# A parallel algorithm for long-timescale simulation of concentrated vesicle suspensions in three dimensions

Dhairya Malhotra[a], Abtin Rahimian[b], Denis Zorin[a], George Biros[c]

[a]*Courant Institute of Mathematical Sciences, New York University, New York, NY 10012*
[b]*Department of Computer Science, University of Colorado, Boulder, CO 80309*
[c]*Institute for Computational Engineering and Science, University of Texas at Austin, Austin 78712*

**Abstract**

We develop a parallel boundary integral method for simulating highly concentrated vesicle suspensions in a Stokesian fluid. This method is an extension of our previous work [1]. The simulation of high volume fraction vesicle suspensions, which are representative of real biological systems (such as blood with 35% $\sim$ 50% volume fraction for RBC) presents several challenges. It requires computing accurate vesicle-vesicle interactions at length scales where standard quadratures are too expensive. The inter-vesicle separation can become arbitrarily small leading to vesicle collisions. Numerical errors can accumulate over time, making long-timescale simulations inaccurate.

We tackle these challenges by developing state-of-the-art parallel algorithms for efficient computation of boundary integrals and an adaptive time-stepping scheme. We have also developed algorithms for handling vesicle collisions, remeshing of vesicle surfaces and correcting drift in vesicle area and volume. We study the accuracy of our method when compared to a reference solution and show convergence with the spatial discretization order and the time step size. We visualize long-timescale simulations for periodic Taylor-Green vortex flow and sedimentation of polydisperse vesicle suspensions with thousands of vesicles. For these flows, we present strong and weak scaling results on thousands of CPU cores on the Stampede system at Texas Advanced Computing Center.

## 1. Introduction

Vesicles are closed phospholipid membranes suspended in a viscous solution. They are found in biological systems and play an important role in intracellular and intercellular transport. Artificial vesicles are used in a variety of drug-delivery systems and in the study of biomembrane mechanics. Vesicle-inspired mechanical models can be used to approximate red blood cell mechanics. For example, at equilibrium, vesicles and healthy red blood cells have a biconcave shape that corresponds to a minimal membrane bending energy. Under nonequilibrium conditions, as experienced in a simple shear flow, the best-studied features of red blood cell dynamics, such as tank-treading and tumbling motions, are shared with vesicles [2–4].

The vesicle evolution dynamics is characterized by an interplay between the membrane's elastic energy, surface inextensibility, vanishing in-plane shear resistance, and non-local hydrodynamic interactions. Simulation of vesicles is a challenging nonlinear free boundary value problem, not amenable to analytical solutions in all but a few simple cases; numerical simulations and experiments are the only options for the quantitative characterization of vesicle flows.

### *1.1. Contributions*

In this paper, we extend our previous work [1, 5, 6] on 3D vesicle flows to allow long-timescale simulation of concentrated vesicle suspensions in parallel. This requires a scalable boundary integration framework that can efficiently handle interactions between vesicles at different length scales (singular,

near-singular and far-field interactions). The complex dynamics of vesicle flows require time-adaptivity and algorithms to detect and handle collisions between vesicles. In each time step, we have to reparameterize the surface mesh and correct for small changes in the area and volume of the vesicles. We summarize these contributions as follows:

- We present a singular integration scheme that has $\mathcal{O}\left(p^5\right)$ setup cost and $\mathcal{O}\left(p^4\right)$ cost for each subsequent evaluation for a $p^{\text{th}}$-order discretization. With the original $\mathcal{O}\left(p^5\right)$ evaluation scheme, singular integration accounted for over 90% of the total runtime; however, with the new scheme, the setup and the singular integration stages together account for less than 20% of the total time.
- We have adapted the near-singular integration method of [7] for use with vesicles. The new scheme supports spherical harmonics representations, uses adaptive quadrature and is implemented in parallel. The modifications for spherical harmonics are not difficult but also not trivial.
- We accelerate the computation of far-field interactions using our PVFMM [8] library. The library is highly optimized using AVX vectorization and uses MPI for distributed memory parallelism. In addition, the library supports periodic boundary conditions and this allows us to simulate vesicles in periodic flows.
- We present an inexpensive method for estimating the error in each of the singular, near-singular and far-field integration schemes. We use this to adaptively adjust the order of the quadrature scheme and achieve the desired accuracy using the least amount of work.
- We introduce an algorithm for detecting collisions between vesicles. Such collisions happen due to the discretization errors in our numerical scheme. To avoid vesicle collisions, we have introduced a short range repulsion term in our model and developed an efficient algorithm for evaluating this repulsion force.
- In long-timescale simulations, errors can gradually change the surface area and volume of the vesicles. We need to correct for this drift by adjusting the area and volume of the vesicles in each time-step. We have developed an efficient algorithm to do this.
- The absence of in-plane shear resistance in vesicles necessitates a reparameterization scheme. The basic algorithm was presented in [6]. In this paper we introduce some modifications to this algorithm and also analyze the scheme for different parameter values. This has significantly improved the quality for our surface meshes.
- We implement an adaptive time-stepping scheme, which is based on the work of [9, 10] in two dimensions. We also present a new variation of this scheme that uses a more robust error estimate. This significantly reduces the solve time over a uniform time-stepping scheme.
- We present numerical results to show convergence of our method and study the dependence of the solution error on different parameter values when compared to a reference solution.
- We present scalability results up to several thousands CPU cores on the Stampede system at Texas Advanced Computing Center.

### *1.2. Limitations*

We restrict our attention to suspensions of vesicles in unbounded or periodic domains. We have ignored inertial terms, so the overall method is restricted to low Reynolds numbers. Only vesicles with spherical topology are considered and topological changes are not allowed. For general topologies one could, for example, use the boundary representation and singular integral quadrature introduced in [7]. We do not have any in-plane shear resistance in our formulation and this is a reasonable assumption for vesicles. However, for red blood cells (RBCs) and other cells, the shear resistance can not be ignored.

The spatial discretization order remains fixed during a simulation and it is the same for all vesicles. While we adaptively select the quadrature order in each time step, it is identical across all vesicles. This has the advantage that the algorithm can be applied to a large number of vesicles at once, resulting in better data locality and higher performance. However, for polydisperse simulations, each vesicle may require different discretization and quadrature orders depending on the size, shape and other properties (such as density, viscosity contrast and bending coefficient) of the vesicle.

We use a first-order time-stepping scheme. A high-order, adaptive scheme based on spectral deferred correction (SDC) was presented in [9, 10] for vesicle flows in 2D and can be adapted to our 3D solver.

The number of GMRES iterations in each time step is relatively large for high volume fraction flows, particularly with periodic boundary conditions. Our current scheme uses an analytical preconditioner constructed for a sphere [6]. From 2D simulations, we know that the inverse of the block diagonal part of the linear system would be a more effective preconditioner. We will report this extension in our future work.

### *1.3. Related work*

An integral equation formulation for vesicle suspensions using spherical harmonic discretization was discussed in [6] and was parallelized in [11]. However, that work did not support near-singular integration. We refer the reader to [6] for more related work on simulating vesicle flows in three-dimensions and to [1] for work on simulations with viscosity contrast. Work on simulating the flow of concentrated vesicle suspensions includes [12–17].

The singular integration scheme, which we have used, was first presented in [18] for Helmholtz problems and was applied to vesicles in [6]. A more efficient scheme (with $\mathcal{O}\left(p^4 \log p\right)$ cost) based on fast rotation of spherical harmonic expansions is discussed in [19]. For small discretization orders ($p \leq 36$), this scheme does not have any performance benefits over our scheme.

The near-singular integration scheme used in the current work was first presented in [7]. The original scheme was designed for B-spline patches and only discussed a sequential algorithm. This method was adapted to 2D vesicle flows in [17]. Other near-singular quadratures include the use of partition of unity along with polar coordinate transform [16]; the use of regularized kernel with corrections discussed in [20]; and the quadrature by expansion (QBX) scheme of [21] applied to simulation of rigid bodies in [22]. In [23], near interactions are computed through simple upsampling.

The fast multipole method (FMM) for gravitational N-body problems is discussed in [24]. FMM for the Stokes kernel includes the work of [25] and the STKFMMLIB3D library [26]. In the current work, we have used the kernel independent FMM (KIFMM) of [27] implemented in the PVFMM library [8]. A discussion of fast multipole accelerated boundary element methods can be found in [28].

Collision handling in 2D using a repulsion force is discussed in [29]. A purely kinematic approach for collision handling in 3D is discussed in [16]. In this approach, a mesh point is moved away from a surface if the separation between the mesh point and the surface is smaller than 2% of the cell radius. In [12], a repulsion based approach is discussed for concentrated emulsions in 3D. The repulsion force becomes infinite in magnitude as the surfaces come closer and decays exponentially with increasing distance between the surfaces. This is similar to the form of the repulsion force that we have used and has the drawback that it can introduce excessive stiffness in time-stepping. In [30] collisions are handled in 2D by enforcing an inequality constraint on a gap function which measures space-time intersection volume. This approach does not suffer from stiffness in time-stepping; however, solving the constraint equation can be expensive.

The adaptive time-stepping scheme used in our work was introduced in [9, 10] for 2D vesicle flows. They also discuss a high-order spectral deferred correction (SDC) time-stepping scheme.

### 1.4. Organization of the Paper

In Section 2, we introduce the mathematical model and the integral equation formulation for vesicles embedded in a Stokesian fluid. Then, in Section 3, we discuss the discretization and the algorithms for numerically solving the discretized equations. Finally, in Section 4, we present convergence and scalability results. In Table 1, we list some frequently used symbols for easy reference.

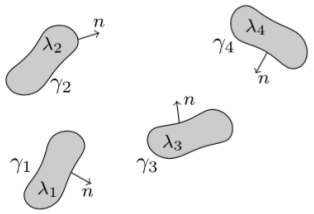
| Symbol | Definition |
|---|---|
| $S^2$ | Unit sphere |
| $(\theta, \phi)$ | Spherical angles |
| $Y_{nm}$ | Spherical harmonic function of degree $n$ and order $m$ |
| $p$ | Degree of spherical harmonic expansion |
| $q$ | Order of quadrature scheme |
| P | Projection operator from grid values to spherical harmonic coefficients. |
| Q | Operator to evaluate spherical harmonic expansion at grid points. |
| $N_\gamma$ | Number of vesicles |
| $\gamma_i$ | Boundary of $i^{th}$ vesicle |
| $W$ | Area element |
| $S_i$ | The single-layer Stokes operator over $i^{th}$ surface |
| $D_i$ | The double-layer Stokes operator over $i^{th}$ surface |
| $\mu$ | Viscosity of ambient fluid |
| $\mu_i$ | Viscosity of fluid in $i^{th}$ vesicle |
| $\rho$ | Density of ambient fluid |
| $\rho_i$ | Density of fluid in $i^{th}$ vesicle |
| $\sigma$ | Tension |
| $f_b$ | Bending force |

| Symbol | Definition |
|---|---|
| $f_\sigma$ | Tension force |
| $f_r$ | Repulsion force |
| $R_{repul}$ | Repulsion parameter: determines the range of repulsion force |
| $\boldsymbol{u}$ | Velocity |
| $\boldsymbol{u}^\infty$ | Background velocity |
| $T$ | Time-horizon of a simulation |
| $\Delta t$ | Time-step size |
| $\mathcal{E}$ | Error tolerance for time-adaptivity |
| $E$ | Reparameterization energy function |
| $a_n$ | Attenuation coefficients that define reparameterization energy function |
| $\Delta\tau_{max}$ | Maximum reparameterization step size |
| $n_p$ | Number of MPI processes |
| $\epsilon_{\text{GMRES}}$ | GMRES tolerance |
| $N_{\text{iter}}$ | Average GMRES iterations per solve |
| $N_{\text{Tstep}}$ | Number of time steps |
| $T_{solve}$ | Time to solution |
| $T_{setup}$ | Setup time |
| $T_{self}$ | Singular integration time |
| $T_{\text{near}}$ | Near-singular integration time |
| $T_{far}$ | Far-field integration time |
| $T_{repar}$ | Reparameterization time |

**Table 1:** *Index of frequently used symbols.*

## 2. Formulation

In this section we formally express the problem statement and give its boundary integral formulation. The detailed derivation of this formulation is given in [31]. The schematic of a typical domain is shown in Fig. 1.




$$\begin{cases} \lambda_i = \mu_i/\mu \quad \text{(viscosity contrast)}, \\ \alpha_i = (1+\lambda_i)/2 \quad \text{(velocity coefficient)}, \\ \omega_i = \text{The interior of the } i^{\text{th}} \text{ vesicle}, \\ \omega \ = \cup_i \omega_i \quad \text{(shaded area)}, \\ \gamma \ = \cup_i \gamma_i. \end{cases}$$

**Figure 1:** *The schematic of the domain.*

### 2.1. Differential Formulation

In the length scale of vesicles, the Reynolds number is very small and the effect of convective terms in the Navier-Stokes equation is negligible. In the limit, the fluid dynamics are governed by the Stokes and continuity equations

$$-\mu\Delta\boldsymbol{u}(\boldsymbol{x}) + \nabla p(\boldsymbol{x}) = 0 \quad \text{and} \quad \operatorname{div}\boldsymbol{u}(\boldsymbol{x}) = 0 \quad \text{for all } \boldsymbol{x} \in \mathbb{R}^3\backslash\omega, \tag{2.1}$$

where $\mathbb{R}^3\backslash\omega$ is the exterior region occupied by the suspending fluid, $\mu$ is the viscosity of the suspending fluid, $\boldsymbol{u}(\boldsymbol{x})$ denotes the fluid velocity, and $p(\boldsymbol{x})$ denotes the pressure. Letting $\mu_i$ denote the viscosity of the fluid inside the vesicles, Eq. (2.1) holds for the interior fluid, $\boldsymbol{x} \in \omega_i$, by replacing $\mu$ with $\mu_i$. We supplement Eq. (2.1) with the no-slip boundary condition on the interface of vesicles and matching far-field velocity as

$$\frac{\partial \boldsymbol{X}}{\partial t} = \boldsymbol{u}(\boldsymbol{X}) \quad \text{for all } \boldsymbol{X} \in \gamma, \quad \boldsymbol{u}(\boldsymbol{x}) \to \boldsymbol{u}^{\infty}(\boldsymbol{x}) \quad \text{as } \|\boldsymbol{x}\| \to \infty, \tag{2.2}$$

where $\boldsymbol{u}^{\infty}$ is the imposed far field velocity field. We use uppercase letters for Lagrangian variables and lowercase letters for Eulerian variables. Moreover, since the surface of the vesicles is locally inextensible, the surface divergence of the velocity field should vanish [6]. Therefore,

$$\operatorname{div}_{\gamma}\boldsymbol{u}(\boldsymbol{X}) = 0 \quad \text{for all } \boldsymbol{X} \in \gamma. \tag{2.3}$$

The balance of momentum on the membrane of vesicles implies that the jump in the surface traction is equal to the total force exerted by the interface onto the fluid, i.e.

$$[\![T\boldsymbol{n}]\!] = \boldsymbol{f}(\boldsymbol{X}) \quad \text{for all } \boldsymbol{X} \in \gamma, \tag{2.4}$$

where $T = -pI + \mu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)$ is the Cauchy stress tensor, $\boldsymbol{n}$ denotes the normal vector to the surface at point $\boldsymbol{X}$, $[\![\cdot]\!]$ denotes the jump across the interface, and $\boldsymbol{f}$ is the force exerted by the membrane onto the surrounding fluid. The interfacial force is composed of bending $\boldsymbol{f}_b$ and tensile $\boldsymbol{f}_\sigma$ forces. The source of the tensile force is the local inextensibility of the membrane. Eqs. (2.1–2.4) defines a complete set of (nonlinear) equations that is solvable for the velocity field, tension, and the evolution of vesicle interfaces.

*Interfacial forces.* Membrane's resistance to bending and extension give rise to interfacial forces, the derivation of which can be found in [32, 33]. For any $\boldsymbol{X} \in \gamma_i$ $(i = 1, \ldots, N_\gamma)$ we define

$$\boldsymbol{f}_b(\boldsymbol{X}) = -\kappa_b\left[\Delta_{\gamma_i}H + 2H(H^2 - K))\right]\boldsymbol{n}, \tag{2.5}$$

$$\boldsymbol{f}_\sigma(\boldsymbol{X}, \sigma) = \sigma\Delta_{\gamma_i}\boldsymbol{X} + \operatorname{grad}_{\gamma_i}\sigma, \tag{2.6}$$

where $\kappa_b$ is the membrane's bending modulus, $H$ and $K$ are respectively mean and Gaussian curvatures at $\boldsymbol{X}$, and $\sigma$ is the tension. Note that the bending force $\boldsymbol{f}_b$ depends only on the current configuration of each vesicle. On the other hand, the tensile force $\boldsymbol{f}_\sigma$ depends on both the configuration and the tension $\sigma$, which is the Lagrange multiplier to enforce the local inextensibility constraint, Eq. (2.3). Therefore, at any given configuration and background velocity, Eq. (2.3) needs to be solved to find the tension $\sigma$.

### 2.2. Boundary Integral Formulation

One can follow the standard approach of potential theory [31, 34] to reformulate Eqs. (2.1–2.4) as an integro-differential equation on the membrane of vesicles. It follows that for all $\boldsymbol{X} \in \gamma_i$ $(i = 1, \ldots, N_\gamma)$

we have

$$\boldsymbol{u}(\boldsymbol{X}) = \frac{1}{\alpha_i}\left(\boldsymbol{u}^{\infty}(\boldsymbol{X}) + \sum_{j=1}^{N_\gamma} \mathcal{S}_j[\boldsymbol{f}_b + \boldsymbol{f}_\sigma](\boldsymbol{X}) + \mathcal{D}_j[\boldsymbol{u}](\boldsymbol{X})\right), \tag{2.7}$$

$$\operatorname{div}_{\gamma_i} \boldsymbol{u}(\boldsymbol{X}) = 0, \tag{2.8}$$

$$\frac{\partial \boldsymbol{X}}{\partial t} = \boldsymbol{u}(\boldsymbol{X}), \tag{2.9}$$

where $\alpha_i = (1+\lambda_i)/2$, $\mathcal{S}_j[\boldsymbol{f}_b + \boldsymbol{f}_\sigma](\boldsymbol{X})$ and $\mathcal{D}_j[\boldsymbol{u}](\boldsymbol{X})$ denote the single-layer and double-layer convolution integrals over the $j^{\text{th}}$ surface with the interfacial force and velocity as respective densities, evaluated at point $\boldsymbol{X}$. The single-layer Stokes integral over the $i^{\text{th}}$ surface, evaluated at point $\boldsymbol{x}$ is defined as

$$\mathcal{S}_i[\boldsymbol{f}](\boldsymbol{x}) := \int_{\gamma_i} S(\boldsymbol{x},\boldsymbol{Y})\boldsymbol{f}(\boldsymbol{Y})\,\mathrm{d}\gamma(\boldsymbol{Y}), \quad S(\boldsymbol{x},\boldsymbol{Y}) = \frac{1}{8\pi\mu}\frac{1}{\|\boldsymbol{r}\|}\left(I + \frac{\boldsymbol{r}\otimes\boldsymbol{r}}{\|\boldsymbol{r}\|^2}\right), \tag{2.10}$$

where $\boldsymbol{r} := \boldsymbol{x} - \boldsymbol{Y}$, $I$ is the identity operator, $\otimes$ denotes the tensor product, and $\|\cdot\|$ is the Euclidean norm. The free-space double-layer integral over the $i^{\text{th}}$ surface, evaluated at point $\boldsymbol{x}$ is defined as

$$\mathcal{D}_i[\boldsymbol{u}](\boldsymbol{x}) := \int_{\gamma_i} D_i(\boldsymbol{x},\boldsymbol{Y})\boldsymbol{u}(\boldsymbol{Y})\,\mathrm{d}\gamma(\boldsymbol{Y}), \quad D_i(\boldsymbol{x},\boldsymbol{Y}) = -\frac{3(1-\lambda_i)}{4\pi}\frac{(\boldsymbol{r}\cdot\boldsymbol{n})(\boldsymbol{r}\otimes\boldsymbol{r})}{\|\boldsymbol{r}\|^5}. \tag{2.11}$$

The subscript $i$ for the double-layer kernel $D_i$ is to emphasize its dependence on the normal to surface $\boldsymbol{n}(\boldsymbol{Y})$ and the viscosity contrast of the $i^{\text{th}}$ vesicle $\lambda_i$. Given the initial distribution of vesicles, the Eqs. (2.7–2.11) may be used to solve for their evolution over time.

### 2.3. *Galerkin Formulation*

Using the spherical harmonic functions $Y_{nm}$ (defined in Eq. (A.1)) as the basis set for $L^2(\mathbb{S}^2)$, one can represent the position and tension in this basis set

$$\boldsymbol{X} = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} \widehat{\boldsymbol{X}}_{nm} Y_{nm} \qquad \text{and} \qquad \sigma = \sum_{n=0}^{\infty}\sum_{m=-n}^{n} \widehat{\sigma}_{nm} Y_{nm}. \tag{2.12}$$

Letting $(\cdot,\cdot)$ denote the inner product in this space — in which vector fields are treated element-wise — the Galerkin method seeks the solution to Eqs. (2.7–2.11) by

$$\alpha_i\,(\boldsymbol{u}, Y_{nm}) = (\boldsymbol{u}^{\infty}, Y_{nm}) + \sum_{j=1}^{N_\gamma}\left(\mathcal{S}_j[\boldsymbol{f}_b + \boldsymbol{f}_\sigma], Y_{nm}\right) + \left(\mathcal{D}_j[\boldsymbol{u}], Y_{nm}\right), \tag{2.13}$$

$$(\operatorname{div}_{\gamma_i}\boldsymbol{u}, Y_{nm}) = 0, \quad \text{for all } i = 1,\ldots,N_\gamma, \tag{2.14}$$

$$\left(\frac{\partial \boldsymbol{X}}{\partial t}, Y_{nm}\right) = (\boldsymbol{u}, Y_{nm}) \tag{2.15}$$

for all $n = 0, 1, \ldots$ and $|m| \le n$. In Section 3, we first outline our approach to perform computation over the surface of vesicles using the spherical harmonics. Afterwards, we outline a time stepping method to update the position of vesicles and then look at different schemes to solve the resulting linear system.

## 3. Numerical Algorithms

We discretize the Galerkin formulation discussed in the previous section. We discuss the spatial discretization in Section 3.1 and algorithms for computing the Stokes single-layer and double-layer potentials from the vesicle surface in Section 3.2. In Section 3.3 we present an algorithm for detecting collisions between vesicles and introduce a repulsion term in our formulation to avoid such collisions.

We present an algorithm for correcting the drift in area and volume of the vesicles in Section 3.4. In Section 3.5, we discuss the reparameterization algorithm and analyze the scheme for different choices of the attenuation coefficient. Then, we present a first order semi-implicit time-stepping scheme in Section 3.6 and discuss an algorithm for selecting the optimal time step size in Section 3.7. Finally, we summarize the overall algorithm in Section 3.8.

### *3.1. Spatial Discretization*

We assume that each surface $\gamma$ is smooth and homeomorphic to a sphere. Therefore, we can construct a $C^\infty$ map from points on $\gamma$ to points on the surface of the unit sphere $\mathbb{S}^2$. This mapping is not unique and can affect the magnitude of truncation errors when the surface is discretized. We will discuss these issues later in Section 3.5. The surface of the unit sphere can be parameterized by the spherical angles $(\theta, \phi) \in [0, \pi] \times [0, 2\pi)$. We approximate a function $f$ on $\gamma$ (mapped to $\mathbb{S}^2$) using the spherical harmonic basis $Y_{nm}$ up to degree $p$

$$f(\theta, \phi) \approx \sum_{n=0}^{p} \sum_{m=-n}^{n} \widehat{f}_{nm} Y_{nm}(\theta, \phi) \tag{3.1}$$

where, $\widehat{f}_{nm}$ are the coefficients in the spherical harmonic expansion. Using orthonormality of the spherical harmonic basis, we can determine these coefficients using the relation $\widehat{f}_{nm} = (f, Y_{nm})$. We construct such spherical harmonic approximations for the surface position, the tension and bending forces and the velocity of surface points.

*3.1.1. Evaluation on Nodal Basis.* When computing integrals over a surface $\gamma$, we require a nodal basis representation of the surface instead of the spherical harmonic basis discussed above. For $q^{\text{th}}$-order quadratures, we discretize the spherical angles using a $(q+1) \times 2q$ grid of points $(\theta_i, \phi_j)$ given by

$$\begin{aligned} \theta_i &= \cos^{-1} x_i && \text{for } i = 0, \cdots, q \\ \phi_j &= \pi / q \; j && \text{for } j = 0, \cdots, 2q-1 \end{aligned}$$

where, $x_i$ are the roots of the Legendre polynomial of degree $q+1$. We refer to this as the $q$-grid. We evaluate the spherical harmonic representation $\widehat{f}$ at points on the $q$-grid as follows

$$f_{ij} = \sum_{n=0}^{p} \sum_{m=-n}^{n} \widehat{f}_{nm} \, Y_{nm}(\theta_i, \phi_j) \qquad \text{for all} \quad i = 0, \cdots, q \quad \text{and} \quad j = 0, \cdots, 2q-1. \tag{3.2}$$

We define a linear operator $Q$, which implements the above computation, so that $f = Q \, \widehat{f}$. Since the spherical harmonic basis functions $Y_{nm}$ are products of the associated Legendre polynomials $P_{nm}(\cos\theta)$ and the Fourier basis functions $e^{im\phi}$, the above transform can be computed efficiently using a tensor product rule. We first evaluate the associated Legendre polynomials at $x_i = \cos\theta_i$ for $i = 0, \cdots, q$. This is followed by computing $q+1$ discrete Fourier transforms in $\phi_i$. The method requires $\mathcal{O}\left(p^2 q + p q^2\right)$ work; with $\mathcal{O}\left(p^2 q\right)$ work for computing the associated Legendre polynomials and $\mathcal{O}\left(p q^2\right)$ work for the Fourier transform. We could use FFT for computing the Fourier transform and FLT (Fast Legendre Transform) for evaluating the associated Legendre polynomials to reduce the complexity to $\mathcal{O}\left(q^2 \log^2 q\right)$ work (when $q \geq p$ and using zero padding); however, this does not provide any noticeable improvement in performance for the small discretization orders ($p, q \leq 32$) used in this work.

We also use Eq. (3.2) to compute derivatives in $\theta$ and $\phi$ by replacing $Y_{nm}$ with $\partial Y_{nm}/\partial\theta$ and $\partial Y_{nm}/\partial\phi$ respectively. Evaluating the derivatives on the $q$-grid from a spherical harmonic approximation of degree $p$ requires $\mathcal{O}\left(p^2 q + p q^2\right)$ work.

*3.1.2. Projection to Spherical Harmonic Basis.* We often have to compute a projection to the spherical harmonic space $\widehat{f}$ from function values $f_{ij}$ on the $q$-grid where, $q \geq p$. To do this, we use the relation $\widehat{f}_{nm} = (f, Y_{nm})$ and compute the inner product using a quadrature rule

$$\widehat{f}_{nm} = \sum_{i=0}^{q} \sum_{j=0}^{2q-1} f_{ij}\, Y_{nm}(\theta_i, \phi_j)\, \frac{\pi}{q}\, w_i \qquad \text{for all} \quad n = 0, \cdots, p \quad \text{and} \quad |m| \leq n. \tag{3.3}$$

Here, we have used the Gauss-Legendre quadrature rule (with weights $w_i$) to integrate in $\phi$ and trapezoidal rule (with weights $\pi/q$) to integrate in $\theta$. As before for Eq. (3.2), this sum can be evaluated efficiently using a tensor product rule and this requires $\mathcal{O}\left(p^2 q + p q^2\right)$ work. We define a linear operator $P$, which computes the above projection, so that $\widehat{f} = P\, f$.

*3.2. Stokes Layer Potentials*

In our boundary integral formulation, we evaluate single- and double-layer potential from $N_\gamma$ surfaces, with position $\boldsymbol{X}$, single-layer density $\boldsymbol{f}_s$ and double-layer density $\boldsymbol{f}_d$

$$\boldsymbol{u}(\boldsymbol{X}) = \sum_{k=1}^{N_\gamma} \mathcal{S}_k[\boldsymbol{f}_s](\boldsymbol{X}) + \mathcal{D}_k[\boldsymbol{f}_d](\boldsymbol{X}). \tag{3.4}$$

We evaluate this potential numerically on the $p$-grid for each surface. For the Galerkin formulation, we then compute a projection to the spherical harmonic space $\widehat{\boldsymbol{u}}_{nm} = (\boldsymbol{u}, Y_{nm})$. In the remainder of this section, we will only discuss computation of the single-layer potential; however, the algorithms are also applicable to computing the double-layer potential.

We consider computation of the Stokes single-layer potential $\mathcal{S}_\gamma[\boldsymbol{f}](\boldsymbol{Y})$ from a single surface $\gamma$ at a target point $\boldsymbol{Y}$. We can express the integral over the surface $\gamma$ as an integral over the spherical angles $\theta$ and $\phi$ as follows

$$\mathcal{S}_\gamma[\boldsymbol{f}](\boldsymbol{Y}) = \int_\gamma S(\boldsymbol{Y}, \boldsymbol{X})\, \boldsymbol{f}\ \mathrm{d}\gamma \ = \ \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} S(\boldsymbol{Y}, \boldsymbol{X}(\theta,\phi))\, \boldsymbol{f}(\theta,\phi)\, W(\theta,\phi)\ \mathrm{d}\phi\ \mathrm{d}\theta \tag{3.5}$$

where, $W(\theta,\phi) = \sqrt{EG - F^2}$ is the area element of the surface (with $E$, $F$ and $G$ denoting the coefficients of the first fundamental form of $\gamma$). When $\boldsymbol{Y}$ is not on the surface, then the integrand is smooth and standard quadratures (Gauss-Legendre quadrature for $\theta$ and trapezoidal quadrature for $\phi$ directions) are sufficient. We discretize the integral in Eq. (3.5) by a quadrature rule on the $q$-grid

$$\mathcal{S}_\gamma[\boldsymbol{f}](\boldsymbol{Y}) \approx \sum_{i=0}^{q} \sum_{j=0}^{2q-1} S(\boldsymbol{Y}, \boldsymbol{X}_{ij})\, \boldsymbol{f}_{ij}\, W_{ij}\, \frac{\pi}{q}\, w_i \tag{3.6}$$

where, $w_i$ are the Gauss-Legendre quadrature weights and $\boldsymbol{X}_{ij}$, $\boldsymbol{f}_{ij}$ and $W_{ij}$ are the surface position, the density function and the area element evaluated on the $q$-grid. This computation requires $\mathcal{O}\left(q^2\right)$ work for each evaluation point. The method is spectrally convergent in $q$; however, to be accurate, it requires that the distance between the points on the $q$-grid should be smaller than the distance between the surface and the evaluation points. Therefore, for $h = \min|\boldsymbol{X}(\theta,\phi) - \boldsymbol{Y}|$, we must have $q = \mathcal{O}\left(h^{-1}\right)$. The method becomes prohibitively expensive as $h$ becomes smaller and does not converge for $h = 0$. Next, we discuss special quadratures for computing these singular and near-singular integrals efficiently.

*3.2.1. Singular Integration.* We use the algorithm discussed in [6] for the Stokes single-layer potential and extended to the Stokes double-layer potential in [1]. This singular integration scheme is spectrally convergent for both single- and double-layer potentials. Below, we briefly summarize this algorithm for computing $\widehat{\boldsymbol{u}}_{nm} = (\mathcal{S}_\gamma[\boldsymbol{f}], Y_{nm})$ for a single surface $\gamma$.

We define the linear operators $\mathrm{R}(\theta, \phi)$, which transform the coefficients in a spherical harmonic expansion to the coefficients in a rotated coordinate space with the north pole at $(\theta, \phi)$. The construction of these operators is discussed in [35]. For a spherical harmonic discretization of degree $p$, the application of the operator requires $\mathcal{O}\left(p^3\right)$ work for each surface. We use these operators to compute the spherical harmonic expansions of the surface position in rotated coordinate space: $\widehat{\boldsymbol{X}}^{ij} = \mathrm{R}(\theta_i, \phi_j)\ \widehat{\boldsymbol{X}}$ for each point $(\theta_i, \phi_j)$ on the $p$-grid. Then, we evaluate the spherical harmonic expansions on the $q$-grid: $\boldsymbol{X}^{ij} = Q\ \widehat{\boldsymbol{X}}^{ij}$. Similarly, we compute the single-layer density on the $q$-grid in rotated coordinate space: $\boldsymbol{f}^{ij} = Q\ \mathrm{R}(\theta_i, \phi_j)\ \widehat{\boldsymbol{f}}$. We also compute the area elements $W^{ij}$ from $\widehat{\boldsymbol{X}}^{ij}$. This requires evaluating the derivatives $\partial \boldsymbol{X}/\partial\theta$ and $\partial \boldsymbol{X}/\partial\phi$ on the $q$-grid. Now, we compute the potential $\boldsymbol{u}_{ij} = \boldsymbol{u}(\theta_i, \phi_j)$ on the $p$-grid

$$\boldsymbol{u}_{ij} = \left(\Lambda_q \circ W^{ij} \circ S(\boldsymbol{X}_{ij}, \boldsymbol{X}^{ij})\right) \cdot \boldsymbol{f}^{ij} \qquad \text{for all} \quad i = 0, \cdots, p \quad \text{and} \quad j = 0, \cdots, 2p-1 \tag{3.7}$$

where, $\Lambda_q$ are the quadrature weights from the scheme of Graham-Sloan [18] for evaluating singular integrals at the pole. Here, we are computing the Stokes single-layer potential at the north pole $\boldsymbol{X}_{ij}$ from each point on the $q$-grid with position $\boldsymbol{X}^{ij}$ and density $\boldsymbol{f}^{ij}$ scaled by the area elements and the quadrature weights at each grid point. Finally, we compute the projection from $\boldsymbol{u}$ on the $p$-grid to the spherical harmonic space $\widehat{\boldsymbol{u}} = P\ \boldsymbol{u}$.

The above algorithm uses a $q$-grid for the singular quadrature rule instead of the $p$-grid used in [6]. We observed that depending on the desired accuracy of the result, we often need to use an upsampled grid such that $q > p$. We will discuss choosing the optimal value for $q$ in Section 3.2.4. The algorithm requires $\mathcal{O}\left(p^5\right)$ work for computing all the rotations, $\mathcal{O}\left(p^2(p^2q + pq^2)\right)$ work for evaluating spherical harmonic expansions on $q$-grid and $\mathcal{O}\left(p^2q^2\right)$ work for evaluating the Stokes operator and computing the weighted inner-product with the density. Overall, this requires $\mathcal{O}\left(p^5 + p^3q^2\right)$ work each time we compute $\widehat{\boldsymbol{u}}$.

In our semi-implicit time-stepping scheme, we use GMRES to solve for the new surface position. Each GMRES iteration involves computing $\boldsymbol{u}$ for the same surface position $\boldsymbol{X}$ but different densities $\boldsymbol{f}$. Therefore, we modify the above algorithm to instead precompute the operator matrix

$$\begin{aligned}
\mathbf{S}_{ij} &= \left(\Lambda_q \circ W^{ij} \circ S(\boldsymbol{X}_{ij}, \boldsymbol{X}^{ij})\right) \cdot Q\ \mathrm{R}(\theta_i, \phi_j) \qquad \text{for all} \quad i = 0, \cdots, p \quad \text{and} \quad j = 0, \cdots, 2p-1 \\
\widehat{\mathbf{S}} &= P\ \mathbf{S}
\end{aligned}$$

where, $\widehat{\mathbf{S}}$ is a $(p+1)^2 \times (p+1)^2$ matrix. In each GMRES iteration, we can now compute $\widehat{\boldsymbol{u}} = \widehat{\mathbf{S}}\ \widehat{\boldsymbol{f}}$. Computing $\widehat{\mathbf{S}}$ still requires $\mathcal{O}\left(p^5 + p^3q^2\right)$ work per surface; however, each subsequent application of $\widehat{\mathbf{S}}$ only requires $\mathcal{O}\left(p^4\right)$ work. This results in a significant improvement in performance over the original scheme.

*3.2.2. Near-Singular Integration.* For computing interactions between a surface $\gamma$ and a target point $\boldsymbol{Y}$, such that $\boldsymbol{Y}$ is not on the surface but at a distance smaller than $h_n$ from the surface, we use a near-singular integration scheme. The scheme is adapted from the method of [7]. The value of $h_n$ depends on the order $q$ of the Nyström scheme used in Eq. (3.6). For a given $q$, the optimal choice for $h_n$ has to be determined empirically; however, we observed that the optimal value can be estimated by the relation $h_n = \sqrt{A/q}$ where, $A$ is the maximum surface area of any vesicle. For this choice of $h_n$, we still get spectral convergence with the Nyström scheme (for target points at a distance $h_n$ or greater from the surface), since distance between points on the $q$-grid $\mathcal{O}\left(\sqrt{A}/q\right)$ decreases faster than $h_n$ as we increase $q$. For points closer than $h_n$ to the surface, the near-singular integration algorithm has the following sequence of steps.

(a) *Identifying Near Points:* The first step in the near-singular integration scheme is to identify, all pairs of surface and target points $(\gamma_i, \boldsymbol{Y})$ that are separated by a distance smaller than $h_n$. We can do this

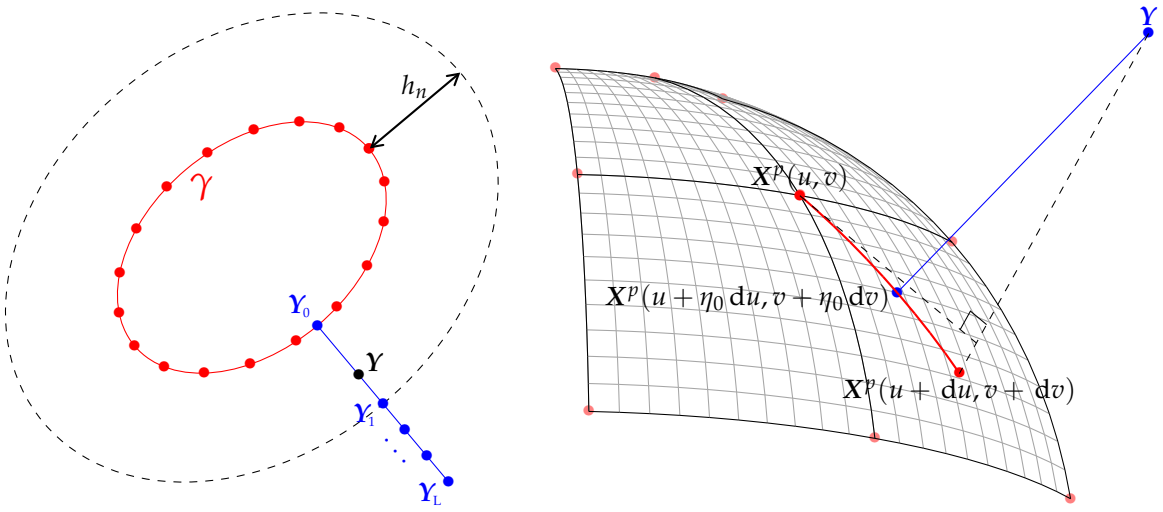



**Figure 2:** *Left: 2D schematic of vesicle surface ($\gamma$) and the surface discretization points. The Nyström scheme is accurate for target points at a distance $h_n$ or greater from the surface. To compute velocity at the point $\mathbf{Y}$, we determine the nearest point $\mathbf{Y}_0$ on the surface. We compute the velocity at $\mathbf{Y}_0$ using a singular quadrature scheme and interpolation on the surface. Using the Nyström scheme, we compute the velocity at a sequence of points $\mathbf{Y}_1, \cdots, \mathbf{Y}_L$ on the line through $\mathbf{Y}_0$ and $\mathbf{Y}$. We compute the velocity at $\mathbf{Y}$ by interpolating the velocity at $\mathbf{Y}_0, \cdots, \mathbf{Y}_L$. Right: Quadratic patch created by interpolating a $3 \times 3$ grid of surface points. We show the first iteration for computing the projection of the target point $\mathbf{Y}$ on this patch. We start at the center of the patch with $(u, v) = (0, 0)$ and compute the update $(\mathrm{d}u, \mathrm{d}v)$ by approximating the surface by the tangent plane passing through $\mathbf{X}^p(u, v)$. Then we search for the surface point nearest to $\mathbf{Y}$ along the line $(u, v) - (u + \mathrm{d}u, v + \mathrm{d}v)$ in parameter space. We iterate until we reach the edge of the patch or the updates are small enough ($|(\mathrm{d}u, \mathrm{d}v)| \leq$ 1E-6).*

by comparing the distance between every target point and every point on the discretized surface ($q$-grid). For $N_{trg}$ target points and $N_\gamma$ surfaces, this requires $\mathcal{O}\left(N_{trg} N_\gamma\right)$ work and can be very expensive when $N_\gamma$ and $N_{trg}$ are large. A more efficient method is to sort all the surface discretization points, then for each target point we can find the surface points close to it by searching in a sorted array. This can be implemented in a number of ways, such as: using radix sort to bin points in $h_n \times h_n \times h_n$ size boxes; using an octrees of depth $\log_2 {h_n}^{-1}$; or sorting points on a space-filling curve. In our implementation, we use the last approach since it is easy to parallelize. We compute the Morton Id with depth $\log_2 {h_n}^{-1}$ for each surface point and sort these using a parallel sorting algorithm [36]. The parallel sorting algorithm requires $\mathcal{O}\left(N/n_p \log N/n_p + N/n_p \log n_p\right)$ time for $N$ points on $n_p$ processes. For each target point, we compute its Morton Id and also the neighboring 26 Morton Ids. We find all the surface points with these 27 Morton Ids using binary search and compare the distance between these surface points and the target point. We make pairs of each vesicle surface and target point such that the surface has a discretization point closer than $h_n$ from the target point.

To do this in parallel on distributed memory systems, during the local binary search, the target points must be on the same MPI process as the surface points. Therefore, we also sort the target points by their Morton Id and partition them across MPI processes using the same partitioning as for the surface points. Another advantage of doing this is that we can now process all the target points with the same Morton Id together and avoid repeated binary searches in the array of surface points. For our parallel implementation, we also have to add ghost Morton Ids to the array of surface points so that for each target point we have all the 27 adjacent Morton Ids available locally on the MPI process. We identify these ghost Morton Ids and communicate them using point-to-point communication. In addition, once we form pairs of surface and target points, we have to

send these to the process where the surface originated before the parallel sort. The rest of the near-singular integration algorithm can then proceed independently on each MPI process.

(b) *Projection on the Surface:* For each pair of surface $\gamma$ and near target point $\boldsymbol{Y}$, we determine the projection $\boldsymbol{Y}_0$ of the target point on the surface (see Fig. 2). To do this, we determine the surface discretization point that is closest to $\boldsymbol{Y}$ and select the $3 \times 3$ grid of points in the surface mesh around this point. If one of the poles is the nearest point, then we select the pole and eight other points adjacent to the pole to form the $3 \times 3$ grid. We create a quadratic surface interpolant $\boldsymbol{X}^p(u, v)$ from this grid, such that $\boldsymbol{X}^p(0, 0)$ is the nearest grid point to $\boldsymbol{Y}$. Now, we use an iterative scheme to find the point on this interpolant that is closest to the target point $\boldsymbol{Y}$. We start from the center of the patch ($u = 0$ and $v = 0$). In each iteration, we linearize the interpolant $\boldsymbol{X}^p$ around $u$ and $v$ so that

$$\boldsymbol{X}^p(u + \mathrm{d}u, v + \mathrm{d}v) \approx \boldsymbol{X}^p(u, v) + \nabla \boldsymbol{X}^p(u, v) \, [\mathrm{d}u \;\; \mathrm{d}v]^T \tag{3.8}$$

We want to find the least squares solution to $\boldsymbol{X}^p(u + \mathrm{d}u, v + \mathrm{d}v) = \boldsymbol{Y}$. Substituting in the above equation and solving for $[\mathrm{d}u \;\; \mathrm{d}v]$, we have

$$[\mathrm{d}u \;\; \mathrm{d}v]^T = (\nabla \boldsymbol{X}^p)^+ \, (\boldsymbol{Y} - \boldsymbol{X}^p) \tag{3.9}$$

where, $(\nabla \boldsymbol{X}^p)^+$ is the pseudo-inverse of $\nabla \boldsymbol{X}^p(u, v)$. To avoid overshooting (due to the curvature of the surface) we now search along the line $(u, v) - (u + \mathrm{d}u, v + \mathrm{d}v)$ in parameter space; i.e. we try to find $\eta_0 \in [0, 1]$ such that $\boldsymbol{X}^p(u + \eta_0 \, \mathrm{d}u, v + \eta_0 \, \mathrm{d}v)$ is nearest to $\boldsymbol{Y}$. We build a quadratic interpolant $p(\eta)$ such that $p(\eta) = |\boldsymbol{X}^p(u + \eta \, \mathrm{d}u, v + \eta \, \mathrm{d}v) - \boldsymbol{Y}|$ for $\eta = 0, 0.5, 1$ and compute $\eta_0 = \arg\min_{\eta \in [0,1]} p(\eta)$. We update $(u, v) \mathrel{+}= \eta_0 (\mathrm{d}u, \mathrm{d}v)$ and repeat the above process until we reach the edge of the patch or the updates $|(\mathrm{d}u, \mathrm{d}v)|$ are smaller than a given tolerance (about 1E-6). When the method converges, we obtain the projection $\boldsymbol{Y}_0 = \boldsymbol{X}^p(u, v)$ of $\boldsymbol{Y}$ on the surface $\gamma$. In our experiments, the method converged to single-precision accuracy in about 10 iterations.

(c) *Interpolation:* For each surface $\gamma$, we compute the Stokes singular integral $\widehat{\boldsymbol{u}}$ as discussed in Section 3.2.1 and evaluate it on the $q$-grid. Now, for each near point ($\boldsymbol{Y}$) of $\gamma$, we interpolate the singular potential at the projection $\boldsymbol{Y}_0$ using the quadratic surface interpolant described above to obtain $\boldsymbol{u}_0 = \boldsymbol{u}(\boldsymbol{Y}_0)$. We construct a set of points $\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_L$ on the line through $\boldsymbol{Y}_0$ and $\boldsymbol{Y}$; distributed evenly between distances $h_n$ and $2h_n$ from point $\boldsymbol{Y}_0$. Since these points are at a distance greater than or equal to $h_n$, we can use the Nyström scheme to compute the potential $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_L$ at the points $\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_L$ respectively. We now construct a Lagrange polynomial interpolant of degree $L$ for the potential $\{\boldsymbol{u}_0, \cdots, \boldsymbol{u}_L\}$ at points $\{\boldsymbol{Y}_0, \cdots, \boldsymbol{Y}_L\}$. Finally, we evaluate this interpolant at $\boldsymbol{Y}$ to obtain the potential $\boldsymbol{u}_{\boldsymbol{Y}} = \boldsymbol{u}(\boldsymbol{Y})$. In our implementation, we have used the interpolation order $L = 8$.

The algorithm assumes that the interpolation points $\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_L$ are at a distance greater than $h_n$ from $\gamma$ so that the potential at these points is smooth. However, this may not be true when the surface has large deformations. In this case, we need to use a larger $q$ so that $h_n$ is smaller and therefore, the interpolation points are separated from the surface by a distance greater than $h_n$. This requires adaptively choosing the appropriate $q$ and is discussed in Section 3.2.4.

The steps (a) and (b) are part of the setup phase and are compute once per time step; however, step (c) must be evaluated each time the layer potential is computed. For $N_\gamma$ surfaces, $N_{trg}$ near target points and $q^{\text{th}}$-order quadratures (where $q > p$), step (a) requires $\mathcal{O}\left(N_\gamma q^2/n_p \log\left(N_\gamma q^2/n_p\right) + N_\gamma q^2/n_p \log n_p\right)$ time, step (b) requires $\mathcal{O}\left(N_{trg}/n_p\right)$ time and step (c) requires $\mathcal{O}\left(N_{trg} L q^2/n_p\right)$ time on $n_p$ processors. Here, $L = 8$ is the order of the Lagrange interpolation discussed above. In our formulation, the target points are the same as the surface discretization points (the $p$-grid) and therefore $N_{trg} = \mathcal{O}\left(N_\gamma p^2\right)$.

*3.2.3. Far-Field Integration.* Computing the summation in Eq. (3.4), using the Nyström integration scheme in Eq. (3.6), requires $\mathcal{O}\left(N_\gamma^2 p^2 q^2\right)$ work for $N_\gamma$ surfaces with spherical harmonic discretization of degree $p$ and $q^{\text{th}}$-order quadrature scheme. This is an N-body problem with $\mathcal{O}\left(N_\gamma p^2\right)$ target points and $\mathcal{O}\left(N_\gamma q^2\right)$ source points; with the source densities scaled by the area element and the quadrature weights.

We can accelerate the above computation by using the Fast Multipole Method (FMM) [24] and compute solutions in $\mathcal{O}\left(N_\gamma(q^2+p^2)\right)$ work. We use our PVFMM library [8], which is an optimized, parallel implementation of the Kernel Independent FMM scheme of [27]. For $N = N_\gamma(p^2+q^2)$ source and target points, our algorithm requires $\mathcal{O}\left(N/n_p \log(N/n_p) + N/n_p \log n_p\right)$ setup time for tree construction and $\mathcal{O}\left(N/n_p + (N/n_p)^{2/3} \log n_p\right)$ time for each subsequent evaluation on $n_p$ processes. In addition to free-space boundary conditions, the library can also compute periodic solutions by creating an infinite periodic tiling of the source density. The far-field computation is by far the most expensive computation in our scheme and the use of the PVFMM library allows us to compute solutions efficiently and scale our scheme to a large number of compute nodes.

Using FMM requires that we compute interactions between all pairs of source and target points. Therefore, we have to use direct summation to compute interactions with the near target points for each surface, subtract it from the FMM solution and then add the contributions from the singular and near-singular integration schemes discussed above. Computing these summations requires an additional $\mathcal{O}\left(N_{trg} q^2\right)$ work, where $N_{trg} = \mathcal{O}\left(N_\gamma p^2\right)$; therefore, we still retain linear work complexity in $N_\gamma$. While it may be more efficient to exclude these self- and near-interactions when computing the FMM sum, it is very complicated and would require implementing an entirely new FMM library.

*3.2.4. Integration Error.* We can estimate the error in the integration scheme discussed above by checking with a known eigenvalue of the double-layer operator. We evaluate the following double-layer integral

$$\boldsymbol{u}(\boldsymbol{Y}) = \sum_{k=1}^{N_\gamma} \mathcal{D}_k[\boldsymbol{f}](\boldsymbol{Y}) \tag{3.10}$$

where, the density $\boldsymbol{f} = 1$ on each surface. For closed and non-overlapping surfaces, this integral has an analytical solution: $\boldsymbol{u}(\boldsymbol{Y}) = 1$ when $\boldsymbol{Y}$ is enclosed by any surface; $\boldsymbol{u}(\boldsymbol{Y}) = 0.5$ when $\boldsymbol{Y}$ is on any surface; and $\boldsymbol{u}(\boldsymbol{Y}) = 0$ when $\boldsymbol{Y}$ is in the exterior of all surfaces. We use this analytical solution to estimate the error for each of the singular, near-singular and far-field integration schemes in each time-step. We then adjust the order of the quadrature schemes, by incrementing the order by one ($q \leftarrow q+1$) if the error is larger than the required accuracy and decrease the order by one ($q \leftarrow q-1$) if the error is smaller than the required accuracy. While we use the same quadrature order for the near-singular and far-field integration schemes, the order for the singular quadrature scheme can be different.

### *3.3. Collision Handling*

During the course of a numerical simulation, two vesicles may intersect. This happens due to various discretization errors introduced in the simulation. When this happens, it leads to non-physical behavior and the simulation may even break (GMRES may fail to converge).

In every time step, after computing the updated vesicle position, we check if the vesicles intersect. To do this, for each surface mesh point $\boldsymbol{X}$, we find the surfaces near this point and for every such surface $\gamma$, we determine the normal projection $\boldsymbol{X}_0$ of $\boldsymbol{X}$ on $\gamma$. This is already done as part of the near-singular integration discussed in Section 3.2.2 and therefore, we need to do this only once per time step. We compute the dot-product of the outward surface normal vector $\mathbf{n}_{\boldsymbol{X}_0}$ at $\boldsymbol{X}_0$ with the vector $\boldsymbol{X} - \boldsymbol{X}_0$. If the dot-product $\mathbf{n}_{\boldsymbol{X}_0} \cdot (\boldsymbol{X} - \boldsymbol{X}_0) \leq 0$, then we know that the vesicles either touch or intersect. Note that this can sometimes fail since we only check for intersection at the mesh points on the vesicles and

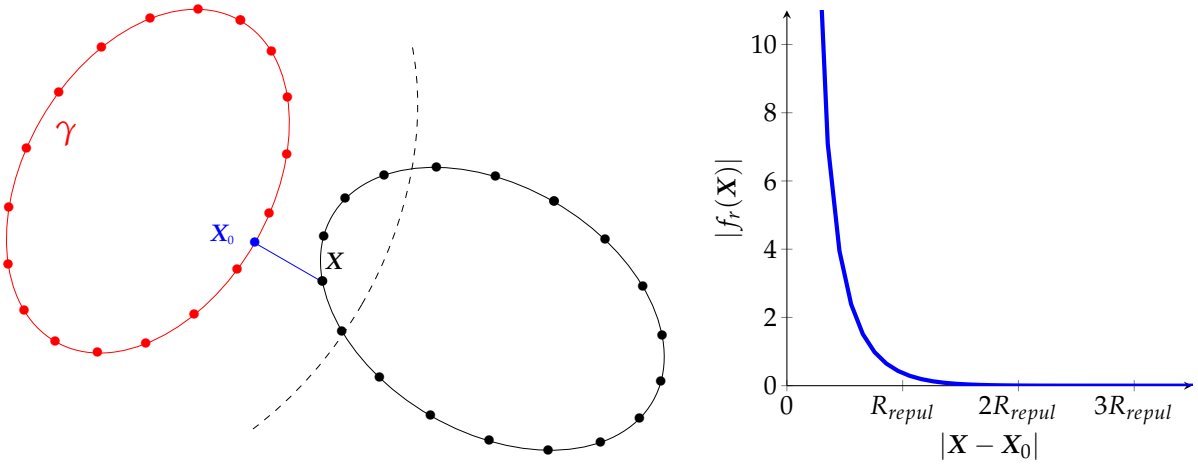



**Figure 3:** *Left: To determine if two vesicles intersect, for each point $\boldsymbol{X}$, we compute its normal projection $\boldsymbol{X}_0$ on the other vesicle. If the vesicles intersect, then $\mathbf{n}_{X_0} \cdot (\boldsymbol{X} - \boldsymbol{X}_0) \leq 0$. We also add a repulsion force given by $f_r(\boldsymbol{X}) = \int_{\boldsymbol{Y} \in \gamma} K(\boldsymbol{X} - \boldsymbol{Y})$. Right: Plot of the repulsion force $f_r(\boldsymbol{X})$ as a function of the distance $|\boldsymbol{X} - \boldsymbol{X}_0|$ for the repulsion function given in Eq. (3.13). The parameter $R_{repul}$ controls the range of the repulsion force. The repulsion becomes infinitely large as the surfaces approach each other and decays rapidly as the distance between them increase. At $|\boldsymbol{X} - \boldsymbol{X}_0| = 3R_{repul}$, the repulsion force $|f_r(\boldsymbol{X})| \sim$ 1E-5.*

also because the quadratic patch is only an approximation of the actual vesicle surface. If we determine that the vesicles intersect, we reject the solution for that time step, reduce the time step by half and recompute the solution.

Despite the above adaptive time-stepping, there are still cases where it is not possible to avoid collision between surfaces. To address this issue, we introduce a repulsion force between the vesicles. The repulsion term is added along with the bending and tensile forces in our Galerkin formulation in Eq. (2.13) as follows

$$\alpha_i \left(\boldsymbol{u}, Y_{nm}\right) = \left(\boldsymbol{u}^{\infty}, Y_{nm}\right) + \sum_{j=1}^{N_\gamma} \left(\mathcal{S}_j[\boldsymbol{f}_b + \boldsymbol{f}_\sigma + f_r], Y_{nm}\right) + \left(\mathcal{D}_j[\boldsymbol{u}], Y_{nm}\right) \tag{3.11}$$

where, the force $f_r$ can be any highly localized repulsive force. We define $f_r(\boldsymbol{X})$ by the convolution of the surface with a kernel function,

$$f_r(\boldsymbol{X}) = \int_{\boldsymbol{Y} \in \gamma} K(\boldsymbol{X} - \boldsymbol{Y}), \qquad \text{where} \quad K(\boldsymbol{X}) = \left(\frac{3{R_{repul}}^4}{2|\boldsymbol{X}|^5} + \frac{{R_{repul}}^2}{|\boldsymbol{X}|^3}\right) \exp\left(\frac{-|\boldsymbol{X}|^2}{{R_{repul}}^2}\right) \boldsymbol{X} \tag{3.12}$$

where, $K$ is the repulsion kernel function and the constant $R_{repul}$ is related to the range of the repulsion force. We choose the repulsion kernel in such a way that the repulsion force becomes infinite as two surfaces approach each other. This allows the surfaces to come arbitrarily close but guarantees that they will not touch. The kernel function also decays quickly as the distance between the surfaces increases to ensure that the repulsion only comes into play when the surfaces are very close together.

Since the evaluation points are very close to the vesicle surface, a Nyström discretization of the integral in Eq. (3.12) will converge very slowly. Instead, we compute this integral analytically by assuming that the vesicle surface is nearly flat in the vicinity of the target point. The repulsion force can then be approximated as,

$$f_r(\boldsymbol{X}) \approx \frac{{R_{repul}}^2}{|\boldsymbol{X} - \boldsymbol{X}_0|^3} \exp\left(\frac{-|\boldsymbol{X} - \boldsymbol{X}_0|^2}{{R_{repul}}^2}\right) (\boldsymbol{X} - \boldsymbol{X}_0) \tag{3.13}$$

When $R_{repul}$ is smaller, the repulsion force is more localized but the equations become stiffer and we need to use smaller time step size. In our simulations we choose $R_{repul} = 2\text{E-}2$.

### 3.4. Area and Volume Correction

The area and volume of a vesicle determine their dynamical behavior through reduced volume [37]. To avoid drift in these values we correct the area $A$ and volume $V$ at each time step by solving the following constraint minimization problem for each vesicle $\gamma$

$$\underset{\text{s.t. } A(\boldsymbol{X})=A_0, V(\boldsymbol{X})=V_0}{\arg\min} \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{X}^\star\|^2_{L^2(\gamma)}, \tag{3.14}$$

where $\boldsymbol{X}^\star$ is the candidate position obtained through time-stepping and reparametrization, whose area and volume may have drifted from $A_0$ and $V_0$. One approach in solving this minimization problem is to linearize the constraints and solve the equality constrained quadratic program directly. The variations of the area and volume of a surface with displacement $\delta\boldsymbol{X}$ are given by [38, Section 9.4]

$$dA(\boldsymbol{X},\delta\boldsymbol{X}) = -2\int_{\gamma(\boldsymbol{X})} H\,\delta\boldsymbol{X}\cdot\boldsymbol{n}\,\mathrm{d}\gamma = -2\,(H,\psi)\,, \tag{3.15}$$

$$dV(\boldsymbol{X},\delta\boldsymbol{X}) = \int_{\gamma(\boldsymbol{X})} \delta\boldsymbol{X}\cdot\boldsymbol{n}\,\mathrm{d}\gamma = (1,\psi)\,. \tag{3.16}$$

In the last terms above, we restricted $\delta\boldsymbol{X}$ to the normal direction $\delta\boldsymbol{X} = \psi\boldsymbol{n}$ because these first variations only depend on the normal perturbation to the surface. The linearized minimization problem is then

$$\underset{\substack{\text{s.t. } dA(\boldsymbol{X},\psi)=\delta A\\ dV(\boldsymbol{X},\psi)=\delta V}}{\arg\min} \frac{1}{2}(\psi,\psi) \tag{3.17}$$

where $\delta A = A_0 - A(\boldsymbol{X})$ and $\delta V = V_0 - V(\boldsymbol{X})$. The Lagrangian for Eq. (3.17) is

$$\mathcal{L}(\psi,\alpha,\beta) = (\psi,\psi) + \alpha\,(dA(\boldsymbol{X},\psi)-\delta A) + \beta\,(dV(\boldsymbol{X},\psi)-\delta V)\,. \tag{3.18}$$

The derivative of $\mathcal{L}$ with respect to the Lagrange multipliers recovers the linearized constraints and the derivative with respect to $\psi$ is

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{X}}(\phi) = (\psi - 2H\alpha + \beta, \phi)\,, \tag{3.19}$$

which gives a linear system for the KKT conditions. The linear system can be simplified to a small system for the Lagrange multipliers:

$$\begin{bmatrix} -4\,(H,H) & 2\,(H,1) \\ 2\,(1,H) & -\,(1,1) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \delta A \\ \delta V \end{bmatrix}. \tag{3.20}$$

After computing $\alpha$ and $\beta$, we evaluate $\psi = 2\alpha H - \beta$ and move the surface to $\boldsymbol{X} + \psi\boldsymbol{n}$. Due to the linearization step we may need to iterate a few times to satisfy the constraint up to the given accuracy. In our experiments, this step typically converges in two or three iterations.

### 3.5. Reparameterization

As a simulation progresses, due to the absence of in-plane shear resistance in vesicles, the quality of the mesh eventually deteriorates. The grid points may cluster in some regions and become sparse in other regions. When unchecked, this leads to unresolvable high frequencies in the spherical harmonic expansion of the vesicle shape and interfacial forces. The distortion of the mesh may also adversely

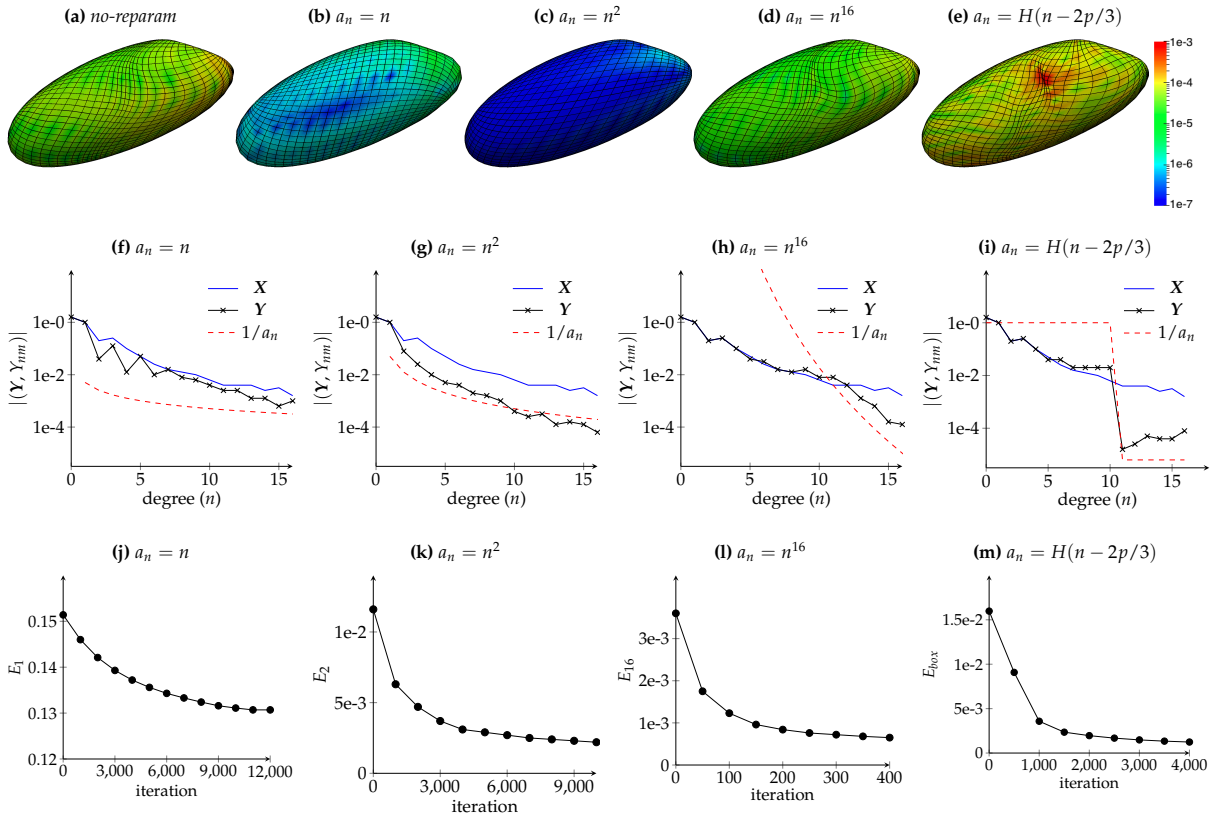



**Figure 4:** *Comparison of different reparameterization schemes. Fig. 4(a) shows the mesh and the singular integration error for a simulation without reparameterization. The reparameterized mesh for different schemes are shown in Figs. 4(b) to 4(e). For each scheme, we also show the spectrum for the original mesh $\boldsymbol{X}$ and the reparameterized mesh $\boldsymbol{Y}$ in Figs. 4(f) to 4(i). The decay of the quality measure $E$ with reparameterization iterations is shown in Figs. 4(j) to 4(m). The reparameterization scheme with the attenuation coefficients $a_n = n^2$ works best as it has small truncation error ($10^{-4}$) for the spherical harmonic expansion and also has about 6-digits of accuracy for double-layer singular integration.*

affect the accuracy of the singular integration scheme. In [6], this is solved by reparameterization of the vesicle surface after each time step. Below, we briefly outline this scheme.

The vesicle surface $\gamma$ parameterized by spherical coordinates is given by the map $\boldsymbol{X}(s) : \mathbb{S}^2 \to \mathbb{R}^3$. Let $F : \mathbb{R}^3 \to \mathbb{R}$ denote the implicit representation of the surface such that $F(\gamma) = 0$ and $\nabla F$ does not vanish. Our goal is the to choose an alternate parameterization $\boldsymbol{Y}(s) : \mathbb{S}^2 \to \mathbb{R}^3$ for the surface such that it minimizes the quality measure $E(\boldsymbol{Y}) = (\boldsymbol{Y}, \boldsymbol{Y})_E$ with the inner product defined as $(\boldsymbol{X}, \boldsymbol{Y})_E := \sum_{n=0}^{p} \sum_{m=-n}^{n} a_n^2 \, (\boldsymbol{X}, Y_{nm}) \, (\boldsymbol{Y}, Y_{nm})$. Therefore, to obtain $\boldsymbol{Y}$ we need to solve the following constrained minimization problem:

$$\underset{\boldsymbol{Y} \in C^\infty(\mathbb{S}^2)}{\arg\min} \; E(\boldsymbol{Y}) \qquad \text{subject to} \qquad F(\boldsymbol{Y}(s)) = 0 \quad \text{for all} \quad s \in \mathbb{S}^2. \tag{3.21}$$

This is reformulated as a pseudo-transient continuation [39] problem and discretized using an explicit first order scheme (see [6] for details) as follows,

$$\boldsymbol{Y}_{k+1} = \boldsymbol{Y}_k - v_k \Delta\tau \qquad \text{where,} \quad v_k = \frac{(I - \mathbf{n}_k \otimes \mathbf{n}_k) \nabla E(\boldsymbol{Y}_k)}{\| (I - \mathbf{n}_k \otimes \mathbf{n}_k) \nabla E(\boldsymbol{Y}_k) \|_\infty} \tag{3.22}$$

We select the reparameterization step size $\Delta\tau$ to be the minimum of $\Delta\tau_{max}$ and $(\boldsymbol{Y}_k, v_k)_E \,/\, (v_k, v_k)_E$. This ensures that the first order reparameterization scheme does not introduce errors larger than $\mathcal{O}(\Delta\tau_{max})$

and having $\Delta\tau \leq \Delta\tau_{max}$ and $(\boldsymbol{Y}_k, \boldsymbol{v}_k)_E \,/\, (\boldsymbol{v}_k, \boldsymbol{v}_k)_E$ guarantees that in each iteration of the algorithm $E(\boldsymbol{Y}_{k+1}) \leq E(\boldsymbol{Y}_k)$, i.e. the quality measure $E(\boldsymbol{Y})$ always decreases. We stop the algorithm when $\Delta\tau$ becomes smaller than a specified tolerance.

In Fig. 4, we analyze the performance of the reparameterization algorithm for different choices of the attenuation coefficients $a_n$ in the definition of the quality measure $E$. Fig. 4(a) shows the mesh with degree $p = 16$ from a simulation without reparameterization. We reparameterize this mesh using different schemes in Figs. 4(b) to 4(e). For each mesh we also visualize the singular integration error computed using the method discussed at the end of Section 3.2.4 for the singular-integration scheme of order $q = 32$. In Figs. 4(f) to 4(i), we plot the spectrum of the spherical harmonic expansion of the reparameterized mesh $\boldsymbol{Y}$ and compare it to that of the original mesh $\boldsymbol{X}$ and the coefficients $1/a_n$. For each reparameterization scheme, we show the decay for the quality measure $E$ with the number of reparameterization iterations in Figs. 4(j) to 4(m). We observe that for scheme $a_n = n$, the spectrum for the reparameterized surface does not decay fast enough and therefore, the surface is not resolved accurately in regions of high curvature. The schemes with $a_n = n^{16}$ and $a_n = H(n - 2p/3)$ affect only the high-order components in the spherical harmonic expansion. In this case, while the surface is resolved accurately, the scheme does not fix the clustering of mesh points and this affects the accuracy for the singular integration scheme. The scheme $a_n = H(n - 2p/3)$ was used in our previous papers. While it worked well for low-order discretizations, our present study shows that it does not work well for high-order discretizations. Finally, the scheme with $a_n = n^2$ has small truncation error for the spherical harmonic expansion and also achieves high accuracy for singular integration. In the remainder of this paper, we always use this reparameterization scheme.

### *3.6. Semi-Implicit Time-Stepping*

At time $t_n$, we denote the membrane position by $\boldsymbol{X}^n$. To compute the updated surface position $\boldsymbol{X}^{n+1}$ at time $t^{n+1}$, we use the semi-implicit time-stepping scheme of [1]. Next, we briefly summarize this scheme.

The interfacial forces $f_b(\boldsymbol{X})$ and $f_\sigma(\boldsymbol{X}, \sigma)$ are defined at each point on the membrane $\gamma$. The bending and tension operators are defined as $\mathcal{S}[f_b(\boldsymbol{X})]$ and $\mathcal{S}[f_\sigma(\boldsymbol{X}, \sigma)]$ respectively. We linearized the bending and tension operators around $\boldsymbol{X}^n$ as follows,

$$B\boldsymbol{u} = \mathcal{S}_{\boldsymbol{X}^n}[f_b(\boldsymbol{X}^n) + f_b{}'(\boldsymbol{X}^n)\boldsymbol{u}\Delta t], \tag{3.23}$$

$$T\sigma = \mathcal{S}_{\boldsymbol{X}^n}[f_\sigma(\boldsymbol{X}^n, \sigma)]. \tag{3.24}$$

The discrete spectral version of these operator are given by: $\widehat{B}\,\widehat{\boldsymbol{u}} = P\,B\,Q\,\widehat{\boldsymbol{u}}$ and $\widehat{T}\,\widehat{\sigma} = P\,T\,Q\,\widehat{\sigma}$. Similarly, we also define the following discrete spectral operators: $\widehat{D}\,\widehat{\boldsymbol{u}} = P\,\mathcal{D}_{\boldsymbol{X}^n}[Q\,\widehat{\boldsymbol{u}}]$ and $\widehat{\mathrm{div}}_\gamma\,(\widehat{\boldsymbol{u}}) = P\,\mathrm{div}_\gamma\,(Q\,\widehat{\boldsymbol{u}})$. The operators $B$, $T$ and $D$ are implemented using the quadrature scheme described in Section 3.2. The Galerkin formulation in Eqs. (2.13–2.15) is now discretized to give the globally semi-implicit time-stepping scheme,

$$\alpha_i\widehat{\boldsymbol{u}}_i = \widehat{\boldsymbol{u}}_i^{\infty} + \sum_{j=1}^{N_\gamma}\left(\widehat{B}_{ij}\widehat{\boldsymbol{u}}_j + \widehat{T}_{ij}\widehat{\sigma}_j + \widehat{D}_{ij}\widehat{\boldsymbol{u}}_j\right) \quad \text{for all } i = 1, \ldots, N_\gamma, \tag{3.25}$$

$$\widehat{\mathrm{div}}_\gamma\,\widehat{\boldsymbol{u}} = 0, \tag{3.26}$$

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \boldsymbol{u}\Delta t. \tag{3.27}$$

The subscripts $i$ and $j$ in operators $\widehat{B}_{ij}$, $\widehat{T}_{ij}$ and $\widehat{D}_{ij}$ denote that the operators are applied to the $j^{\text{th}}$ surface and the target points are on the $i^{\text{th}}$ surface. We use GMRES to solve this linear system for the tension $\sigma$, velocity $\boldsymbol{u}$ and the updated position $\boldsymbol{X}^{n+1}$.

### 3.7. Adaptive Time-Stepping

Our adaptive time-stepping scheme is based on the work of [9, 10]. Even though we have a first order time-stepping scheme, we present the algorithm for choosing the time step size for a general $k^{\text{th}}$-order scheme. In each step of the simulation, we use an estimate of the error $e_n$ incurred in the current iteration with time step size $\Delta t_n$ to determine the optimal time step size $\Delta t_{n+1}$ for the next step. The error $e_n$ is estimated in one of the following two ways:

- We determine the new position $\hat{\boldsymbol{X}}_n$ and $\boldsymbol{X}_n$ using different numerical schemes. From the vesicle position $\boldsymbol{X}_{n-1}$ at time $t_{n-1}$, we compute $\hat{\boldsymbol{X}}_n$ by taking one time step of size $\Delta t_n$ and compute $\boldsymbol{X}_n$ by taking two time steps of size $\Delta t_n/2$. Then, we defined the error estimate as $e_n := \|\boldsymbol{X}_n - \hat{\boldsymbol{X}}_n\|_2$.
- We measure the change in invariant quantities such as the surface area $A_n$ and volume $V_n$. We define the error estimate as $e_n := \max(|\Delta A|/A, |\Delta V|/V)$.

In the first case, we have to perform extra computation to determine two numerical solutions for the position and this can be expensive. In the second case, it requires very little computation to determine the change in area and volume between time steps. While the second error estimate worked well in 2D, it is not robust enough 3D and we observed several instances where the method underestimated the error. Therefore, in this work we always use the first error estimate. In both cases, the total error also depends on other factors such as truncation errors, accuracy of the Stokes operator (FMM and quadratures) and the GMRES tolerance used to enforce inextensibility. For a $k^{\text{th}}$-order time-stepping scheme, the error is observed to scale with $\Delta t_n$ as $e_n = \mathcal{O}\left(\Delta t_n^{k+1} + \epsilon_{other}\,\Delta t_n\right)$. Therefore, have to ensure that $\epsilon_{other}$ is also small enough.

For a long time scale simulation with time-horizon $T$ and an overall error tolerance $\mathcal{E}$, we present an algorithm to determine the optimal step size $\Delta t_n$ at each time step. In $(n+1)^{\text{th}}$ time step, we choose the step-size $\Delta t_{n+1}$ such that the error $e_{n+1}$ is as close as possible to the maximum allowed error but does not exceed this error. Therefore, we want $e_{n+1}/\Delta t_{n+1} = \beta\mathcal{E}/T$, where $\beta < 1$ is a safety factor. In our experiments, we choose $\beta = 0.9$.

We assume that for a $k^{\text{th}}$-order time-stepping scheme, the error scales with $\Delta t_n$ as $e_n = \mathcal{O}\left(\Delta t_n^{k+1}\right)$. We also assume that the constants in this order estimate do not change significantly between consecutive time steps. Then, we have, $\Delta t_{n+1}^{k+1}/e_{n+1} = \Delta t_n^{k+1}/e_n$. We substitute $e_{n+1}/\Delta t_{n+1} = \beta\mathcal{E}/T$ to obtain the new time step size $\Delta t_{n+1}$,

$$\Delta t_{n+1} = \Delta t_n \left(\beta\,\frac{\mathcal{E}}{T}\,\frac{\Delta t_n}{e_n}\right)^{1/k}$$

Then, we compute the solution $\boldsymbol{X}_{n+1}$ at time $t_{n+1} = t_n + \Delta t_{n+1}$. We measure the error $e_{n+1}$ and check if it satisfies the condition $e_{n+1}/\Delta t_{n+1} \leq \mathcal{E}/T$. If the condition is satisfied, we accept the solution and proceed to the next time step. If the condition is not satisfied, we reject the solution and update the time step size as follows,

$$\Delta t_{n+1} = \Delta t_{n+1} \left(\beta\,\frac{\mathcal{E}}{T}\,\frac{\Delta t_{n+1}}{e_{n+1}}\right)^{1/k}$$

We repeat the above process with this updated time step size.

### 3.8. Algorithm Summary and Computational Cost

For a simulation of $N_\gamma$ vesicles, the initial conditions are defined by the position and shape of the vesicles using $p^{\text{th}}$-order spherical harmonic discretization $\widehat{\boldsymbol{X}}$. For each surface, we provide the bending modulus, the excess density and the viscosity contrast of the fluid inside the vesicle compared to the fluid outside. We also specify the boundary conditions (periodic with period length or free space),

the background velocity field $\boldsymbol{u}^{\infty}$ and the time-horizon for the simulation $T$. In addition, we provide the following simulation parameters: the error tolerance for time-adaptivity $\mathcal{E}$, the initial time step size $\Delta t$, the tolerance for each GMRES solve $\epsilon_{\text{GMRES}}$, the maximum reparameterization step size $\Delta\tau_{max}$, the distance parameter for repulsion $R_{repul}$, the initial quadrature $q$ order and the required accuracy for singular, near-singular and far-field integration. The discretization order $p$ and the tolerance for time-adaptivity $\mathcal{E}$ are decided based on the accuracy requirements for the simulation. We choose the GMRES tolerance $\epsilon_{\text{GMRES}}$ and the quadrature accuracy to be about 1E-6 or smaller. The choice of the initial time step size $\Delta t$ and the initial quadrature order $q$ has little effect since they are determined adaptively. The reparameterization step size $\Delta\tau_{max}$ is set to roughly match the accuracy of the surface discretization ($\sim$ 1E-3 in our experiments). The repulsion distance $R_{repul}$ must be large enough so that the repulsion force can be resolved using $p^{\text{th}}$-order spherical harmonics. The results in Fig. 5(f) can serve as a guideline for selecting $R_{repul}$ for different values of $p$.

To compute the new surface position $\widehat{\boldsymbol{X}}^{+}$ after time $\Delta t$, we solve the linear system discussed in Section 3.6. To do this, we setup the RHS for the linear system and initialize the linear operator. Constructing the linear operator requires setting up the operators for the linearized interfacial forces and the Stokes single and double-layer potentials. Detailed discussion of linearized interfacial forces can be found in [1, 6]. For the Stokes layer potentials, we perform the setup for each of the singular, near-singular and far-field integration algorithms discussed in Section 3.2. This involves: computing the singular integration matrices; identifying the pairs of vesicles and their near target points, computing projection of these target points on the vesicle surface; and constructing the octree for the FMM. During the setup for near-singular integration, we also check for vesicle collision, compute the repulsion force and add it to the RHS. We also check the accuracy of the quadratures using the scheme described in Section 3.2.4 and update the order of the quadratures. Then, we solve this linear system using GMRES to obtain the new position $\widehat{\boldsymbol{X}}^{+}$ and the surface tension $\widehat{\sigma}^{+}$. Each GMRES iteration involves computing the linearized interfacial forces and then evaluating the Stokes layer potential.

For adaptive time-stepping, we perform three GMRES solves as discussed in Section 3.7 to estimate the solution error. If the error is smaller than $\mathcal{E}\Delta t/T$, then we accept the new solution and advance the time by $\Delta t$; otherwise, we reject the solution. Then, we update the time step size to be used in the next iteration. After each successful update, we apply the area and volume correction algorithm discussed in Section 3.4 and reparameterize the surface discretization as described in Section 3.5. We repeat the above steps until $t = T$.

A summary of the computational cost associated with the different stages of our algorithm is given in Table 2.

## 4. Results

We present some numerical results to show the accuracy and time-to-solution on a single node in Section 4.1 and strong and weak scalability of our method in Sections 4.2 and 4.3 respectively. All results are presented for the Stampede system at the Texas Advanced Computing Center (TACC). It is a Linux cluster consisting of 6,400 compute nodes connected by 56Gb/s FDR Mellanox InfiniBand network in a fat tree configuration. Each compute node has dual eight-core Intel Xeon E5-2680 CPUs running at 2.7GHz and 32GB of memory. In addition, most nodes have an Intel Xeon Phi SE10P co-processor, while a few have an NVIDIA K20 GPU co-processor; however, our current implementation can not utilize these accelerators.

### *4.1. Single Node Results*

We present results for two vesicles in shear flow as shown in Figs. 5(a) to 5(d). The fluid inside and outside the vesicles is identical, the vesicles have a reduced volume of 0.85 and bending modulus of 0.01 and the simulation has a time-horizon of $T = 160$. The experiment is designed to underline the significance of high-order spatial discretization, time-adaptivity, near-singular integration, reparame-

| | | | | |
|---|---|---|---|---|
| Setup | $T_{setup} =$ | $N_{\text{Tstep}} \times \frac{N_\gamma}{n_p} p^3 (p^2 + q^2)$ | + | (singular) |
| | | $N_{\text{Tstep}} \times \frac{N_\gamma}{n_p} q^2 \left( \log \frac{N_\gamma}{n_p} q^2 + \log n_p \right)$ | + | (near-singular) |
| | | $N_{\text{Tstep}} \times \frac{N_\gamma}{n_p} q^2 \left( \log \frac{N_\gamma}{n_p} q^2 + \log n_p \right)$ | | (far-field) |
| Singular Integration | $T_{self} =$ | $N_{\text{Tstep}} \times N_{\text{iter}} \times \frac{N_\gamma}{n_p} p^4$ | | |
| Near-singular Integration | $T_{\text{near}} =$ | $N_{\text{Tstep}} \times N_{\text{iter}} \times \frac{N_\gamma}{n_p} p^2 q^2$ | | |
| Far-field Integration | $T_{\text{near}} =$ | $N_{\text{Tstep}} \times N_{\text{iter}} \times \frac{N_\gamma}{n_p} (p^2 + q^2)$ | + | (computation) |
| | | $N_{\text{Tstep}} \times N_{\text{iter}} \times \left( \frac{N_\gamma}{n_p} q^2 \right)^{2/3} \log n_p$ | | (communication) |
| Reparameterization | $T_{repar} =$ | $N_{\text{Tstep}} \times N_{repar} \times \frac{N_\gamma}{n_p} p^3$ | | |

**Table 2:** *Time complexity for $N_\gamma$ vesicles with $p^{\text{th}}$-order surface discretization, $q^{\text{th}}$-order quadratures (such that $p < q$), $N_{\text{Tstep}}$ GMRES solves of the implicit time-stepping scheme with $N_{\text{iter}}$ GMRES iterations per solve on $n_p$ processors. We denote the average number of reparameterization iterations in each time step by $N_{repar}$.*

terization and collision handling. The simulation reveals significant inter-vesicle interactions and the vesicles undergo large deformations. The vesicles are closest at $t = 136$ (see Fig. 5(c)). Without repulsion, we require a discretization order of at least $p = 32$ to accurately resolve this flow. For lower orders, the spatial discretization errors cause the vesicles to intersect and this causes the simulation to break.

*4.1.1. Convergence Analysis.* We study dependence of the solution error on the discretization order ($p$), the tolerance for time-adaptivity ($\mathcal{E}$) and the repulsion distance ($R_{repul}$). All solutions are computed using the first order implicit time-stepping scheme and using the block-diagonal preconditioner. We use a fixed GMRES tolerance of $\epsilon_{\text{GMRES}} = 1\text{E-}7$ for the implicit solver. All boundary integrals are computed using $2\times$ upsampling of the mesh; i.e. the quadrature order for singular, near-singular and far-field integration is $q = 2p$ for $p^{\text{th}}$-order surface discretization. We reparameterize using the attenuation coefficients $a_n = n^2$ with reparameterization time step size $\Delta\tau_{\max} = 1\text{E-}4$ and reparameterization termination condition $\Delta\tau < 1\text{E-}5$.

*Reference Solution.* We construct a reference solution without repulsion ($R_{repul} = 0$), using adaptive time-stepping with $\mathcal{E} = 0.1$ and with discretization order $p = 32$. Computing the reference solution required about two days of compute time on a single node of Stampede. Attaining similar accuracy without time-adaptivity would be $5\times$ more expensive.

*Time-Adaptivity and Relation Between $\mathcal{E}$ and $\Delta t$.* In Fig. 5(e), for fixed $p = 32$ and no repulsion, we plot the time step size $\Delta t$ during the second half of the simulation ($80 \leq t \leq 160$). We vary the tolerance for time-adaptivity ($\mathcal{E}$) and observe an approximately linear relationship between $\Delta t$ and $\mathcal{E}$ due to the first order semi-implicit time-stepping scheme.

*Effect of Repulsion Distance $R_{repul}$.* In Fig. 5(f), we introduce repulsion between the surfaces and plot the error in the vesicle position (compared with the reference solution) at the end of the simulation as a

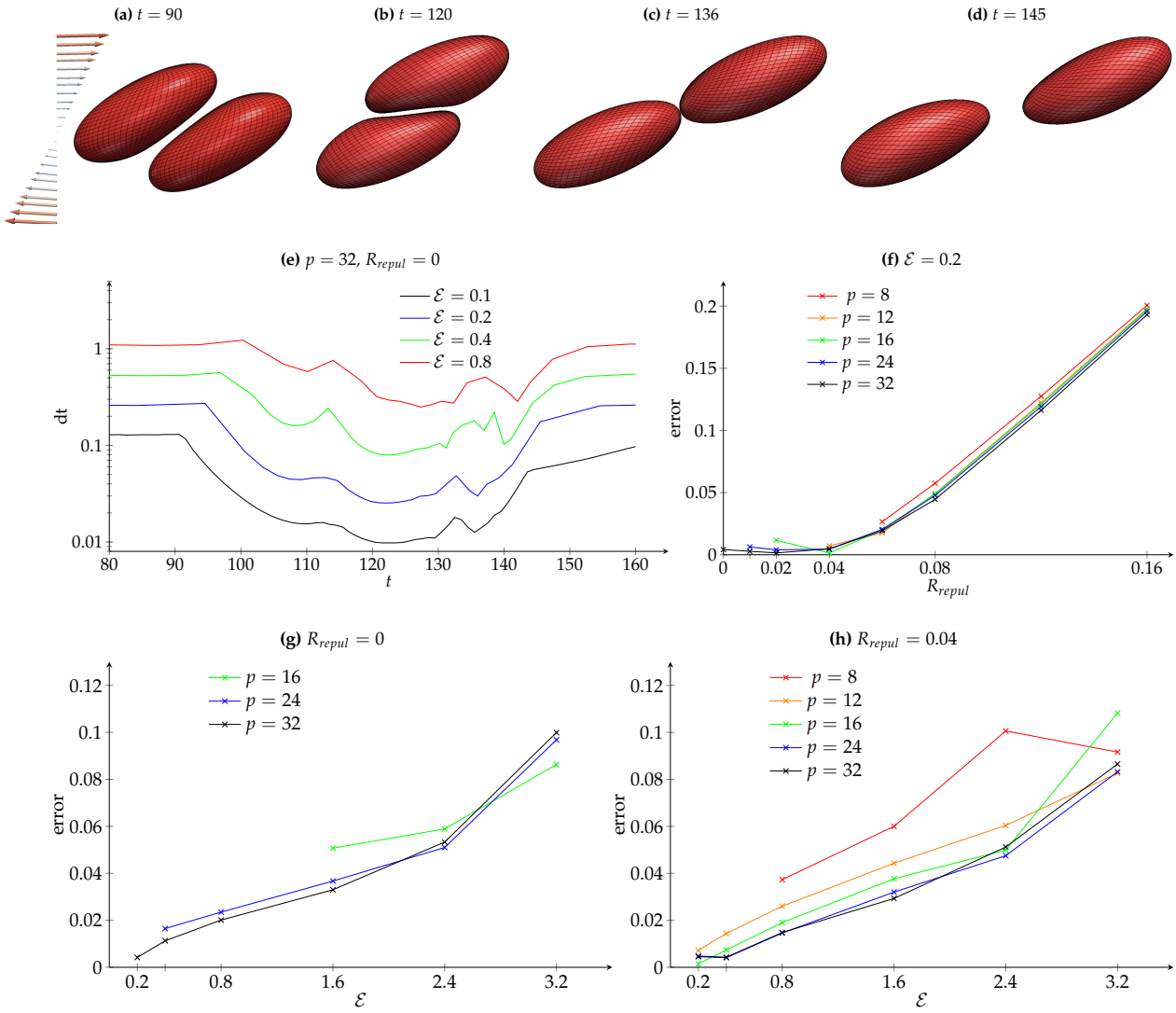



**Figure 5:** *Convergence results for two vesicles in shear flow. The reference solution shown in figures Figs. 5(a) to 5(d) is computed using discretization order $p = 32$, time-adaptivity error tolerance $\mathcal{E} = 0.1$ and no repulsion force. In Fig. 5(e), we show the time step size $\Delta t$ at different points during the simulation for the reference solution ($\mathcal{E} = 0.1$) and also larger values of $\mathcal{E}$ while keeping $p = 32$ fixed. The first order behavior of the time-stepping scheme can be observed. We also observe a smaller time step size in regions where vesicle-vesicle interactions become significant. Fig. 5(f) shows the dependence of error in vesicle position (compared to the reference solution) when a short range repulsion force in added between the vesicles. The repulsion force allows us to use lower order discretizations and does not introduce significant errors for $R_{repul} \leq 0.04$. For different discretization orders, we show the linear dependence of the vesicle position error with the tolerance for time-adaptivity $\mathcal{E}$ in Figs. 5(g) and 5(h) for the cases without repulsion and with repulsion ($R_{repul} = 0.04$) respectively.*

function of the repulsion distance ($R_{repul}$) for different discretization orders and time-adaptivity tolerance $\mathcal{E} = 0.2$. Due to reparameterization, we can not directly compare the position of the surface grid points; therefore, we compare the surface center of mass $c_i = \int_{\gamma_i} \mathrm{d}\gamma_i$ for the $i^{\text{th}}$ vesicle. We report the maximum error in the position for any vesicle at the end of the simulation and this error is normalized by the vesicle length. We observe that for $p = 32$, adding repulsion does not cause any noticeable increase in error for $R_{repul} < 0.04$. This is because the range of the repulsive force is smaller than the separation between the vesicles. We observe a steady increases in error with $R_{repul}$ for $R_{repul} \geq 0.04$. By adding repulsion, we are also able to compute solutions with lower order spatial discretizations. In general, the repulsion distance should be at least of the order of the distance between the grid points so that the repulsion force can be resolved on the vesicle surface. For $R_{repul} = 0.04$, we can compute solutions with $p \geq 12$ and this does not appear to significantly affect the solution accuracy.

*Relation Between $\mathcal{E}$ and Position Error.* In Figs. 5(g) and 5(h), we plot the error in vesicle position as a function of the time-adaptivity tolerance $\mathcal{E}$ for $R_{repul} = 0$ and $R_{repul} = 0.04$ respectively. In both cases we observe an approximately linear relationship between the position error and $\mathcal{E}$.

| $p$ | $\mathcal{E}$ | $R_{repul}$ | *error* | $N_{\text{Tstep}}$ | $N_{\text{iter}}$ | $T_{solve}$ | $T_{setup}$ | $T_{self}$ | $T_{near}$ | $T_{far}$ | $T_{repar}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 0.2 | 0.0 | 4.1E-3 | 3270 | 34 | 64321 | 23592 | 2724 | 6684 | 22925 | 4208 |
| 32 | 0.4 | 0.0 | 1.1E-2 | 1116 | 40 | 25864 | 8184 | 1104 | 2417 | 9228 | 3285 |
| 32 | 0.8 | 0.0 | 2.0E-2 | 441 | 50 | 13087 | 3316 | 558 | 1007 | 4500 | 2907 |
| 32 | 1.6 | 0.0 | 3.3E-2 | 216 | 67 | 8902 | 1670 | 363 | 572 | 2966 | 2802 |
| 32 | 2.4 | 0.0 | 5.4E-2 | 153 | 81 | 8828 | 1219 | 310 | 471 | 2522 | 3844 |
| 32 | 0.2 | 0.0 | 4.1E-3 | 3270 | 34 | 64321 | 23592 | 2724 | 6684 | 22925 | 4208 |
| 24 | 0.4 | 1E-2 | 1.3E-2 | 1191 | 39 | 8474 | 1933 | 262 | 954 | 3283 | 1157 |
| 16 | 0.8 | 2E-2 | 2.8E-2 | 513 | 36 | 1091 | 127 | 29 | 97 | 358 | 263 |
| 12 | 1.6 | 4E-2 | 4.3E-2 | 228 | 45 | 333 | 21 | 8 | 21 | 100 | 90 |
| 8 | 2.4 | 6E-2 | 4.3E-2 | 174 | 27 | 106 | 5 | 2 | 4 | 30 | 38 |

**Table 3:** *We present convergence results for the shear flow problem visualized in Figs. 5(a) to 5(d). For different values of error tolerance $\mathcal{E}$, we report the error in vesicle position compared to a reference solution computed with $\mathcal{E} = 0.1$, $p = 32$ and no repulsion. We also report the number of solves of the implicit time-stepping scheme $N_{\text{Tstep}}$, the average number of GMRES iterations needed for each solve $N_{\text{iter}}$ and the time-to-solution $T_{solve}$. In addition, we report a breakdown of the time spent in different stages of the algorithm: the cost for reparameterization $T_{repar}$, the cost for singular $T_{self}$, near-singular $T_{near}$ and far-field $T_{far}$ interactions and their setup cost $T_{setup}$. We compare results for high-order discretization and no repulsion with results for low order discretizations, which are made possible due to the addition of a repulsion term. For about 5% error in vesicle position, using low order discretization ($p = 8$) is over 80× faster when compared with a solution computed using $p = 32$.*

*4.1.2. Timing Results.* In Table 3, we present detailed results for the above shear flow test case. In addition to the error in vesicle position, we also report the number of solves of the implicit time-stepping scheme $N_{\text{Tstep}}$, the average number of GMRES iterations $N_{\text{iter}}$, the overall solve time $T_{solve}$ and a detailed breakdown of the time spent in different stages of the algorithm. We report two sets of results.

*Fixed Spatial Discretization Order $p = 32$.* In the first set, we report results for a fixed discretization order $p = 32$, no repulsion and varying tolerance values for time-adaptivity $\mathcal{E}$. Due to the first order time-stepping scheme, we observe an approximately linear relationship between the number of time steps $N_{\text{Tstep}}$ and inverse of the error tolerance $1/\mathcal{E}$. At the same time, we observe that with increasing time step size (due to increasing $\mathcal{E}$), the number of GMRES iterations per time step increase significantly. This happens because the semi-implicit scheme becomes more ill-conditioned. Therefore, we do not

observe the expected speedup in $T_{solve}$ with increasing $\mathcal{E}$. The interactions between vesicles computed through single-layer and double-layer Stokes kernel functions are evaluated at every GMRES iteration while the setup phase is execute once for each time step. Therefore, for a fixed discretization order $p$, we observe that $T_{setup}$ scales as $\mathcal{O}\left(N_{\text{Tstep}}\right)$ while $T_{self}$, $T_{near}$ and $T_{far}$ scale as $\mathcal{O}\left(N_{\text{Tstep}} N_{\text{iter}}\right)$. We also note that the reparameterization time $T_{repar}$ shows little variation with $\mathcal{E}$. This is because for larger $\mathcal{E}$ even though we reparameterize fewer times (since $N_{\text{Tstep}}$ is smaller), a larger time step size means that we require many more reparameterization iterations.

*Optimal Spatial Discretization Order.* In the second set of results in Table 3, we add a short range repulsion force between the surfaces to prevent collision between them. This allows us to use lower order discretizations and obtain a faster time to solution. Comparing these results with high-order discretizations, we observe that the error in vesicle position and number of time steps $N_{\text{Tstep}}$ remain the same for the same value of $\mathcal{E}$. However, the number of GMRES iterations are much smaller for low order discretizations. We believe this is due to the smaller size of the linear system being solved. In addition, each GMRES iteration has drastically lower computational cost for smaller discretization orders $p$ since we have $\mathcal{O}\left(N_\gamma p^4\right)$ cost for singular and near-singular integration and $\mathcal{O}\left(N_\gamma p^2\right)$ cost for far-field interactions. The setup stage (with $\mathcal{O}\left(N_\gamma p^5\right)$ cost for computing the singular integration operator) and the reparameterization algorithm (with $\mathcal{O}\left(N_\gamma p^3\right)$ cost per iteration) are also significantly less expensive for smaller $p$. For the same solution accuracy, we observe nearly two orders of magnitude speedup when using low order discretizations with repulsion compared to high-order discretizations without repulsion.

### *4.2. Strong Scaling*

In this section, we present strong scaling results for the periodic Taylor-vortex flow simulation shown in Fig. 6. We used the following background velocity field

$$\boldsymbol{u}^{\infty}(x,y,z) = \alpha \sin\left(\tfrac{2\pi x}{L}\right)\cos\left(\tfrac{2\pi y}{L}\right)\sin\left(\tfrac{2\pi z}{L}\right)\mathbf{e_x} + \alpha\cos\left(\tfrac{2\pi x}{L}\right)\sin\left(\tfrac{2\pi y}{L}\right)\sin\left(\tfrac{2\pi z}{L}\right)\mathbf{e_y} \quad (4.1)$$

where, $L = 17$ is the period length of the domain and $\alpha = 0.1$ is a scaling factor. The domain has 1408 biconcave shaped vesicles with 35% volume fraction. Each vesicle has an approximate diameter of 1.89, a height of 0.54, a reduced volume of 0.65 and a bending modulus of 0.1. For this simulation, we used 16th-order spherical harmonic discretization with 50th-order quadratures for singular integration and 24th-order quadratures for near-singular and far-field integration for about 5-digits of accuracy. We used our adaptive time-stepping scheme with an error tolerance of $\mathcal{E} = 0.02$ for time-horizon $T = 2$. The linear system for the semi-implicit time-stepping scheme was solved using GMRES with a relative tolerance of 1E-5. For time-horizon $T = 2$, we needed 18 GMRES solves and each solve required an average of $N_{\text{iter}} = 139$ iterations for discretization order $p = 16$ and $N_{\text{iter}} = 311$ iterations for $p = 32$.

In Fig. 7, we report the total CPU time (wall-time$\times$CPU cores). For $p = 16$ (figure on the left), we scale from 16 CPU cores (1 compute node) to 1024 cores (64 compute nodes). For $p = 32$ (figure on the right), $4\times$ more memory is required for storing the singular integration operators and therefore we start from 256 CPU cores (16 compute node) and scale up to 2048 cores (128 compute nodes). For $p = 16$, we achieve a $26.5\times$ speedup in the total wall-time or 41.4% strong scaling efficiency and for $p = 32$, we achieve a $3.8\times$ speedup in the total wall-time or 47.9% strong scaling efficiency. Overall, we observe that the case with $p = 32$ is about $10\times$ more expensive that $p = 16$.

We also provide a breakdown of the time spent in the different stages of our algorithm. The cost of the setup stage is dominated by the computation of singular integration matrix for each surface. Since the setup is performed only once for every $N_{\text{iter}}$ evaluations, the setup cost is dwarfed by the evaluation cost (singular, near-singular and far-field integration) and requires just 3% $\sim$ 6% of the runtime. The singular integration requires very little work (about 1% for $p = 16$ and 3% for $p = 32$ of

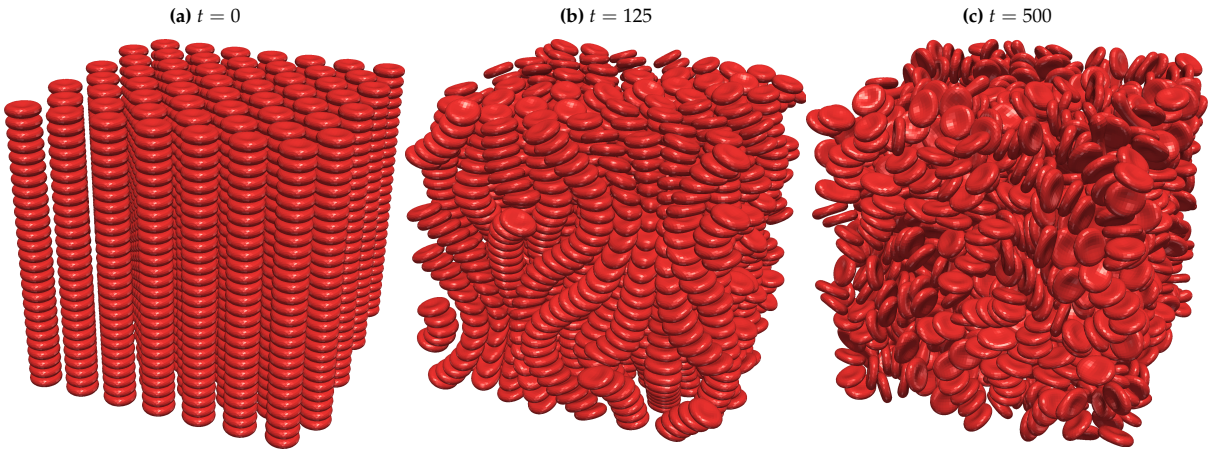



**Figure 6:** *A simulation of* 1408 *vesicles in a periodic Taylor-vortex flow. The vesicles have a volume fraction of* 35% *and each vesicle has a biconcave shape with a reduced volume of* 0.65*. For this simulation, we used 16th-order discretization with 50th-order quadratures for singular integration and 24th-order quadratures for near-singular and far-field integration. We used adaptive time-stepping with error-factor* $\mathcal{E}/T = 0.01$ *and with a tolerance of* 1E-5 *for the GMRES solve.*

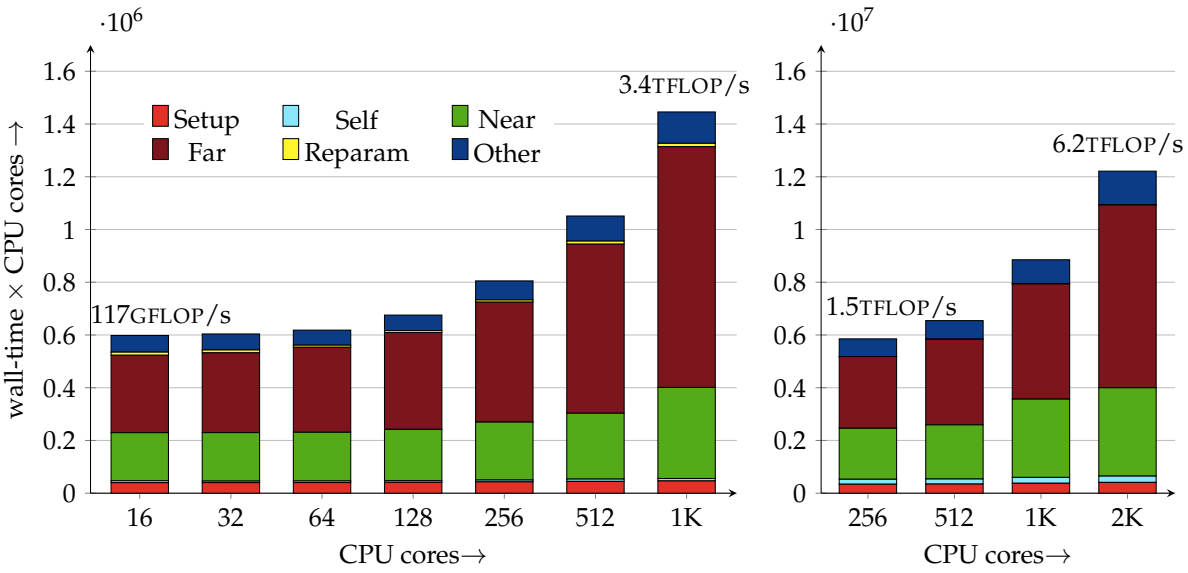



**Figure 7:** *Strong scalability results for the periodic Taylor-vortex flow in Fig. 6. We present results for discretization order* $p = 16$ *on the left and* $p = 32$ *on the right. In both cases, we used 50th-order quadratures for singular integration and 24th-order quadratures for near-singular and far-field integration. We solved the problem for a time-horizon* $T = 2$ *and the adaptive time-stepping scheme required* 18 *GMRES solves. On average, each GMRES solve requires* $N_{\text{iter}} = 138$ *iterations for* $p = 16$ *and* $N_{\text{iter}} = 311$ *iterations for* $p = 32$*.*

runtime) due to our $\mathcal{O}\left(p^4\right)$ scheme. Due to the dense packing of the vesicles, we need to compute near-singular integration for a large number of target points. Therefore, near-singular integration is relatively expensive and requires about 23% $\sim$ 33% of the CPU time; however, it scales well since it is compute bound and requires very little communication. The far-field computation is the most expensive stage in our scheme and requires 46% $\sim$ 63% of the total CPU time. It is implemented using FMM and gives good performance up to 256 cores for $p = 16$ and up to 1024 cores for $p = 32$. As we increase the

number of CPU cores further, the problem size per core is too small to remain efficient and we begin to lose performance. The reparameterization does not require any communication and is inexpensive compared to the overall solve time. The remaining time (about 8% $\sim$ 11%) is mostly spent inside the GMRES solve in PETSc.

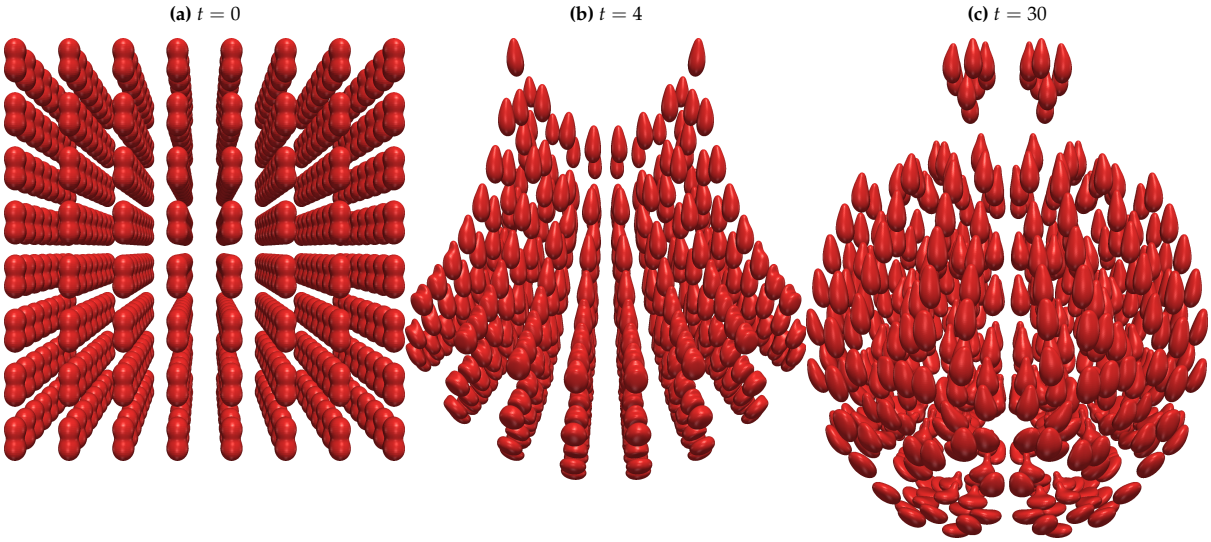



**Figure 8:** *A simulation showing sedimentation of vesicles under gravitational force. We start with* 512 *vesicles arranged in an* $8 \times 8 \times 8$ *lattice at where, each vesicle has a reduced volume of* 0.85*, bending modulus of* 0.05 *and has excess density* $\rho_i - \rho = 1.0$ *(and gravitational acceleration* $g = 1.0$*). We used our adaptive time-stepping scheme with error factor* $\mathcal{E}/T = 0.02$ *and a spatial discretization order* $p = 16$*.*

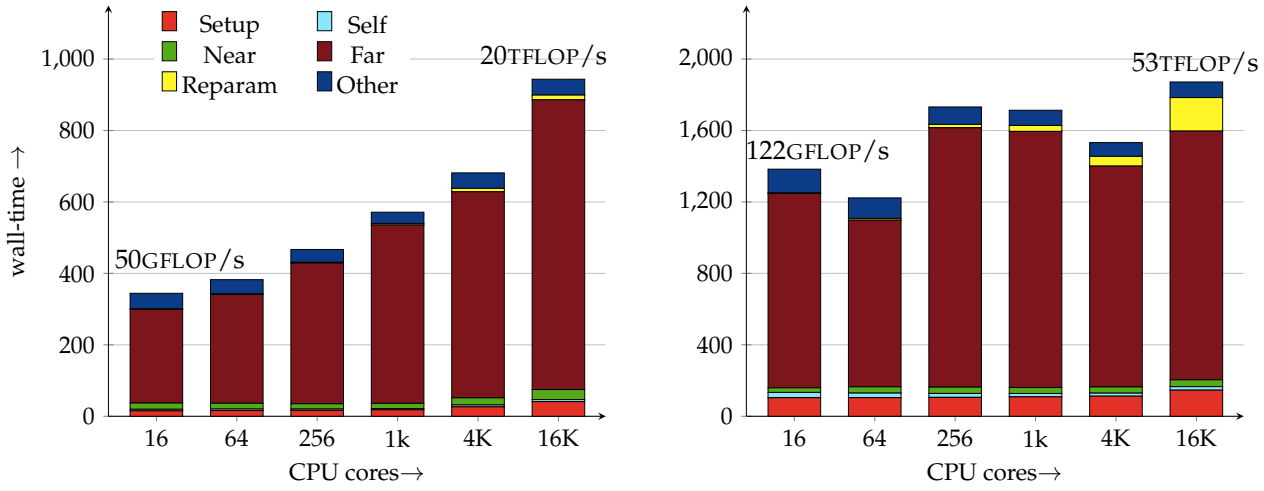
### *4.3. Weak Scaling*

In Fig. 9, we present weak scaling results for a polydisperse sedimentation flow on 16$K$ CPU cores. We used 16th-order discretization, a fixed step-size of $\Delta t = 0.01$ and a time-horizon $T = 0.2$. For the semi-implicit scheme, we used a GMRES tolerance of 1E-5 and the average number of GMRES iterations varied from 64 iterations for the smallest problem size to 25 iterations for the largest problem size. We present two sets of results for different grain sizes. In the first case (Fig. 9:left), we have one vesicle per CPU core. We achieve 50GFLOP/s on 16 cores and 20TFLOP/s on 16$K$ cores; i.e. 400$\times$ increase in performance for 1024$\times$ increase in the number of processors. In the second case (Fig. 9:right), we have seven vesicle per CPU core. We achieve 122GFLOP/s on 16 cores and 53TFLOP/s on 16$K$ cores; i.e. 437$\times$ increase performance for 1024$\times$ increase in the number of processors.

We have presented a detailed breakdown of the time in different stages of the algorithm as we scale from 16 cores to 16$K$ cores. The solve time is dominated by the far-field integration stage, which accounts for 74% $\sim$ 86% of the total time. Unlike the Taylor-Green vortex flow discussed in Section 4.2, for sedimentation flow, the near-singular integration requires very little work (about 2% $\sim$ 5% of $T_{solve}$). This is due to the relatively lower density of vesicles resulting in fewer near-singular interactions between vesicles. The setup and self interaction stages requires about 4% $\sim$ 8% and 1% $\sim$ 2% of the solve time respectively. The reparameterization time $T_{repar}$ ranges from 0.3% $\sim$ 10% of the total time.

## 5. Conclusions

We have presented new algorithms for fast simulation of vesicle in a Stokesian fluid. These include efficient singular, near-singular and far-field integration algorithms for computing single- and double-layer Stokes potentials. We have developed algorithms for detecting collision between vesicles and to




| cores | $N_\gamma$ | $N_{\text{iter}}$ | $T_{solve}$ | GFLOP/s |
|---|---|---|---|---|
| 16 | 16 | 64 | 344 | 50 |
| 64 | 64 | 59 | 382 | 204 |
| 256 | 256 | 52 | 466 | 766 |
| 1024 | 1024 | 44 | 571 | 2675 |
| 4096 | 4096 | 37 | 681 | 6246 |
| 16384 | 16384 | 31 | 943 | 20173 |

| cores | $N_\gamma$ | $N_{\text{iter}}$ | $T_{solve}$ | GFLOP/s |
|---|---|---|---|---|
| 16 | 112 | 57 | 1384 | 122 |
| 64 | 448 | 48 | 1223 | 404 |
| 256 | 1792 | 41 | 1732 | 1350 |
| 1024 | 7168 | 34 | 1713 | 4832 |
| 4096 | 28672 | 30 | 1532 | 15206 |
| 16384 | 114688 | 25 | 1872 | 53350 |

**Figure 9:** *Weak scalability results for polydisperse sedimentation similar to the flow visualized in Fig. 8 on 16K CPU cores. We compute solutions for a time-horizon $T = 0.2$ using a fixed time step size $\Delta t = 0.01$. We used discretization order $p = 16$ and singular, near-singular and far-field quadratures of order $q = 28$. The vesicles have a reduced volume of 0.85, bending modulus in the range $[0.05, 0.1]$, viscosity contrast in the range $[0.5, 5]$ and an excess density of 1. We show results for two different problem sizes: on the left for 1 vesicle per core and on the right for 7 vesicles per core. For each case, we present a bar graph showing the breakdown of the solve time into each of the different stages of the algorithm. We also report the average number of GMRES iteration $N_{\text{iter}}$ and the overall performance in GFLOP/s for each test case.*

handle such collisions in a robust way by adding a repulsion force between the surfaces. We have also developed an algorithm for correcting the area and volume of vesicles in long-time scale simulations. We have analyzed our surface reparameterization algorithm to determine the optimal choice of the quality measure function. We have implemented an adaptive time-stepping scheme, which allows us to choose the optimal time step size. This has significantly improved time-to-solution while still achieving the desired error tolerance. These algorithms have enabled us to compute accurate, long-time scale simulations for flows with high volume fraction of vesicles.

We have presented convergence studies to show first order convergence in the time step size. We also analyzed the effect of our collision handling scheme (using repulsion force) on the solution accuracy. Our algorithms are extremely efficient and show good strong and weak scalability on distributed memory architectures.

## Appendix A. Spherical harmonic basis functions

The spherical harmonic function of degree $n$ ($n = 0, 1, 2, \cdots$) and order $m$ ($|m| \leq n$) is defined as

$$Y_{nm}(\theta, \phi) = \frac{1}{\sqrt{2\pi}} P_{nm}(\cos\theta) e^{im\phi} \tag{A.1}$$

where, $P_{nm}$ is the normalized associated Legendre polynomial of degree $n$ and order $m$. The normalization is such that $\int_{-1}^{1} P_{lm}(x)P_{nm}(x)\,\mathrm{d}x = \delta_{lm}$ for any fixed $m$.

**References**


[1] A. Rahimian, S. K. Veerapaneni, D. Zorin, G. Biros, Boundary integral method for the flow of vesicles with viscosity contrast in three dimensions, Journal of Computational Physics 298 (2015) 766–786.

[2] J. Beaucourt, F. Rioual, T. Séon, T. Biben, C. Misbah, Steady to unsteady dynamics of a vesicle in a flow, Physical Review E 69 (1) (2004) 011906.

[3] S. R. Keller, R. Skalak, Motion of a tank-treading ellipsoidal particle in a shear flow, Journal of Fluid Mechanics 120 (1982) 27–47.

[4] M. Kraus, W. Wintz, U. Seifert, R. Lipowsky, Fluid vesicles in shear flow, Physical Review Letters 77 (17) (1996) 3685–3688.

[5] S. K. Veerapaneni, D. Gueyffier, G. Biros, D. Zorin, A numerical method for simulating the dynamics of 3D axisymmetric vesicles suspended in viscous flows, Journal of Computational Physics 228 (19) (2009) 7233–7249.

[6] S. K. Veerapaneni, A. Rahimian, G. Biros, D. Zorin, A fast algorithm for simulating vesicle flows in three dimensions, Journal of Computational Physics 230 (14) (2011) 5610–5634.

[7] L. Ying, G. Biros, D. Zorin, A high-order 3d boundary integral equation solver for elliptic pdes in smooth domains, Journal of Computational Physics 219 (1) (2006) 247–275.

[8] D. Malhotra, G. Biros, PVFMM: A Parallel Kernel Independent FMM for Particle and Volume Potentials, Communications in Computational Physics 18 (2015) 808–830.

[9] B. Quaife, G. Biros, Adaptive time stepping for vesicle suspensions, Journal of Computational Physics 306 (2016) 478–499.

[10] B. Quaife, G. Biros, High-order adaptive time stepping for vesicle suspensions with viscosity contrast, Procedia IUTAM 16 (2015) 89–98.

[11] A. Rahimian, I. Lashuk, S. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, D. Zorin, G. Biros, Petascale direct numerical simulation of blood flow on 200K cores and heterogeneous architectures, in: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 1–11. doi:http://dx.doi.org/10.1109/SC.2010.42.

[12] M. Loewenberg, E. J. Hinch, Numerical simulation of a concentrated emulsion in shear flow, Journal of Fluid Mechanics Digital Archive 321 (-1) (1996) 395–419.

[13] M. Loewenberg, Numerical simulation of concentrated emulsion flows, Journal of Fluids Engineering-Transactions of the ASME 120 (4) (1998) 824–832.

[14] A. Zinchenko, R. Davis, Shear flow of highly concentrated emulsions of deformable drops by numerical simulations, Journal of Fluid Mechanics 455 (2002) 21–61.

[15] A. Zinchenko, R. H. Davis, Large-scale simulations of concentrated emulsion flows, Philosophical Transactions Of The Royal Society Of London Series A-Mathematical Physical And Engineering Sciences 361 (1806) (2003) 813–845.

[16] H. Zhao, A. H. Isfahani, L. N. Olson, J. B. Freund, A spectral boundary integral method for flowing blood cells, Journal of Computational Physics 229 (10) (2010) 3726–3744. doi:10.1016/j.jcp.2010.01.024.

[17] B. Quaife, G. Biros, High-volume fraction simulations of two-dimensional vesicle suspensions, Journal of Computational Physics 274 (2014) 245–267. doi:10.1016/j.jcp.2014.06.013.

[18] I. Graham, I. Sloan, Fully discrete spectral boundary integral methods for helmholtz problems on smooth closed surfaces in $\mathbb{R}^3$, Numerische Mathematik 92 (2) (2002) 289–323. doi:10.1007/s002110100343.

[19] Z. Gimbutas, S. Veerapaneni, A fast algorithm for spherical grid rotations and its application to singular quadrature, SIAM Journal on Scientific Computing 35 (6) (2013) A2738–A2751. doi:10.1137/120900587.

[20] S. Tlupova, J. T. Beale, Nearly singular integrals in 3d stokes flow, Communications in Computational Physics 14 (05) (2013) 1207–1227. doi:10.4208/cicp.020812.080213a.

[21] A. Klöckner, A. Barnett, L. Greengard, M. O'Neil, Quadrature by expansion: A new method for the evaluation of layer potentials, Journal of Computational Physics 252 (2013) 332–349. doi:10.1016/j.jcp.2013.06.027.

[22] L. af Klinteberg, A.-K. Tornberg, A fast integral equation method for solid particles in viscous flow using quadrature by expansion, Journal of Computational Physics 326 (2016) 420–445. doi:10.1016/j.jcp.2016.09.006.

[23] E. Corona, L. Greengard, M. Rachh, S. Veerapaneni, An integral equation formulation for rigid bodies in stokes flow in three dimensions, Journal of Computational Physics 332 (2017) 504–519. doi:10.1016/j.jcp.2016.12.018.

[24] L. Greengard, The Rapid Evaluation of Potential Fields in Particle Systems, MIT Press, Cambridge, Mass., 1988.

[25] A.-K. Tornberg, L. Greengard, A fast multipole method for the three-dimensional Stokes equations, Journal of Computational Physics 227 (3) (2008) 1613–1619.

[26] Z. Gimbutas, L. Greengard, FMMLIB3D 1.2, 2012, http://www.cims.nyu.edu/cmcl/fmm3dlib/fmm3dlib.html.

[27] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole method in two and three dimensions, Journal of Computational Physics 196 (2) (2004) 591–626.

[28] Y. Liu, Fast Multipole Boundary Element Method, Cambridge University Press, 2009. doi:10.1017/cbo9780511605345.

[29] J. Freund, Leukocyte margination in a model microvessel, Physics of Fluids 19 (2007) 023301.

[30] L. Lu, A. Rahimian, D. Zorin, Contact-aware simulations of particulate stokesian suspensions, Journal of Computational Physics 347 (2017) 160–182. doi:10.1016/j.jcp.2017.06.039.

[31] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, Cambridge University Press, Cambridge, 1992.

[32] C. Pozrikidis, Effect of membrane bending stiffness on the deformation of capsules in simple shear flow, Journal of Fluid Mechanics 440 (2001) 269–291.

[33] J. L. Weiner, On a problem of Chen, Willmore, et al., Indiana University Mathematics Journal 27 (1978) 19–35.

[34] C. Pozrikidis, Interfacial dynamics for Stokes flow, Journal of Computational Physics 169 (2001) 250–301.

[35] Z. Gimbutas, L. Greengard, Short note: A fast and stable method for rotating spherical harmonic expansions, J. Comput. Phys. 228 (16) (2009) 5621–5627.

[36] H. Sundar, D. Malhotra, G. Biros, Hyksort: a new variant of hypercube quicksort on distributed memory architectures, in: Proceedings of the 27th international ACM conference on International conference on supercomputing, ACM, 2013, pp. 293–302.

[37] D. Abreu, M. Levant, V. Steinberg, U. Seifert, Fluid vesicles in flow, Advances in Colloid and Interface Science 208 (2014) 129–141.

[38] M. Delfour, J.-P. Zolsio, Shapes and Geometries: Metrics, Analysis, Differential Calculus, and Optimization, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2010.

[39] C. T. Kelley, D. E. Keyes, Convergence analysis of pseudo-transient continuation, SIAM Journal on Numerical Analysis 35 (1998) 508–523.