step sizes (dashed) and an **adaptive** time step size (solid) with a tolerance of 1E−6 and $\boldsymbol{n_{\text{sdc}} = 1}$, $p = 2$. The open circles indicate the times when a time step is rejected. Top right: The errors in area and length using **constant** (dashed) and **adaptive** (solid) time steps, and the desired error (black) of the adaptive time step. Bottom: The error versus the required CPU time for the three different second-order time integrators.

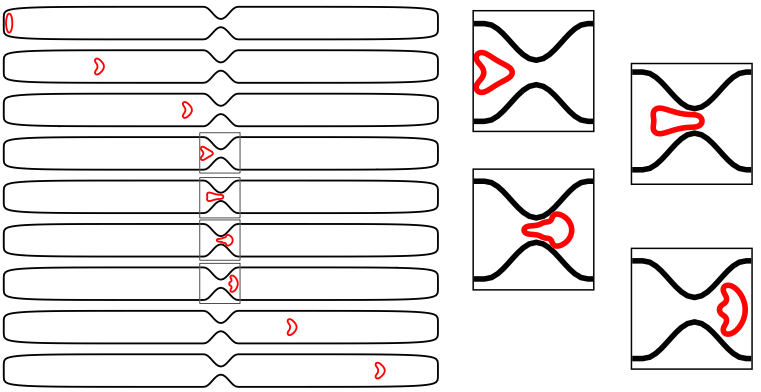

Fig. 6. A single vesicle discretized with $N = 128$ points passing through a constricted tube (**stenosis**) discretized with $N_{\text{wall}} = 768$ points. The boundary condition at the intake and outtake has a parabolic-profile and on the rest of the solid wall is zero. On the right are magnifications of the vesicle as it passes through the constriction.

benefits greatly from an adaptive time step size. This is verified in right plot of Fig. 8 where we see that our new time integrator outperforms BDF. In particular, we see that adaptive $n_{\text{sdc}} = 1$ achieves close to an extra digit of accuracy for a fixed CPU time.

5.4. Couette flow

As a final example, we consider two suspensions in a Couette apparatus, a test case we also considered in our previous work [42]. We first consider a suspension of 42 vesicles where the inner and outer boundaries are not aligned. This results

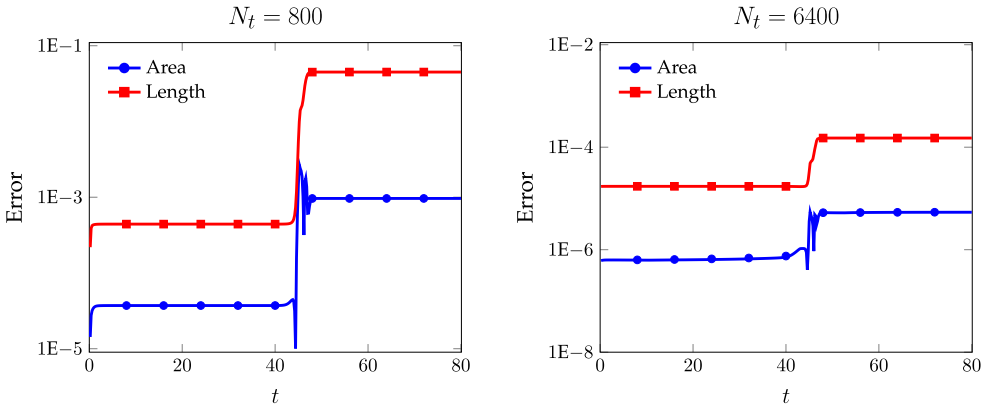

Fig. 7. The errors in area and length for a single vesicle in a constricted tube (**stenosis**) with a **constant** time step size using **BDF**. Notice that the error increases sharply as the vesicle passes through the constriction for both time step sizes. This indicates that a smaller time step size should be taken as the vesicle passes through the constriction.

Table 10
The errors in area and length, the CPU time, and the number of accepted and rejected time steps for a single vesicle in a constricted tube (**stenosis**) with an **adaptive** time step size using $\boldsymbol{n_{\text{sdc}} = 1}$ and $p = 2$ Gauss–Lobatto points. The CPU times are relative to the cheapest simulation ($N_t = 800$ and BDF) which took approximately 61 minutes.

Tolerance	e_A	e_L	Accepts	Rejects	# fmm	CPU
1E−1	6.29E−3	5.11E−3	95	37	3.33E3	0.5
1E−2	2.59E−3	1.37E−3	235	54	7.55E3	0.9
1E−3	7.11E−4	4.55E−4	697	46	1.83E4	2.3
1E−4	6.49E−5	5.62E−5	2289	93	5.62E4	7.2

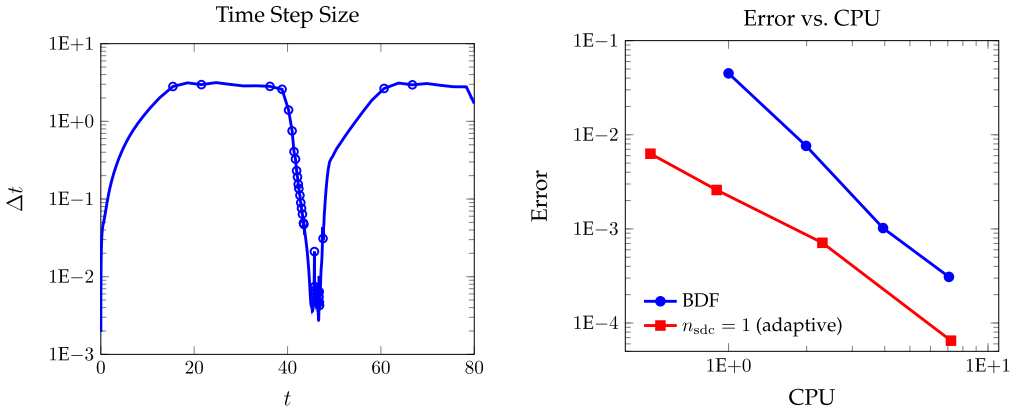

Fig. 8. Results for the **stenosis** flow. Left: The time step size using **adaptive** $\boldsymbol{n_{\text{sdc}} = 1}$, $p = 2$ with a tolerance of 1E−3. Notice that the time step size decreases as the vesicle passes through the constriction. The open circles indicate the times when the time step size is rejected. Right: The error versus the required CPU time for BDF and adaptive $\boldsymbol{n_{\text{sdc}} = 1}$, $p = 2$. We see that with our adaptive time integrator, we are able to achieve close to an extra digit of accuracy for a fixed CPU time.

in a narrow region that vesicles must pass through. We take a time horizon of $T = 10$ corresponding to one full rotation of the inner boundary (Fig. 9). In Tables 11 and 12, we report convergence results for BDF and adaptive $n_{\text{sdc}} = 1$. We can see in Fig. 10 that if the time step size is too large, then the method is either unstable or the error grows sharply and results in the error tolerance being exceeded. In particular, there is an interaction near $t = 1$ that needs to be resolved[3] with a time size smaller than $\Delta t = 3\text{E}{-2}$. However, at later times, a larger time step size can be chosen. By taking an adaptive time step size, all the interactions are resolved appropriately, no trial-and-error procedure to find the required time step size is used, and the total error committed per time step is controlled. The time step size is illustrated in the left plot of Fig. 10.

Finally, we consider 150 vesicles in a Couette apparatus with a longer time horizon (Fig. 11); the inner boundary makes five full revolutions. In Tables 13 and 14, we report the convergence results for BDF and adaptive $n_{\text{sdc}} = 1$. Similar to the 42 vesicle simulation, in Fig. 12, we see that the errors exceed the tolerance or have sharp jumps if the time step size is

[3] This interaction is due to the blue vesicle with its neighboring vesicles and the solid wall.

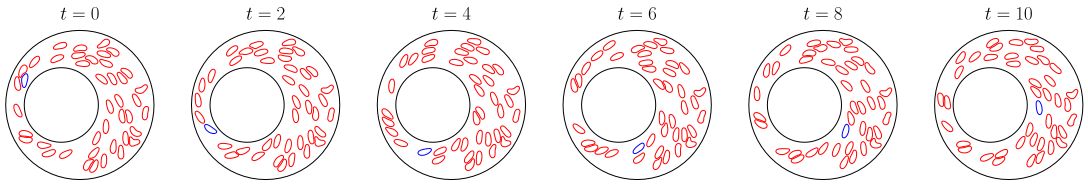

Fig. 9. 42 vesicles in a **Couette** apparatus. The outer boundary is stationary and the inner boundary has constant angular velocity and has completed one full rotation at $T = 10$. A single vesicle is colored in blue to help with the visualization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 11
The errors in area and length and the CPU time for 42 vesicles in a **Couette** apparatus with a **constant** time step size using **BDF**. The CPU times for Tables 11 and 12 are relative to the cheapest simulation which took approximately 4 hours and 45 minutes. We demonstrate how a user often has to do several runs with different time step sizes before a desired tolerance (for this example, 1E−1) is achieved.

N_t	e_A	e_L	# fmm	CPU
160	tolerance exceeded at $t = 0.8$		7.76E4	0.1
240	tolerance exceeded at $t = 0.8$		1.13E4	0.2
280	tolerance exceeded at $t = 1.1$		1.39E4	0.2
286	tolerance exceeded at $t = 2.0$		1.79E4	0.3
288	5.10E−4	9.94E−2	4.01E4	1.0
320	3.76E−3	7.92E−2	4.34E4	1.1
640	9.48E−5	8.60E−4	8.34E4	2.1
1144	2.80E−4	2.17E−5	1.71E5	4.4

Table 12
The errors in area and length, the CPU time, and the number of accepted and rejected time steps for 42 vesicles in a **Couette** apparatus with an **adaptive** time step size using $\boldsymbol{n_{sdc} = 1}$. Our new time integrator commits slightly less error for a fixed CPU time, but, more importantly, only a single simulation is required to achieve the desired tolerance.

Tolerance	e_A	e_L	Accepts	Rejects	# fmm	CPU
1E−1	1.24E−2	2.53E−2	69	23	3.07E4	0.7
1E−2	3.51E−3	2.15E−3	155	42	7.07E4	1.7

constant,[4] and this shortcoming is removed by using an adaptive time step size. The time step size is illustrated in the right plot of Fig. 12. Notice that the time step size is small near $t = 6$, indicating that there is at least one vesicle that is committing a large error if too large of a time step is taken. The left plot of Fig. 12 indicates that this is exactly the case. Also notice that the BDF scheme is sensitive on the time step size. If we select $N_t = 900$ (Table 13), we fail at almost the cost of a successful simulation which is $N_t = 920$. Of course one can take much larger number of time steps, but the cost may be too expensive and the accuracy achieved may be unnecessary on an actual science run. On the other hand, the adaptive SDC behaves in a controlled way and delivers the desired accuracy.

In these simulations, if we could guess the right time step size a priori, a constant time step BDF scheme will be slightly faster than the adaptive SDC scheme. The main reason is that these flows do not have multiple time scales, two vesicles are always nearby or near the wall, thus dictating the same time step across the simulation. Indeed, in the adaptive time stepping results, the local truncation error is estimated by taking the maximum error committed over all the vesicles. Therefore, the time step size is set by the vesicle that commits the most error. We could easily change the setup (e.g., by making the rotation velocity of the Couette apparatus to be time dependent with large variation) to introduce multiple time scales, but we think our point of robustness is clear and has been demonstrated in the stenosis and extensional flow examples. Perhaps a faster method can be developed by allowing each vesicle to have its own time step size, that is, using a multirate time integrator, but this is beyond the scope of this paper.

6. Conclusions

We have developed and tested an adaptive time stepping method for an integro-differential equation formulation for the numerical simulation of vesicle suspensions. Adaptive time stepping alleviates the need for time step size selection which increases robustness, increases the performance in the presence of time scales that vary significantly throughout the simulation, and can be used to deliver any final-time accuracy in a controlled way.

[4] The sharp jump near $t = 6$ is due to the interaction of the blue vesicle with its neighboring vesicles and the solid wall.

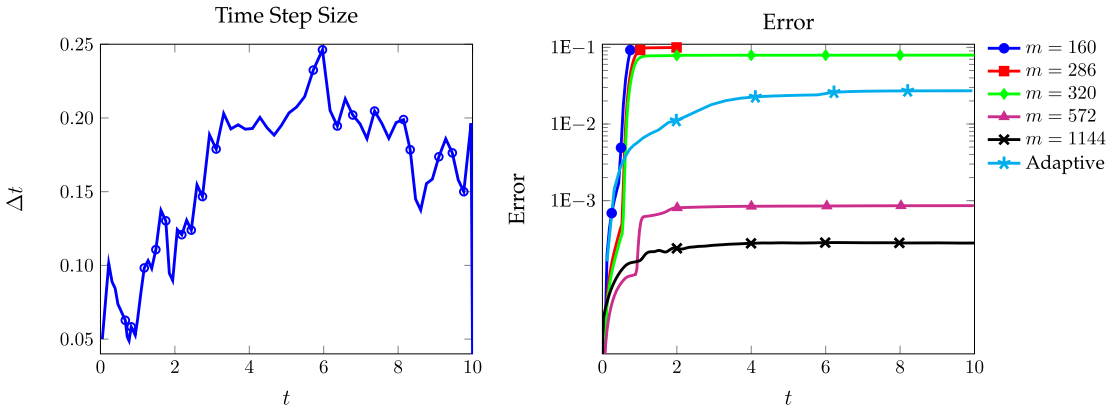

Fig. 10. Left: The time step size using **adaptive** $\boldsymbol{n_{sdc} = 1}$, $p = 2$ with a tolerance of 1E−2. The open circles indicate the locations where a time step size is rejected. Right: The error of **BDF** and **adaptive** $\boldsymbol{n_{sdc} = 1}$ for 42 vesicles in a **Couette** apparatus. If a user specifies too large of a time step, then the error exceeds the tolerance of 1E−1. By using an **adaptive** time step size, the error committed per time step is controlled.

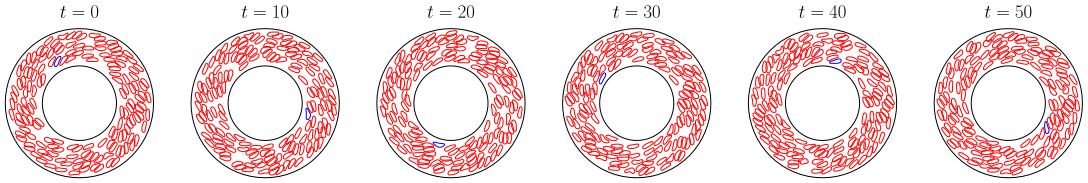

Fig. 11. 150 vesicles in a **Couette** apparatus. The outer boundary is stationary and the inner boundary has constant angular velocity and has completed five full rotations at $T = 50$. A single vesicle is colored in blue to help with the visualization. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 13
The errors in area and length and the CPU time for 150 vesicles in a **Couette** apparatus with a **constant** time step size using **BDF**. The CPU times for Tables 13 and 14 are relative to the cheapest successful simulation ($N_t = 920$) which took approximately 71.7 hours. We demonstrate how a user often has to do several runs with different time step sizes before a desired tolerance (for this example, 1E−1) is achieved.

N_t	e_A	e_L	# fmm	CPU
500	tolerance exceeded at $t = 1.2$		7.35E4	0.01
750	tolerance exceeded at $t = 5.7$		1.76E5	0.19
850	tolerance exceeded at $t = 7.5$		2.29E5	0.24
900	tolerance exceeded at $t = 38.1$		6.94E5	0.78
920	1.55E−2	7.15E−2	8.91E5	1.0
1000	8.21E−3	4.42E−2	8.71E5	1.1

Table 14
The errors in area and length, the CPU time, and the number of accepted and rejected time steps for 150 vesicles in a **Couette** apparatus with an **adaptive** time step size using $\boldsymbol{n_{sdc} = 1}$.

Tolerance	e_A	e_L	Accepts	Rejects	# fmm	CPU
3E−1	3.77E−2	5.13E−2	469	135	1.25E6	1.5
1E−1	1.65E−2	1.50E−2	803	153	2.00E6	2.1

To select the time step size, we followed the standard strategy of estimating the local truncation error and then adjusting the time step size so that the same amount of error is committed per time step. Standard methods require the computation of more accurate provisional solutions, which can be very expensive in our context. Instead, we introduced a technique where the error in area and length of the vesicles, which are invariant for vesicles, are used to estimate the local truncation error—basically for free. We justified the use of these invariants by (empirically) showing that the error in position decays at the same rate as the errors in area and length. While this method proved to be reliable, it can only be applied to problems with measurable invariants. But many other particulate flows share similar properties; for example, any non-permeable capsule filled with an incompressible fluid must preserve its volume.

To allow for larger time steps and longer time horizons, we introduced new high-order time integrators based on spectral deferred corrections (SDC). An advantage of these integrators is that, in contrast to our previous second-order integrator, BDF, they are compatible with an adaptive time step size. This time integrator is accelerated using the fast multipole method

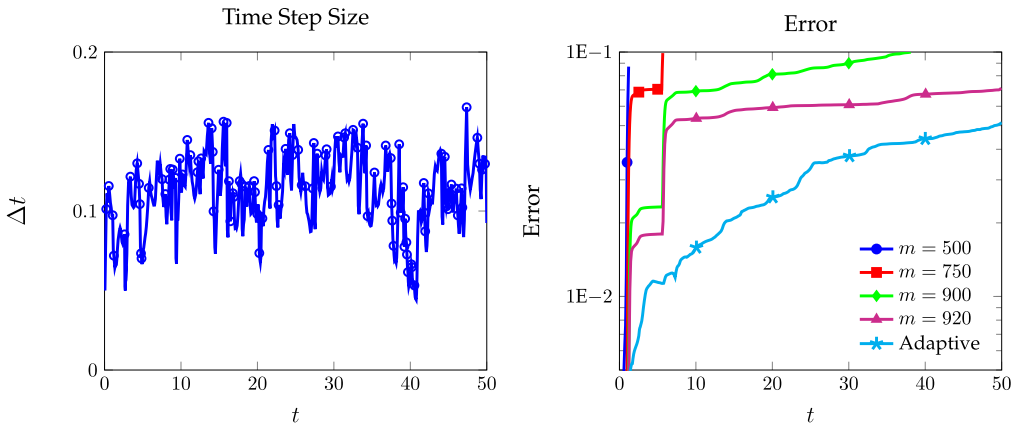

Fig. 12. Left: The time step size as a function of time for the **adaptive** $n_{\text{sdc}} = 1$ with a tolerance of 3E−1. The open circles indicate the locations where a time step size is rejected. Right: The error of **BDF** and **adaptive** $n_{\text{sdc}} = 1$ for 150 vesicles in a **Couette** apparatus. If a user specifies too large of a time step, then the error exceeds the tolerance of 1E−1. Moreover, even if the tolerance is not exceeded, there still are instances when the error quickly grows. For instance, near times $t = 2$ and $t = 6$. By using an **adaptive** time step size, the amount of error committed per time step is controlled.

and a preconditioner that is factored once per time step, and then frozen for all the Gauss–Lobatto substeps and SDC iterations. To the best of our knowledge, this work is the first to apply IMEX methods coupled with SDC to a two-dimensional integro-differential equation.

These contributions are a major step towards the development of a robust solver for vesicle suspensions, but several other features need to be introduced.

- Theoretical results, in particular the convergence rates of SDC, need to be developed. This will allow us to take the largest possible time step size given the user specified tolerance. Since the governing equations for vesicle suspension are very stiff, order reduction is expected and observed. There are methods to remove order reduction such as Krylov deferred correction [11,12,28,30,31] and multilevel SDC [21,36,51]. Another possible direction to accelerate the method is to use inexact SDC [36,51,52]. To the best of our knowledge, none of these methods have been applied to integro-differential equations, such as those that govern vesicle suspensions.
- Many vesicle suspensions could benefit from a time integrator that treats each vesicle with its own time step size, or a multirate time integrator. This is apparent in the *Couette* example where the actual error was not very tight to the desired tolerance.
- All of our results are two-dimensional. However, none of the methods that we have introduced in this paper are restricted to two dimensions. In particular, the volume and a surface area of a three-dimensional vesicle can be accurately computed using its spherical harmonic representation [56]. Performing three-dimensional simulations only requires a careful implementation of the methods and the development of suitable preconditioners.

Acknowledgements

This material is based upon work supported by AFOSR grants FA9550-12-10484 and FA9550-11-10339; and NSF grants CCF-1337393, OCI-1029022; and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Numbers DE-SC0010518, DE-SC0009286, and DE-FG02-08ER2585; and by the Technische Universität München—Institute for Advanced Study, funded by the German Excellence Initiative (and the European Union Seventh Framework Programme under grant agreement 291763). Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the AFOSR or the NSF.

Appendix A. Complete SDC formulation

Here we write our SDC formulation for the unbounded vesicle formulation that includes the tension σ_j. We do not use the notation $\mathbf{v}(\mathbf{x}_j; \mathbf{x}_k)$ to help clarify the discretization of the prediction and SDC equations. Given the solution at time t_m, let $\tilde{\mathbf{x}}$ and $\tilde{\sigma}$ be a provisional solution of

$$\mathbf{x}_j(t) = \mathbf{x}_j(t_m) + \int_{t_m}^{t} \left(\mathbf{v}_\infty(\mathbf{x}_j) + \sum_{k=1}^{M} \mathcal{S}(\mathbf{x}_j, \mathbf{x}_k)(-\mathcal{B}(\mathbf{x}_k)(\mathbf{x}_k) + \mathcal{T}(\mathbf{x}_k)(\sigma_k)) \right) d\tau, \quad t \in [t_m, t_{m+1}], \tag{18}$$

which can be formed by a time stepping method followed by an interpolation step. Then, the residual for $t \in [t_m, t_{m+1}]$ is

$$\mathbf{r}_j(t;\tilde{\mathbf{x}},\tilde{\sigma}) = \tilde{\mathbf{x}}_j(t_m) - \tilde{\mathbf{x}}_j(t) + \int_{t_m}^{t} \left(\mathbf{v}_\infty(\tilde{\mathbf{x}}_j) + \sum_{k=1}^{M} \mathcal{S}(\tilde{\mathbf{x}}_j,\tilde{\mathbf{x}}_k)(-\mathcal{B}(\tilde{\mathbf{x}}_k)(\tilde{\mathbf{x}}_k) + \mathcal{T}(\tilde{\mathbf{x}}_k)(\tilde{\sigma}_k)) \right) d\tau .$$

As before, the residual is approximated with a quadrature rule, and, therefore, the interpolation step is not required in practice. The error in the position is $\mathbf{e}_{\mathbf{x}_j} = \mathbf{x}_j - \tilde{\mathbf{x}}_j$, and the error in tension is $e_{\sigma_j} = \sigma_j - \tilde{\sigma}_j$. Substituting the errors into (18), for $t \in [t_m, t_{m+1}]$,

$$\begin{aligned}
\mathbf{e}_{\mathbf{x}_j}(t) = \mathbf{r}_j(t_m;\tilde{\mathbf{x}}_j,\tilde{\sigma}_j) &+ \int_{t_m}^{t} (\mathbf{v}_\infty(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}) - \mathbf{v}_\infty(\tilde{\mathbf{x}}_j)) d\tau \\
&- \int_{t_m}^{t} \sum_{k=1}^{M} \left(\mathcal{S}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}, \tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) \mathcal{B}(\tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) - \mathcal{S}(\tilde{\mathbf{x}}_j,\tilde{\mathbf{x}}_k)\mathcal{B}(\tilde{\mathbf{x}}_k) \right) (\tilde{\mathbf{x}}_k) d\tau \\
&+ \int_{t_m}^{t} \sum_{k=1}^{M} \left(\mathcal{S}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}, \tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) \mathcal{T}(\tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) - \mathcal{S}(\tilde{\mathbf{x}}_j,\tilde{\mathbf{x}}_k)\mathcal{T}(\tilde{\mathbf{x}}_k) \right) (\tilde{\sigma}_k) d\tau \\
&+ \int_{t_m}^{t} \sum_{k=1}^{M} \mathcal{S}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}, \tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k})(-\mathcal{B}(\tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k})(\mathbf{e}_{\mathbf{x}_k}) + \mathcal{T}(\tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k})(e_{\sigma_k})) d\tau .
\end{aligned} \tag{19}$$

For confined flows, $\mathbf{v}_\infty$ depends on yet another variable, a density function defined on the solid walls Γ, which we denote by $\boldsymbol{\eta}$. The dependence is through a double-layer potential $\mathcal{D}(\tilde{\mathbf{x}}_j, \Gamma)$. In this case, the term involving $\mathbf{v}_\infty$ in (19) becomes

$$\int_{t_m}^{t} (\mathcal{D}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}, \Gamma) - \mathcal{D}(\tilde{\mathbf{x}}_j, \Gamma))(\boldsymbol{\eta}) d\tau, \quad t \in [t_m, t_{m+1}],$$

and the final term in (19) includes the term $\mathcal{D}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}, \Gamma)(\mathbf{e}_{\eta_k})$. The additional equation required, since we have introduced the new variable $\boldsymbol{\eta}$, is a no-slip boundary condition on the solid walls.

The inextensibility constraint (6) is

$$\begin{aligned}
0 &= \mathrm{Div}(\mathbf{x}_j) \left(\mathbf{v}_\infty(\mathbf{x}_j) + \sum_{k=1}^{M} \mathcal{S}(\mathbf{x}_j,\mathbf{x}_k)(-\mathcal{B}(\mathbf{x}_k)(\mathbf{x}_k) + \mathcal{T}(\mathbf{x}_k)(\sigma_k)) \right) \\
&= \frac{d\mathbf{x}_j}{ds} \cdot \frac{d}{ds} \left(\mathbf{v}_\infty(\mathbf{x}_j) + \sum_{k=1}^{M} \mathcal{S}(\mathbf{x}_j,\mathbf{x}_k)(-\mathcal{B}(\mathbf{x}_k)(\mathbf{x}_k) + \mathcal{T}(\mathbf{x}_k)(\sigma_k)) \right),
\end{aligned} \tag{20}$$

and the errors satisfy

$$0 = \mathrm{Div}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}) \left(\mathbf{v}_\infty(\mathbf{x}_j) + \sum_{k=1}^{M} \mathcal{S}(\tilde{\mathbf{x}}_j + \mathbf{e}_{\mathbf{x}_j}, \tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) \left(-\mathcal{B}(\tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k})(\tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k}) + \mathcal{T}(\tilde{\mathbf{x}}_k + \mathbf{e}_{\mathbf{x}_k})(\tilde{\sigma}_k + e_{\sigma_k}) \right) \right). \tag{21}$$

A.1. Discretization

Using first-order IMEX time stepping [42], the provisional solution is found at the Gauss–Lobatto quadrature points $t_m = t_1 < \cdots < t_p = t_{m+1}$ by discretizing (18) and (20) as

$$\mathbf{x}_j^{m+1} = \mathbf{x}_j^m + \Delta t_m \left(\mathbf{v}_\infty(\mathbf{x}_j^m) + \sum_{k=1}^{M} \mathcal{S}(\mathbf{x}_j^m,\mathbf{x}_k^m)(-\mathcal{B}(\mathbf{x}_k^m)(\mathbf{x}_k^{m+1}) + \mathcal{T}(\mathbf{x}_k^m)(\sigma_k^{m+1})) \right), \quad m = 1, \ldots, p-1,$$

where $\Delta t_m = t_{m+1} - t_m$, and

$$\mathrm{Div}(\mathbf{x}_j^m) \left(\sum_{k=1}^{M} \mathcal{S}(\mathbf{x}_j^m,\mathbf{x}_k^m)(-\mathcal{B}(\mathbf{x}_k^m)(\mathbf{x}_k^{m+1}) + \mathcal{T}(\mathbf{x}_k^m)(\sigma_k^{m+1})) \right) = 0, \quad m = 1, \ldots, p-1.$$

For $m = 1, \ldots, p-1$, the SDC updates (19) and (21) are discretized as

$$\mathbf{e}_{\mathbf{x}_j}^{m+1} = \mathbf{e}_{\mathbf{x}_j}^{m} + \mathbf{r}_j^{m+1} - \mathbf{r}_j^{m} + \Delta t_m \left(\sum_{k=1}^{M} \mathcal{S}(\tilde{\mathbf{x}}_j^{m+1}, \tilde{\mathbf{x}}_k^{m+1}) \left(-\mathcal{B}(\tilde{\mathbf{x}}_k^{m+1})(\mathbf{e}_{\mathbf{x}_k}^{m+1}) + \mathcal{T}(\tilde{\mathbf{x}}_k^{m+1})(e_{\sigma_k}^{m+1}) \right) \right),$$

and

$$\begin{aligned} &\mathrm{Div}(\tilde{\mathbf{x}}_j^{m+1}) \left(\sum_{k=1}^{M} \mathcal{S}(\tilde{\mathbf{x}}_j^{m+1}, \tilde{\mathbf{x}}_k^{m+1}) \left(-\mathcal{B}(\tilde{\mathbf{x}}_k^{m+1})(\mathbf{e}_{\mathbf{x}_k}^{m+1}) + \mathcal{T}(\tilde{\mathbf{x}}_k^{m+1})(e_{\sigma_k}^{m+1}) \right) \right) \\ &\quad = -\mathrm{Div}(\tilde{\mathbf{x}}_j^{m+1}) \left(\mathbf{v}_\infty(\tilde{\mathbf{x}}_j^{m+1}) + \sum_{k=1}^{M} \mathcal{S}(\tilde{\mathbf{x}}_j^{m+1}, \tilde{\mathbf{x}}_k^{m+1}) \left(-\mathcal{B}(\tilde{\mathbf{x}}_k^{m+1})(\tilde{\mathbf{x}}_k^{m+1}) + \mathcal{T}(\tilde{\mathbf{x}}_k^{m+1})(\tilde{\sigma}_k^{m+1}) \right) \right). \end{aligned}$$

With this discretization of (21), we see that if $\mathbf{e}_{\mathbf{x}_k}^{m} = 0$ and $e_{\sigma_k}^{m} = 0$, then

$$\mathrm{Div}(\tilde{\mathbf{x}}_j^{m+1}) \left(\mathbf{v}_\infty(\tilde{\mathbf{x}}_j^{m+1}) + \sum_{k=1}^{M} \mathcal{S}(\tilde{\mathbf{x}}_j^{m+1}, \tilde{\mathbf{x}}_k^{m+1}) \left(-\mathcal{B}(\tilde{\mathbf{x}}_k^{m+1})(\tilde{\mathbf{x}}_k^{m+1}) + \mathcal{T}(\tilde{\mathbf{x}}_k^{m+1})(\tilde{\sigma}_k^{m+1}) \right) \right) = 0,$$

which means that

$$\mathrm{Div}(\tilde{\mathbf{x}}_j^{m}) \left(\sum_{k=1}^{M} \mathbf{v}(\tilde{\mathbf{x}}_j^{m}; \tilde{\mathbf{x}}_k^{m}) \right) = 0, \quad m = 1, \ldots, p.$$

In other words, by using the discretization (13), the fixed point of the SDC iteration exactly satisfies the inextensibility condition, in addition to converging to the solution of the fully-implicit collocation scheme of (18).

References

[1] Sebastian Aland, Sabine Egerer, John Lowengrub, Axel Voight, Diffuse interface models of locally inextensible vesicles in a viscous fluid, J. Comput. Phys. 277 (2014) 32–47.
[2] B.K. Alpert, Hybrid Gauss-trapezoidal quadrature rules, SIAM J. Sci. Comput. 20 (1999) 1551–1584.
[3] U.M. Ascher, S.J. Ruuth, B.T.R. Wetton, Implicit–explicit methods for time-dependent partial differential equations, SIAM J. Numer. Anal. 32 (1995) 797–823.
[4] J. Beaucourt, F. Rioual, T. Séon, T. Biben, C. Misbah, Steady to unsteady dynamics of a vesicle in a flow, Phys. Rev. E 69 (2004) 011906.
[5] T. Biben, C. Misbah, Tumbling of vesicles under shear flow within an advected-field approach, Phys. Rev. E 67 (2003) 031908.
[6] Thierry Biben, Klaus Kassner, Chaouqi Misbah, Phase-field approach to three-dimensional vesicle dynamics, Phys. Rev. E 72 (2005) 041921.
[7] K. Böhmer, H.J. Stetter, Defect Correction Methods, Theory and Applications, Springer-Verlag, New York, 1984.
[8] Sebastiano Boscarino, Error analysis of IMEX Runge–Kutta methods derived from differential-algebraic systems, SIAM J. Numer. Anal. 45 (4) (2007) 1600–1621.
[9] Anne Bourlioux, Anita T. Layton, Michael L. Minion, High-order multi-implicit spectral deferred correction methods for problems of reactive flow, J. Comput. Phys. 189 (2003) 651–675.
[10] Elizabeth L. Bouzarth, Michael L. Minion, A multirate time integrator for regularized Stokeslets, J. Comput. Phys. 229 (11) (2010) 4208–4224.
[11] Sunyoung Bu, Jingfang Huang, Michael L. Minion, Semi-implicit Krylov deferred correction methods for ordinary differential equations, in: Proceedings of the 15th American Conference on Applied Mathematics, AMATH'09, Stevens Point, Wisconsin, USA, World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 95–100.
[12] Sunyoung Bu, Jingfang Huang, Michael L. Minion, Semi-implicit Krylov deferred correction methods for differential algebraic equations, Math. Comput. 81 (2012) 2127–2157.
[13] I. Cantat, K. Kassner, C. Misbah, Vesicles in haptotaxis with hydrodynamical dissipation, Eur. Phys. J. E 10 (2003) 175–189.
[14] A.J. Christlieb, C.B. Macdonald, B.W. Ong, R.J. Spiteri, Revisionist integral deferred correction with adaptive step-size control, Commun. Appl. Math. Comput. Sci. 10 (1) (2015) 1–25.
[15] Andrew Christlieb, Benjamin Ong, Implicit parallel time integrators, J. Sci. Comput. 49 (2011) 167–179.
[16] Andrew Christlieb, Benjamin Ong, Jing-Mei Qiu, Comments on high-order integrators embedded within integral deferred correction methods, Commun. Appl. Math. Comput. Sci. 4 (2009) 27–56.
[17] Andrew J. Christlieb, Colin B. Macdonald, Benjamin W. Ong, Parallel high-order integrators, SIAM J. Sci. Comput. 32 (2) (2010) 818–835.
[18] Andrew J. Christlieb, Ronald D. Haynes, Benjamin W. Ong, A parallel space–time algorithm, SIAM J. Sci. Comput. 34 (5) (2012) 233–248.
[19] Qiang Du, Jian Zhang, Adaptive finite element method for a phase field bending elasticity model of vesicle membrane deformations, SIAM J. Sci. Comput. 30 (3) (2008) 1634–1657.
[20] A. Dutt, L. Greengard, V. Rokhlin, Spectral deferred correction methods for ordinary differential equations, BIT Numer. Math. 40 (2000) 241–266.
[21] Matthew Emmett, Michael L. Minion, Toward and efficient parallel in time method for partial differential equations, Commun. Appl. Math. Comput. Sci. 7 (1) (2012) 105–132.
[22] Germund G. Dahlquist, Werner Liniger, Olavi Nevanlinna, Stability of two-step methods for variable integration steps, SIAM J. Numer. Anal. 20 (5) (October 1983) 1071–1085.
[23] G. Ghigliotti, A. Rahimian, G. Biros, C. Misbah, Vesicle migration and spatial organization driven by flow line curvature, Phys. Rev. Lett. 106 (2011) 028101.
[24] Ernest Hairer, Gerhard Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, Springer, 1996.
[25] Ernest Hairer, Gerhard Wanner, Syvert Paul Nørsett, Solving Ordinary Differential Equations I: Nonstiff Problems, Springer, 1993.
[26] Anders C. Hansen, John Strain, On the order of deferred correction, Appl. Numer. Math. 61 (2011) 961–973.
[27] Po-Wen Hsieh, Ming-Chih Lai, Suh-Yuh Yang, Cheng-Shu You, An unconditionally energy stable penalty immersed boundary method for simulating the dynamics of an inextensible interface interacting with a solid particle, J. Sci. Comput. 64 (2) (2015) 289–316, http://dx.doi.org/10.1007/s10915-014-9933-y.

[28] Jingfang Huang, Jun Jia, Michael Minion, Accelerating the convergence of spectral deferred correction methods, J. Comput. Phys. 214 (2006) 633–656.
[29] Jingfang Huang, Ming-Chih Lai, Yang Xiang, An integral equation method for epitaxial step-flow growth simulations, J. Comput. Phys. 216 (2006) 724–743.
[30] Jingfang Huang, Jun Jia, Michael Minion, Arbitrary order Krylov deferred correction methods for differential algebraic equations, J. Comput. Phys. 221 (2) (2007) 739–760.
[31] Jun Jia, Jingfang Huang, Krylov deferred correction accelerated method of lines transpose for parabolic equations, J. Comput. Phys. 227 (2008) 1739–1753.
[32] B. Kaoui, N. Tahiri, T. Biben, H. Ez-Zahraouy, A. Benyoussef, G. Biros, C. Misbah, What dictates red blood cell shapes and dynamics in the microvasculature?, Phys. Rev. E (2011) 1–11.
[33] Yongsam Kim, Ming-Chih Lai, Simulating the dynamics of inextensible vesicles by the penalty immersed boundary method, J. Comput. Phys. 229 (12) (2010) 4840–4853.
[34] Aymen Laadhari, Pierre Saramito, Chaouqi Misbah, Computing the dynamics of biomembranes by combining conservative level set and adaptive finite element methods, J. Comput. Phys. 263 (2014) 328–352.
[35] Bengt Lindberg, Error estimation and iterative improvement for discretization algorithms, BIT Numer. Math. 20 (1980) 486–500.
[36] M.L. Minion, R. Speck, M. Bolten, M. Emmett, D. Ruprecht, Interweaving PFASST and parallel multigrid, arXiv:1407.6486, 2015.
[37] Michael L. Minion, Semi-implicit spectral deferred correction methods for ordinary differential equations, Commun. Math. Sci. 1 (3) (2003) 471–500.
[38] C. Misbah, Vacillating breathing and tumbling of vesicles under shear flow, Phys. Rev. Lett. 96 (2) (2006).
[39] H. Noguchi, D.G. Gompper, Tom C. Lubensky, Shape transitions of fluid vesicles and red blood cells in capillary flows, Proc. Natl. Acad. Sci. USA 102 (2005) 14159–14164.
[40] Benjamin Ong, Andrew Melfi, Andrew Christlieb, Parallel semi-implicit time integrators, arXiv:1209.4297v1, 2012.
[41] C. Pozrikidis, The axisymmetric deformation of a red blood cell in uniaxial straining Stokes flow, J. Fluid Mech. 216 (1990) 231–254.
[42] Bryan Quaife, George Biros, High-volume fraction simulations of two-dimensional vesicle suspensions, J. Comput. Phys. 274 (2014) 245–267.
[43] Abtin Rahimian, Ilya Lashuk, Shravan K. Veerapaneni, Aparna Chandramowlishwaran, Dhairya Malhotra, Logan Moon, Rahul Sampath, Aashay Shringarpure, Jeffrey Vetter, Richard Vuduc, Denis Zorin, George Biros, Petascale direct numerical simulation of blood flow on 200K cores and heterogeneous architectures, in: SC '10: Proceedings of the 2010 ACM/IEEE Conference on Supercomputing, IEEE Press, Piscataway, NJ, USA, 2010, pp. 1–12.
[44] Abtin Rahimian, Shravan Kumar Veerapaneni, George Biros, Dynamic simulation of locally inextensible vesicles suspended in an arbitrary two-dimensional domain, a boundary integral method, J. Comput. Phys. 229 (2010) 6466–6484.
[45] Abtin Rahimian, Shravan K. Veerapaneni, Denis Zorin, George Biros, Boundary integral method for the flow of vesicles with viscosity contrast in three dimensions, J. Comput. Phys. 298 (2015) 766–786.
[46] S. Ramanujan, C. Pozrikidis, Deformation of liquid capsules enclosed by elastic membranes in simple shear flow: large deformations and the effect of fluid viscosities, J. Fluid Mech. 361 (1998) 117–143.
[47] E. Sackmann, Supported membranes: scientific and practical applications, Science 271 (1996) 43–48.
[48] Robert D. Skeel, A theoretical framework for proving accuracy results for deferred corrections, SIAM J. Numer. Anal. 19 (1) (1982) 171–196.
[49] J.S. Sohn, Y.-H. Tseng, S. Li, A. Voigt, J.S. Lowengrub, Dynamics of multicomponent vesicles in a viscous fluid, J. Comput. Phys. 229 (January 2010) 119–144.
[50] Robert Speck, Daniel Ruprecht, Matthew Emmett, Matthias Bolten, Rolf Krause, A space–time parallel solver for the three-dimensional heat equation, in: Parallel Computing: Accelerating Computational Science and Engineering (CSE), in: Advances in Parallel Computing, vol. 25, IOS Press, 2014, pp. 263–272.
[51] Robert Speck, Daniel Ruprecht, Matthew Emmett, Michael Minion, Matthias Bolten, Rolf Krause, A multi-level spectral deferred correction method, BIT Numer. Math. 55 (3) (2014) 843–867, http://dx.doi.org/10.1007/s10543-014-0517-x.
[52] Robert Speck, Daniel Ruprecht, Michael Minion, Matthew Emmett, Rolf Krause, Inexact spectral deferred corrections, arXiv:1401.7824, 2014.
[53] Hans J. Stetter, Analysis of Discretization Methods for Ordinary Differential Equations, Springer Tracts in Natural Philosophy, vol. 23, 1973.
[54] S. Sukumaran, U. Seifert, Influence of shear flow on vesicles near a wall: a numerical study, Phys. Rev. Lett. E 64 (2001) 011916.
[55] S.K. Veerapaneni, D. Gueyffier, 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 (7) (2009) 2334–2353.
[56] Shravan K. Veerapaneni, Abtin Rahimian, George Biros, Denis Zorin, A fast algorithm for simulating vesicle flows in three dimensions, J. Comput. Phys. 230 (14) (2011) 5610–5634.
[57] Martin Weiser, Faster SDC convergence on non-equidistant grids by DIRK sweeps, Technical Report 13-30, ZIB, Takustr. 7, 14195, Berlin, 2013.
[58] Mathias Winkel, Robert Speck, Daniel Ruprecht, A high-order Boris integrator, J. Comput. Phys. 295 (2015) 456–474.
[59] Hong Zhao, Eric S.G. Shaqfeh, The dynamics of a vesicle in simple shear flow, J. Fluid Mech. 674 (2011) 578–604.
[60] Hong Zhao, Eric S.G. Shaqfeh, The dynamics of a non-dilute vesicle suspension in simple shear flow, J. Fluid Mech. 725 (2013) 709–731.