A scalable computational platform for particulate Stokes suspensions

Wen Yan[a,∗], Eduardo Corona[c], Dhairya Malhotra[b], Shravan Veerapaneni[d], Michael Shelley[a,b]

[a]*Center for Computational Biology, Flatiron Institute, Simons Foundation*
[b]*Courant Institute of Mathematical Sciences, New York University*
[c]*Department of Mathematics, New York Institute of Technology*
[d]*Department of Mathematics, University of Michigan*

Abstract

We describe a computational framework for simulating suspensions of rigid particles in Newtonian Stokes flow. One central building block is a collision-resolution algorithm that overcomes the numerical constraints arising from particle collisions. This algorithm extends the well-known complementarity method for non-smooth multi-body dynamics to resolve collisions in dense rigid body suspensions. This approach formulates the collision resolution problem as a linear complementarity problem with geometric 'non-overlapping' constraints imposed at each time-step. It is then reformulated as a constrained quadratic programming problem and the Barzilai-Borwein projected gradient descent method is applied for its solution. This framework is designed to be applicable for any convex particle shape, e.g., spheres and spherocylinders, and applicable to any Stokes mobility solver, including the Rotne-Prager-Yamakawa approximation, Stokesian Dynamics, and PDE solvers (e.g., boundary integral and immersed boundary methods). In particular, this method imposes Newton's Third Law and records the entire contact network. Further, we describe a fast, parallel, and spectrally-accurate boundary integral method tailored for spherical particles, capable of resolving lubrication effects. We show weak and strong parallel scalings up to 8×10^4 particles with approximately 4×10^7 degrees of freedom on 1792 cores. We demonstrate the versatility of this framework with several examples, including sedimentation of particle clusters, and active matter systems composed of ensembles of particles driven to rotate.

1. Introduction

Particulate suspensions are important to both technology and fundamental science. The presence of particles suspended in fluid leads to rich rheological behaviors of the mixture [1], and novel applications. For example, shear-thickening colloidal fluids have been used to reinforce the Kevlar-woven fabrics [2] in bulletproof vests. Colloidal suspensions are important model systems in the study of jamming [3] and phase transitions [4]. Active suspensions [5] are prototypes of active matter [6], where the immersed particles may be self-propelled, driven to rotate, or coupled together by biologically active cross-linkers [7, 8].

Here we discuss simulating the dynamics of particulate suspensions in a Newtonian solvent. The accurate predictions of the dynamic properties of such systems are key to understanding their behaviors and to conceiving new applications. However, simulating such systems is difficult. First, the intrinsic length scales in Stokes suspensions can be large due to long-range many-body hydrodynamic interactions (HIs hereafter). Second, boundary conditions (no-slip, slip, electrostatic, magnetic, etc.) on each particle must be accurately satisfied. Otherwise, for example, the osmotic pressure measured from dynamic simulations may show significant deviations from its true value due to the numerical error in resolving particle-particle collisions [9]. Third, such mixture systems must often be tracked for long times due to slow relaxation processes or significant Brownian noise in the system.

Many numerical methods have been developed to simulate suspensions in a Newtonian solvent. The Rotne-Prager-Yamakawa tensor [10, 11, 12, 13, 14, 15, 16, 17] is a popular approximation to account for HIs. This

---

method keeps the mobility matrix $\mathscr{M}$ symmetric-positive-definite (SPD), but is rather crude for dense colloidal suspensions. A more accurate method is Stokesian Dynamics [18, 19, 20, 21, 22], which splits the HIs into a far-field and near-field part. The far-field part represents the HIs through multipole expansions truncated at the stresslet level. The near-field lubrication effects are then added pairwise between particles with asymptotic lubrication resistance functions. Stokesian Dynamics has also been extended to electrorheological suspensions [23, 24] and non-spherical particles [25]. However, it is difficult to improve the accuracy of Stokesian Dynamics further due to the algebraic complexity of including higher force moments on particles beyond the stresslet level.

In all such computational methods, the efficient handling of collisions between particles remains a long-standing problem. Theoretically, for smooth rigid particles with no-slip boundary conditions and moving under finite forces, lubrication effects prevents their collision [26]. However, collisions are inevitable in simulations because numerically it is impossible to fully resolve the opposing lubrication effects with the finite time-step sizes and finite accuracy of the HI solvers. Further, particles with surface roughness or surface slip velocity may actually collide. This feature is believed to underly the observed loss of flow reversibility in the shearing of non-Brownian, "spherical" particle suspensions [27]. In simulations, such collisions are usually handled, or resolved, by using a prescribed short-range, pairwise, repulsive potential between particles. For example, an exponentially decaying pairwise repulsive force is used in Stokesian Dynamics. Such pairwise potentials must be steep and therefore imposes a restrictive upper bound on the time-step size to maintain stability. Penalty methods [28] work similarly and suffer from the same stability constraints. [1]

To build a stable and efficient numerical scheme, we reconsider the collision resolution method. Pairwise potentials and penalty methods compute collision forces on each particle at the start of each time-step. Alternatively, collision forces can be solved for by imposing non-overlapping constraints between bodies on particle configurations over time. In this way, the stability issues induced by the stiffness induced by potentials or penalties can be avoided. Foss and Brady [9] applied a 'potential-free' algorithm in this fashion, where overlaps are resolved by iteratively moving each overlapping pair of particles apart until a non-overlapping configuration is achieved. This scheme, however, may converge slowly in dense systems and does not allow many-body HI coupling between particles. Another method is the constrained minimization scheme developed by Maury [29]. This method works as a prediction-correction scheme, where a predicted velocity U_p for each particle is first computed, and then a corrected velocity U satisfying the no-overlap constraints is obtained by minimizing the L_2 norm of the difference $U - U_p$ for overdamped systems. For underdamped systems, the objective function being optimized is modified to include the effects of mass and acceleration. This scheme, however, does not determine collision forces between each colliding pair, making it difficult to compute the mechanical stress induced by collisions.

Non-overlapping constraints can be coupled with collision forces in inelastic collisions to construct complementarity constraints, since each close pair of particles must be at one of the two possible states: (1) they have collided, and so the collision force is positive and their minimal separation is zero, or (2) they did not collide, and so the collision force is zero and the minimal separation is positive. The early development of these methods formulated collision resolution problems in rigid body dynamics in various forms and probed their mathematical properties, for example, the existence of solutions [30, 31, 32, 33, 34]. A linear complementarity formulation was soon developed to practically simulate collections of frictionless [35] and frictional [36, 37] rigid particles. This formulation uses a first-order Euler temporal discretization and solves linear complementarity problems at each time-step. This method has been proved [38, 39] to generate convergent particle trajectories as the time-step $\Delta t \to 0$. These early developments have been summarized by Stewart [40], and extended to handle stiff external forces [41] and bilateral constraints such as mechanical joints [42]. This early formulation, however, considered non-convex linear complementarity formulations for frictional particles and has been superseded by the modern formulation of Anitescu [43], where a convex cone quadratic program is solved at each time-step. This convexity allows effient numerical solution in large scale simulations for frictional granular flow problems [44, 45, 46, 47, 48, 49, 50], and open source software implementations of this method are now available [51, 52]. Recently, this formulation has been extended to deformable particles [53, 54]. To speed up these large scale simulations, various iterative solvers have been discussed and compared in [55, 56, 57, 58, 59]. Recent work by two of the co-authors (Corona and Veerapaneni) and their co-workers demonstrated the scalability of these methods

---

[1] In one interpretation, collision resolution algorithms are also models for the irreversible processes associated with the collisions of physical, and hence rough, particles.

on large distributed memory machines, simulating the frictional multibody dynamics of hundreds of millions of granular particles [60].

For particulate Stokes flow problems, the recent work of Lu et al. [61] applied a similar explicit contact constraint enforcement approach for simulating deformable particles. Furthermore, this approach was extended to three dimensional problems in [62]. Both works demonstrated impressive gains in computational efficiency by improving the stability and accuracy of the underlying numerical solvers. However, we note that their implementation is tightly bound to a particular boundary integral (BI) fluid solver. Moreover, the computed collision forces do not follow Newton's Third Law where collision forces for a colliding pair must be equal and opposite. Therefore, although the collision forces are explicitly computed in this method, the collision stress may still be incorrect.

A proper extension of the complementarity method to Stokes particulate suspensions is difficult because the particle motion is overdamped in contrast to granular flows dominated by inertia. This requires significant reformulation of the complementarity problem. The solver must also be improved because the complementarity problem now involves a full dense matrix due to the many-body hydrodynamic coupling in Stokes suspensions. Yan et al. [63] presented a method to resolve normal collisions for arbitrarily-shaped rigid particles in Stokes suspensions, as an extension of [35]. The complementarity problem for collision resolution is reformulated and a different but more efficient solution algorithm is described, utilizing the symmetric-positive-definiteness of the mobility matrix. This approach is generic because the motion induced by collision force is represented by this mobility matrix, and any hydrodynamic solver can be used. This method imposes Newton's Third Law and guarantees the symmetry and translational invariance of the collision stress tensor. They validated this method by computing the equation of state for Brownian spherocylinders, and demonstrated its application to various self-propelled rod systems.

Although the method is generic, results presented in [63] neglected the many-body HI coupling. In this work, we extend this generic method to Stokes particulate suspensions with full many-body HIs. We use BI methods specialized for spheres and achieving spectral accuracy by spherical harmonic expansions for singular and near-singular integration [64]. The implementation is fully parallelized with hybrid `OpenMP` and `MPI` to achieve scalability. We demonstrate the application of this platform to various problems, such as sedimentation of particle ensembles, and the collective dynamics of rotor systems.

The latter class of many-particle systems has been studied of late as an active matter system, showing aspects such as activity-induced phase separation [65], crystallization [66], odd rheological and surface flow dynamics [67], and forms of active turbulence [68]. Motivated by the simulations of [65], Fig. 1 shows a large-scale simulation of $20,000$ particles, densely packed and suspended on a plane in the fluid, with each particle driven by an external out-of-plane torque. This simulation has roughly 6×10^6 degrees of freedom, and was run on 576 CPU cores. The simulation shows the development of an extensive and inhomogeneous fine-scaled collision network, as well as the development of large-scale collective rotation induced by long-ranged hydrodynamic coupling. Of scientific interest is the interaction of multiple such ensembles, and the detailed structure of particle flows within them.
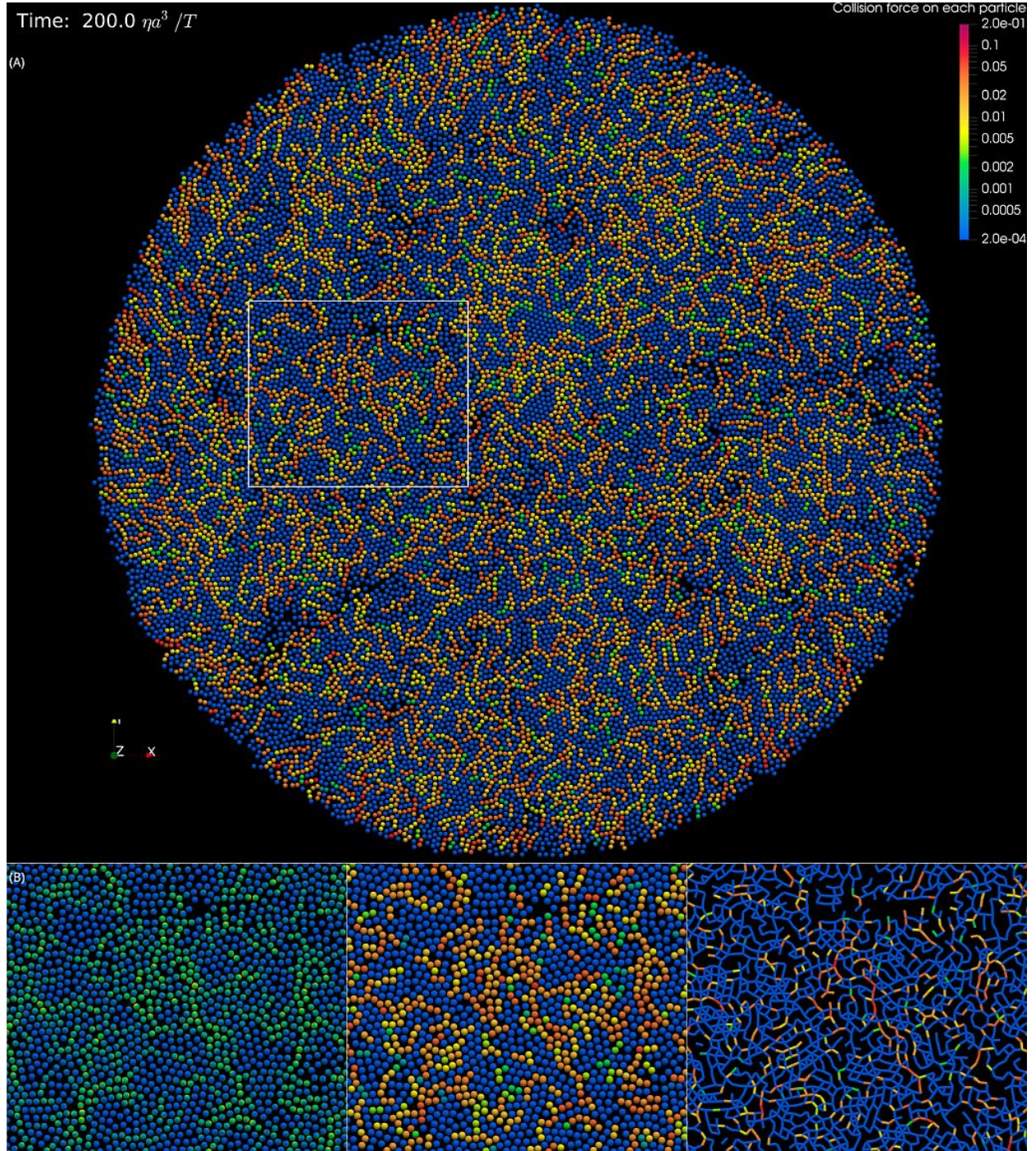
Figure 1: A simulation snapshot of a monolayer of 20000 spherical particles with radii a at approximately 60% area fraction. A constant external torque $\boldsymbol{T}$ in the direction perpendicular to this monolayer is exerted on each particle to drive a counter-clockwise planar rotation of this mobolayer. (A) shows the entire simulation, with each particle colored by its total collision force magnitude. (B) shows the region inside the white box. The left panel of (B) shows the hydrodynamic force density (traction) on the sphere surfaces induced by collisions, scaled by T/a^3. The center and right panels show the net collision forces, scaled by T/a, on each particle and on each collision constraint, respectively. The three panels of (B) are colored by the same colormap as in (A).

4

2. Problem formulation

In this section we briefly summarize the well-known mobility problem, i.e., the motion of arbitrarily-shaped particles driven by forces and torques and immersed in a 3D Newtonian Stokes flow. Here, for simplicity, the fluid occupies all space but the particle volumes. Hence, consider a suspension of N rigid particles, like that shown in Fig. 1 but not limited to spherical shapes. Let $\{V_j, \Gamma_j, \boldsymbol{c}_j\}_{j=1}^N$ denote the volume, boundary, and tracking points of the particles, respectively. Here, the 'tracking point' refers to any point on one particle, not limited to its center-of-mass, because in this manuscript we consider only overdamped dynamics of particles where inertia can be neglected. Each particle moves with a translational velocity $\boldsymbol{U}_{j,e}$ and angular velocity $\boldsymbol{\Omega}_{j,e}$, in response to an externally applied body force $\boldsymbol{F}_{j,e}$ and a body torque $\boldsymbol{T}_{j,e}$.

Denoting the fluid viscosity by η, the fluid velocity by $\boldsymbol{u}$, and the pressure by p, the fluid stress is $\boldsymbol{\sigma} = -p\boldsymbol{I} + \eta\left[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right]$. Neglecting inertial and body forces gives that $\nabla\cdot\boldsymbol{\sigma} = 0$ in the fluid domain. Assuming fluid incompressibility, this gives the Stokes equations:

$$-\nabla p + \eta\nabla^2\boldsymbol{u} = 0 \quad \text{in} \quad \mathbb{R}^3 \setminus \cup_j V_j, \tag{1a}$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{in} \quad \mathbb{R}^3 \setminus \cup_j V_j, \tag{1b}$$

$$\boldsymbol{u} \to 0 \quad \text{as} \quad \boldsymbol{x} \to \infty. \tag{1c}$$

The traction $\boldsymbol{f}$, i.e., the hydrodynamic force density on the particle surface applied by the fluid, is determined by the stress tensor $\boldsymbol{\sigma}$ as $\boldsymbol{f} = \boldsymbol{\sigma}\cdot\boldsymbol{n}$, where the surface normal $\boldsymbol{n}$ points into the fluid domain. The solution $(\boldsymbol{u}_e, p_e)$ to Eq. (1)(a-c) is subject to the boundary conditions:

$$\boldsymbol{u}_e = \boldsymbol{U}_{j,e} + \boldsymbol{\Omega}_{j,e} \times (\boldsymbol{x} - \boldsymbol{c}_j) \quad \text{on} \quad \Gamma_j, \forall j, \tag{2a}$$

$$\int_{\Gamma_j} \boldsymbol{f}_e \, dS = \boldsymbol{F}_{j,e}, \quad \forall j, \tag{2b}$$

$$\int_{\Gamma_j} (\boldsymbol{x} - \boldsymbol{c}_j) \times \boldsymbol{f}_e \, dS = \boldsymbol{T}_{j,e}, \quad \forall j. \tag{2c}$$

There may also be collision a force $\boldsymbol{F}_{j,c}$ and torque $\boldsymbol{T}_{j,c}$ on each particle j applied by other particles. The consequent velocities $(\boldsymbol{U}_{j,c}, \boldsymbol{\Omega}_{j,c})$ induced by $\boldsymbol{F}_{j,c}$ and $\boldsymbol{T}_{j,c}$ satisfy the same Stokes equation (1) and boundary conditions:

$$\boldsymbol{u}_c = \boldsymbol{U}_{j,c} + \boldsymbol{\Omega}_{j,c} \times (\boldsymbol{x} - \boldsymbol{c}_j) \quad \text{on} \quad \Gamma_j, \forall j, \tag{3a}$$

$$\int_{\Gamma_j} \boldsymbol{f}_c \, dS = \boldsymbol{F}_{j,c}, \quad \forall j, \tag{3b}$$

$$\int_{\Gamma_j} (\boldsymbol{x} - \boldsymbol{c}_j) \times \boldsymbol{f}_c \, dS = \boldsymbol{T}_{j,c}, \quad \forall j. \tag{3c}$$

Due to the linearity of the Stokes equation (1) and boundary conditions Eqs. (2) and (3), the overall solution is simply the superposition of the two separate mobility problems induced by external and collisional forces and torques:

$$\boldsymbol{u} = \boldsymbol{u}_e + \boldsymbol{u}_c, \quad p = p_e + p_c, \quad \boldsymbol{f} = \boldsymbol{f}_e + \boldsymbol{f}_c, \quad \boldsymbol{U}_j = \boldsymbol{U}_{j,e} + \boldsymbol{U}_{j,c}, \quad \boldsymbol{\Omega}_j = \boldsymbol{\Omega}_{j,e} + \boldsymbol{\Omega}_{j,c}. \tag{4}$$

Conventionally, the solution to a mobility problem can be written compactly as $\mathscr{U} = \mathscr{M}\mathscr{F}$, where $\mathscr{U} = \left(\ldots, U_j^x, U_j^y, U_j^z, \Omega_j^x, \Omega_j^y, \Omega_j^z, \ldots\right)$ and $\mathscr{F} = \left(\ldots, F_j^x, F_j^y, F_j^z, T_j^x, T_j^y, T_j^z, \ldots\right)$ are both column vectors with 6 degrees of freedom per particle. $\mathscr{M}$ is the *mobility matrix*, which is a dense square matrix containing all the information of the Stokes equation and boundary conditions. The solutions given by Eq. (2) and Eq. (3) can be written compactly as $\mathscr{U}_e = \mathscr{M}\mathscr{F}_e$ and $\mathscr{U}_c = \mathscr{M}\mathscr{F}_c$, respectively.

$\mathcal{M}$ is usually not formed explicitly because of the difficulty and high cost of computing its entries, except in cases where the many-body coupling of HIs is treated with very crude approximations [10, 11]. Instead, a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is usually constructed to solve the mobility problem. The linear operator $\boldsymbol{A}$ is usually computed from the geometry and boundary conditions, containing all the information of $\mathcal{M}$. The right hand side vector $\boldsymbol{b}$ is usually computed from $\mathscr{F}$, and the velocities $\mathscr{U}$ are computed according to the solution $\boldsymbol{x}$. The linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is ususally solved iteratively.

There are a variety of methods to construct the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Stokesian Dynamics [19, 25, 21] forms the linear system using multipole expansions for both spherical and non-spherical particles. BI methods forms the linear system using boundary integral operators and discretization of particle surfaces [69]. When high levels of accuracy are necessary, BI methods provide the most accurate and efficient solvers to the mobility problem; for example, see [70, 64, 71, 72] and references therein.

It is well-known that $\mathcal{M}$ is symmetric-positive-definite [18, 26]. Physically, this is related to the dissipative nature of a Stokes suspensions. The energy injected by driving forces and torques is always dissipated instantaneously by the fluid flow. This is the key to our collision resolution algorithm.

3. Collision resolution algorithm

In this section we describe our collision resolution algorithm in detail, making no assumptions on the particle shape, or on the numerical method to solve the mobility problem $\mathscr{U} = \mathcal{M}\mathscr{F}$. The dynamics of particulate Stokes suspensions is overdamped, i.e., the particle inertia is negligible. In this case, collisions are inelastic with a zero coefficient of restitution. Consequently, a colliding pair of particles remain in contact until they are driven apart by flow or collisions with other particles. This is different from granular flow, where inertial effects dominate and particles may rebound after collisions, depending on their coefficients of restitution. We also ignore inter-particle friction for Stokes suspensions since, physically, hydrodynamic lubrication effects dominate for smooth particles close to contact.

3.1. Complementarity formulation and equation of motion

In general, Eq. (4) can be generalized as:

$$\mathscr{U} = \mathscr{U}_{nc} + \mathscr{U}_c. \tag{5}$$

Here the subscript nc stands for all non-collisional motion, such as the motion $\mathscr{U}_e$ driven by external forces and torques. In other problems, $\mathscr{U}_{nc}$ may originate from prescribed motions or other physical processes, such as Brownian fluctuations or electrophoresis.

The geometric configuration of a collection of rigid particles in 3D space is fully specified by the tracking point location $\boldsymbol{c}_j$ and the orientation unit quaternion $\boldsymbol{\theta}_j = \{s, \boldsymbol{p}\} \in \mathbb{R}^4$ of each particle j. The temporal evolution of $\boldsymbol{c}_j$ and $\boldsymbol{\theta}_j$ is given by:

$$\dot{\boldsymbol{c}}_j = \boldsymbol{U}_j, \tag{6}$$

$$\dot{\boldsymbol{\theta}}_j = \boldsymbol{\Psi}_j \boldsymbol{\Omega}_j, \quad \text{where} \quad \boldsymbol{\Psi}_j = \frac{1}{2}\begin{bmatrix} -\boldsymbol{p}_j^T \\ s_j\boldsymbol{I} - \boldsymbol{p}_j \end{bmatrix} \in \mathbb{R}^{4\times 3}. \tag{7}$$

Here we follow the rigid body kinematic equation using quaternions by Delong et al. [73]. Similar to $\mathscr{F}$ and $\mathscr{U}$, we define the configuration $\mathscr{C}$ as a column vector with $7N$ entries, containing $\boldsymbol{c}_j$ and $\boldsymbol{\theta}_j$ for all N particles. The overdamped equation of motion for these particles can be written compactly as:

$$\dot{\mathscr{C}} = \mathscr{G}\mathscr{U} = \mathscr{G}\mathscr{U}_{nc} + \mathscr{G}\mathcal{M}\mathscr{F}_c. \tag{8}$$

where $\mathscr{G} \in \mathbb{R}^{7N\times 6N}$ is a block diagnoal matrix, containing 3×3 identity matrices and 4×3 matrices $\boldsymbol{\Psi}_j$ corresponding to $\boldsymbol{c}_j$ and $\boldsymbol{\theta}_j$ for each particle j, respectively. Note that $\mathscr{G}$ depends on those quaternion $\boldsymbol{\theta}$ components in $\mathscr{C}$, but not on those $\boldsymbol{c}$ components. If the orientation of each particle is represented with Euler angles or rotation matrices, the kinematic equation can still be written in the form of Eq. (8), but the dimension of $\mathscr{C}$ and the

definition of each Ψ_j block must both be adjusted accordingly. However, different representations of orientation does not affect the derivation of collision resolution algorithms [39].

The equation of motion Eq. (8) should be augmented by geometric constraints to generate trajectories without overlaps between particles. The constraints are simply that the minimal separation distance $\Phi_\ell(\mathscr{C})$ between each pair ℓ of close rigid objects remain non-negative for all configurations $\mathscr{C}$. In total there are potentially $n_c = N(N+1)/2$ such close particle-particle pairs. For each pair ℓ, we denote the collision force magnitude between this pair of particles as γ_ℓ. The pair (Φ_ℓ, γ_ℓ) must satisfy one of two conditions:

$$\text{No contact:} \quad \Phi_\ell > 0, \quad \gamma_\ell = 0. \tag{9}$$
$$\text{In contact:} \quad \Phi_\ell = 0, \quad \gamma_\ell > 0. \tag{10}$$

Mathematically, this yields a *complementarity problem*, written vertically over the set of pairs as:

$$0 \leq \boldsymbol{\Phi} \perp \boldsymbol{\gamma} \geq 0, \tag{11}$$

where $\boldsymbol{\Phi} = (\Phi_0, \Phi_1, ...) \in \mathbb{R}^{n_c}$ and $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, ...) \in \mathbb{R}^{n_c}$ denote the collections of minimal distances and contact force magnitudes for each ℓ.

For N rigid particles, let $\boldsymbol{D}_\ell \in \mathbb{R}^{6N}$ be a sparse column vector mapping the magnitude γ_ℓ to the collision force (and torque) vector applied to each particle. We define a sparse matrix $\mathscr{D} \in \mathbb{R}^{6N \times n_c}$ as a collection of all $\boldsymbol{D}_\ell$:

$$\mathscr{D} = \begin{bmatrix} \boldsymbol{D}_0 \ \boldsymbol{D}_1 \ ... \ \boldsymbol{D}_{n_c} \end{bmatrix} \in \mathbb{R}^{6N \times n_c}, \quad \boldsymbol{\mathscr{F}}_c = \mathscr{D}\boldsymbol{\gamma}. \tag{12}$$

For each pair ℓ, $\boldsymbol{D}_\ell$ defined in this way has 12 non-zero entries for non-spherical shapes, corresponding to 3 translational and 3 rotational degrees of freedom for each particle. For the special cases of two spheres, $\boldsymbol{D}_\ell$ has 6 non-zero entries, as normal collision forces induce no torques on spheres. The entries of each $\boldsymbol{D}_\ell$ are explicitly given in [35]. For rigid particles, there is an important relation between the transpose of $\mathscr{D}$ and $\boldsymbol{\Phi}$ [35]:

$$\mathscr{D}^T = (\nabla_\mathscr{C}\boldsymbol{\Phi})\mathscr{G}. \tag{13}$$

This relation holds for different choices of orientation representations used in $\mathscr{C}$ and $\mathscr{G}$, including quaternions and Euler angles [35].

Combining Eq. (8), CP (11), and Eq. (12), we reach a differential variational inequality (DVI) for N particles and $N(N+1)/2$ constraints.

$$\dot{\mathscr{C}} = \mathscr{G}\boldsymbol{\mathscr{U}}_{nc} + \mathscr{G}\mathscr{M}\mathscr{D}\boldsymbol{\gamma}, \tag{14a}$$
$$0 \leq \boldsymbol{\Phi}(\mathscr{C}) \perp \boldsymbol{\gamma} \geq 0. \tag{14b}$$

Here $\mathscr{G}$, $\boldsymbol{\mathscr{U}}_{nc}$, $\mathscr{M}$, $\mathscr{D}$, and $\boldsymbol{\Phi}$ depend only on the geometric configuration $\mathscr{C}$. This DVI is solvable and integrable over time once a relation between the configuration $\mathscr{C}$ and the collision force $\boldsymbol{\gamma}$ is supplied, that is, a time-stepping scheme. Higher order schemes such as the Runge-Kutta scheme can be used, but for simplicity of presentation we derive a first order Euler integration scheme here.

3.2. Time-Stepping and linearization

With a given time-step Δt, the total number of constraints n_c can be greatly reduced from $N(N+1)/2$ to $O(N)$, since only those pairs that are close enough to be possibly in contact within this time-step need to be included in the constraints. For example, for hexagonal close packing of spheres, every sphere is in contact with 12 neighbors. This gives $n_c = 6N$. We define $\mathscr{A}$ as the set of pairs of bodies 'close-to-collision', that is, for which the minimal separation distance function Φ_ℓ is smaller than a positive threshold δ:

$$\mathscr{A}(\mathscr{C}, \delta) = \{\ell \,|\, \Phi_\ell(\mathscr{C}) \leq \delta\}. \tag{15}$$

The choice of δ depends on the particle velocity $\boldsymbol{\mathscr{U}}$ and the time-step Δt, and should be taken large enough so that no possible collisions are missed in the collision resolution algorithm. For each pair $\ell \in \mathscr{A}$, the non-overlap

condition can be simply stated as $\Phi_\ell(\mathscr{C}) \geq 0$. Increasing δ gives a larger set $\mathscr{A}$, and increases the dimension of the CP (11). Empirically, we set δ to 30% of the sum of the radii for each pair of spherical particles.

The time-stepping scheme should evolve a non-overlapping configuration $\mathscr{C}^k$ at t^k to a non-overlapping $\mathscr{C}^{k+1}$ at t^{k+1}. Therefore, for the discretized version of the DVI (14), the complementarity condition is between the geometry at the end of this time-step $\Phi(\mathscr{C}^{k+1})$ and the collision force at this time-step γ^k:

$$\mathscr{C}^{k+1} = \mathscr{C}^k + \mathscr{G}^k \mathscr{U}_{nc}^k \Delta t + \mathscr{G}^k \mathscr{M}^k \mathscr{D}^k \gamma^k \Delta t, \tag{16}$$

$$0 \leq \Phi\left(\mathscr{C}^{k+1}\right) \perp \gamma^k \geq 0. \tag{17}$$

This is a nonlinear complementarity problem (NCP) because the minimum separation Φ is a nonlinear function of $\mathscr{C}$. For first-order Euler time-stepping, this NCP can be linearized as an LCP by taking a one-term Taylor expansion of Φ over $\mathscr{C}$:

$$0 \leq \Phi(\mathscr{C}^k) + (\nabla_\mathscr{C} \Phi)^k \mathscr{G}^k \mathscr{U}_{nc}^k \Delta t + (\nabla_\mathscr{C} \Phi)^k \mathscr{G}^k \mathscr{M}^k \mathscr{D}^k \gamma^k \Delta t \perp \gamma^k \geq 0. \tag{18}$$

At each time-step t^k we solve the LCP (18) to compute the collision force magnitude γ. Then we use Eq. (16) to update the geometry of all the particles. In summary, the procedures of this LCP method include:

1. Compute $\mathscr{U}_{nc}^k$ at time-step k.

2. Construct the sparse matrix $\mathscr{D}^k$ given the geometric configuration $\mathscr{C}^k$ and apply threshold δ for possible contacts.

3. Solve the LCP for γ^k. $\mathscr{U}_c^k$ and $\mathscr{F}_c^k$ are computed simultaneously.

4. Evolve the system to $\mathscr{C}^{k+1}$ with $\mathscr{U}_{nc}^k + \mathscr{U}_c^k$.

This derivation is mostly the same as in the granular flow case [39], except that inertia and friction are ignored. However, this does not simplify the problem, because the mobility matrix $\mathscr{M}$ is a full dense matrix, in contrast to the block-diagonal inertial and moment-of-inertia matrix in the granular flow case. The high cost involved in computing $\mathscr{M}\mathscr{D}\gamma$ requires careful attention to the solution strategy. Once the solution to LCP (18) is found, a subset $\mathscr{A}_c$ of $\mathscr{A}$ can be identified as the 'active constraints', where the collision force between two particles is non zero:

$$\mathscr{A}_c = \{\ell \,|\, \gamma_\ell > 0\} \subseteq \mathscr{A}. \tag{19}$$

3.3. LCP solvers

The success of the LCP collision resolution algorithm depends on the efficiency of the solver. In this section we briefly discuss the state-of-the-art solvers used in this work. The superscript k denoting the time-steps are dropped in this subsection to simplify notation.

The LCP defined in LCP (18) can be written as the following standard form, using Eq. (13):

$$0 \leq M\gamma + q \perp \gamma \geq 0, \tag{20a}$$

$$M = \mathscr{D}^T \mathscr{M} \mathscr{D}, \quad q = \frac{1}{\Delta t}\Phi(\mathscr{C}) + \mathscr{D}^T \mathscr{U}_{nc}, \tag{20b}$$

where M and q have been scaled by $1/\Delta t$ for convenience. The term $\mathscr{D}^T \mathscr{U}_{nc}$ computes the (linearized) change in the minimum separation function Φ due to non-collisional motion. Each evaluation of $M\gamma$ corresponds to the solution of a mobility problem for the contact force $\mathscr{F}_c = \mathscr{D}\gamma$. For large enough numbers of particles, it may thus be preferable to rely on matrix-free operations instead of forming M explicitly.

3.3.1. Iterative solution methods

The matrix M defined in LCP (20) is in general symmetric-positive-semidefinite (SPSD), because the mobility matrix $\mathcal{M}$ is SPD. Therefore the LCP can be conveniently converted to a convex constrained quadratic programming (CQP) [74]:

$$\gamma = \arg\min_{\gamma \geq 0}\left[f(\gamma) = \frac{1}{2}\gamma^T M \gamma + q^T \gamma \right]. \tag{21}$$

For convenience, we denote the gradient of the objective function $f(\gamma)$ as $g = \nabla f = M\gamma + q$. From a physics perspective, minimizing $f(\gamma)$ can be understood as minimizing the total virtual work done by collision forces. Popular methods to iteratively solve this CQP includes the second-order minimum-map Newton method, and the first-order projected gradient descent (PGD) method.

The minimum-map Newton method [74] proceeds by reformulating the LCP as a root-finding problem for the L_2 norm of the componentwise minimum-map function $H = \min(\gamma, g)$, which means taking the smaller entry at every corresponding component of the two vectors: $H_i = \min\{\gamma_i, g_i\}$.

$$\varphi(\gamma, g) = \|\min(\gamma, g)\|_2 = 0. \tag{22}$$

The solution γ to the CQP (21) is reached when $\varphi = 0$. This Newton-type method can achieve high accuracy, but the cost is usually much higher than first order methods because a linear system must be iteratively solved at every minimization step to compute the Newton step.

Various PGD methods have been proposed to solve this CQP efficiently because the gradient $g = M\gamma + q$ is straightforward to compute, and the constraint $\gamma \geq 0$ can be conveniently handled by setting the negative components of γ to 0 because negative components violate the non-negative constraint of the LCP problem. This is a 'projection' into the feasible region $\gamma \geq 0$, at each gradient descent step. Such first-order methods do not find the root to Eq. (22). Instead, Eq. (22) is used to check the convergence with a prescribed residual tolerance ϵ_{tol}. When $\varphi(\gamma, g) < \epsilon_{tol}$, the PGD iterations stop.

The key ingredient in all PGD methods is to choose a proper step-size α_j for the j-th GD step: $\gamma_{j+1} = \gamma_j - \alpha_j \nabla f(\gamma_j)$. The Accelerated PGD (APGD) by Mazhar et al. [57] is shown to be a competitive choice in a recent survey article [58] in the context of granular flow. However, the step size α_j chosen by APGD converges only when $\alpha_j < 1/L$, where L is the Lipschitz constant for the linear operator M and is usually not known a priori. Consequently, α_j is adaptively adjusted at each iteration to fit a local estimated Lipschitz constant L_j at the j-th step. This adjustment process is prohibitively expensive in our context since the gradient g is evaluated a large number of times, each of which requires solving a mobility problem.

A much more efficient method for Eq. (21) is the Barzilai-Borwein PGD (BBPGD) method [75, 76, 77]. This method has been successfully applied to our collision resolution method for cases where the many-body coupling of HIs are ignored [63]. It constructs the step size α_j from the previous two steps, and does not require any step size adjustment as APGD. A version of this method adapted to Eq. (21) is summarized by Algorithm 1. In this algorithm, the two different choices of step size α_j^{BB1} and α_j^{BB2} can be used either consistently throughout all steps, or alternatively for odd and even j-th steps. We find that there is almost no difference in performance for different choices of α_j^{BB1} or α_j^{BB2} (Step 15) in solving our problems, and we use α_j^{BB1} for all results reported in this work. In BBPGD, two evaluations, $M\gamma_0$ and Mg_0 (Step 2 and 6), are necessary before the first iteration, and thereafter only one evaluation $M\gamma_j$ is needed per iteration. In our numerical tests BBPGD shows a similar convergence rate as APGD, but each APGD iteration is significantly more expensive because of the necessity to check the Lipschitz condition.

3.3.2. Choice of ϵ_{tol}

The error bound for the general case SPSD matrix M [78] is highly complicated and there is no direct relation between the error bound to our residual function φ. However, for most practical simulations the matrix M is SPD. This is because if we assume that there is a vector $\gamma \neq 0$ such that $\gamma^T M \gamma = 0$, then $\mathcal{D}\gamma = 0$ because $\mathcal{M}$ is SPD in LCP (20). This means $\mathcal{D}$ must be rank-deficient for its column vectors. There exists at least one column vector D_ℓ linearly dependent on others. Geometrically, this is possible but only for very special shapes and configurations of particles. In most cases, the column vectors (constraints) in $\mathcal{D}$ are independent of each other, and $M = \mathcal{D}^T \mathcal{M} \mathcal{D}$ is SPD.

Algorithm 1 The Barzilai-Borwein Projected Gradient Descent method

---

1: Solve LCP (20) with initial guess γ_0.
2: $g_0 = M\gamma_0 + q$.
3: **if** $\varphi(\gamma_0, g_0) < \epsilon_{tol}$ **then**
4:     Solution is γ_0.
5: **end if**
6: Simple gradient-descent step size: $\alpha_0 = g_0^T g_0 / g_0^T M g_0$.
7: **for** $j = 1 : j_{\max}$ **do**
8:     Compute the descent step: $\gamma_j = \gamma_{j-1} - \alpha_{j-1}^{BB} g_{j-1}$.
9:     Projection: set the negative components of γ_j to zero.
10:    Compute the gradient: $g_j = M\gamma_j + q$.
11:    **if** $\varphi(\gamma_j, g_j) < \epsilon_{tol}$ **then**
12:        Stop iteration, solution is γ_j.
13:    **end if**
14:    Update: $s_{j-1} = \gamma_j - \gamma_{j-1}$, $y_{j-1} = g_j - g_{j-1}$.
15:    Update: $\alpha_j^{BB1} = s_{j-1}^T s_{j-1} / s_{j-1}^T y_{j-1}$ or $\alpha_j^{BB2} = s_{j-1}^T y_{j-1} / y_{j-1}^T y_{j-1}$. Set α_j^{BB} to either α_j^{BB1} or α_j^{BB2}.
16: **end for**

This allows us to use the simpler error bound for SPD matrices [79]. The absolute and relative error bounds for Eq. (21) are between an arbitrary vector γ and the exact solution γ^* to the LCP (20):

$$\|\gamma - \gamma^*\|_2 \leq \frac{\|M\|_2 + 1}{\lambda_{\min}} \varphi(\gamma, g), \tag{23}$$

$$\frac{\|\gamma - \gamma^*\|_2}{\|\gamma^*\|_2} \leq \text{cond}(M)(\|M\|_2 + 1) \frac{\varphi(\gamma, g)}{\|\max(0, -q)\|_2}. \tag{24}$$

Here $\lambda_{\min}$ is the smallest eigenvalue of M, $\|M\|_2$ is the L_2-norm of M, $\text{cond}(M)$ is the condition number of M, and $\max(., .)$ is the 'componentwise maximum map function', a counterpart to $H = \min(., .)$.

However, $\lambda_{\min}$, $\|M\|_2$, and $\text{cond}(M)$ are all very difficult to estimate, because $M = \mathscr{D}^T \mathscr{M} \mathscr{D}$ and only very crude estimates of $\mathscr{M}$ are available [80]: $\text{cond}(\mathscr{M}) \approx O(1)$ of dilute suspensions and $\text{cond}(\mathscr{M}) \approx O(10^4)$ for densely packed spherical particles. A detailed error bound analysis is beyond the scope of this work. Instead, we pick ϵ_{tol} based on physical intuition and practical considerations.

We choose to set ϵ_{tol} as an absolute error bound, following Eq. (23). This is because the collision force for different constraints can be orders of magnitude different, as illustrated by the center panel of Fig. 1 and in other results reported in Section §6 and Section §7. This is one of the physical features of rigid body suspensions, where the particle-particle interactions induced by perturbation may decay rapidly as the perturbation propagates and the particle-particle interaction in some regions of the system may be orders of magnitude larger or smaller than other regions. One example phenomena is Brinkman screening [81] where the perturbation is 'screened' hydrodynamically. If using the relative bound, as in Eq. (24), the large collision forces dominate the norm and the smaller entries in γ may be completely inaccurate, leading to inaccurate trajectories or even possibly overlapping configurations. We set $\epsilon_{tol} = 10^{-5}$ for all results reported in this work unless otherwise specified.

In other works on granular flow [56, 57], there are other residual functions used besides $\varphi(\gamma, g)$. This is because in granular flow the matrix M involves only a block diagonal inertial and moment-of-inertia matrix, instead of a full dense mobility matrix $\mathscr{M}$. Various matrix splitting and manipulation techniques can then be used to compute various residual functions. For our problem Eq. (21), we stick to $\varphi(\gamma, g)$ because of its negligible extra cost of computation at every gradient descent step, and because it is directly related to the theoretical error bounds Eqs. (23) and (24).

3.3.3. Extensions

In the above discussion we constructed the basic form of our collision resolution algorithm. There are two important extensions that can be easily incorporated.

The first straightforward extension is to include collisions between particles and external boundaries. Here a 'boundary' refers to any object either static or moving with a prescribed velocity. In other words, a 'boundary' does not appear in the mobility matrix $\mathcal{M}$. For each collision pair ℓ between a particle and a boundary, a vector D_ℓ can be added to $\mathcal{D}$ in Eq. (12). The only difference is that these D_ℓ column vectors have 6 non-zero entries corresponding to the degrees of freedom of the particle only. Once $\mathcal{D}$ is constructed including all these particle-boundary collisions, the solution to Eq. (21) remains exactly the same.

Another straightforward extension is to include particle motions other than those driven by external forces. For example, consider squirmers, which are rigid particle models for ciliated organisms [82, 83]. Squirmers propel themselves through a quiescent fluid, without any external driving force, by inducing a nonzero surface slip velocity $u_{j,slip}$ on the outer flow. For these particles, the swimming motion $(U_{j,swim}, \Omega_{j,swim})$ can be computed by solving the Stokes equation (1) subject to the boundary conditions:

$$u = u_{j,slip} + U_{j,swim} + \Omega_{j,swim} \times (x - c_j) \quad \text{on} \quad \Gamma_j, \forall j, \tag{25a}$$

$$\int_{\Gamma_j} f \, dS = 0, \quad \forall j, \tag{25b}$$

$$\int_{\Gamma_j} (x - c_j) \times f \, dS = 0, \quad \forall j. \tag{25c}$$

When external forces exist or squirmers may collide with each other, this set of boundary conditions Eq. (25) can be superposed to the boundary conditions Eq. (2) and Eq. (3). The swimming velocities $\mathcal{U}_{swim}$ and externally driven velocities $\mathcal{U}_e$ can be computed straightforwardly and independently of each other, without any knowledge about the collisional motion, at the beginning of each time-step. Then, we can apply the collision resolution algorithm described in this section to compute the collision velocities $\mathcal{U}_c$, by simply setting the non-collisional velocities as the sum of swimming velocities and externally driven velocities:

$$\mathcal{U}_{nc} = \mathcal{U}_{swim} + \mathcal{U}_e. \tag{26}$$

4. Boundary integral formulation for the mobility problem

The collision resolution method described above can be applied in conjunction with any mobility problem solvers for arbitrarily-shaped convex bodies. Since one mobility problem needs to be solved at each evaluation $M\gamma$ during the LCP solution, the computational cost and accuracy strongly depend on the mobility solver being used.

In this work, we apply a recently developed 'indirect' BI formulation [70] to solve the mobility problem. While the PDEs Eq. (1) and Eq. (2) can be reformulated as an integral equation in a number of ways using potential theory [84], the advantage of this particular formulation is that second-kind integral equations are obtained without introducing any additional unknowns and constraints such as those in the work by Power and Miranda [69]. The discretized linear system can be solved rapidly with iterative solvers and, in most cases, $10 \sim 30$ iterations are sufficient irrespective of the problem size.

4.1. Boundary integral formulation

The reformulation of Eq. (1) into a set of integral equations relies on the following standard operators:

$$\text{Single Layer Operator:} \quad \mathscr{S}_\Gamma[\rho](x) = \int_\Gamma G(x - y) \rho(y) \, dS_y, \tag{27}$$

$$\text{Double Layer Operator:} \quad \mathscr{D}_\Gamma[\rho](x) = \int_\Gamma T(x - y) n(y) \rho(y) \, dS_y, \tag{28}$$

$$\text{Traction Operator:} \quad \mathscr{K}_\Gamma[\rho](x) = \int_\Gamma T(x - y) n(x) \rho(y) \, dS_y. \tag{29}$$

Here n is the outward surface normal vector to the rigid body boundary Γ, and G and T are the fundamental solutions to the Stokes equations. In particular, G and T are 2rd and 3nd rank tensors known as the Stokeslet and traction kernels, respectively, and are given by

$$G_{ij}(\boldsymbol{r}) = \frac{1}{8\pi}\left(\frac{\delta_{ij}}{|\boldsymbol{r}|} + \frac{r_i r_j}{|\boldsymbol{r}|^3}\right), \tag{30}$$

$$T_{ijk}(\boldsymbol{r}) = -\frac{3}{4\pi}\frac{r_i r_j r_k}{|\boldsymbol{r}|^5}. \tag{31}$$

Note that the viscosity η is set to unity here. For N rigid bodies suspended in free-space with respective boundaries $\Gamma = \{\Gamma_j\}_{j=1}^N$, the indirect BI formulation of [70] sets the fluid velocity $\boldsymbol{u}$ at an arbitrary point $\boldsymbol{x}$ in the fluid domain as

$$\boldsymbol{u}(\boldsymbol{x}) = \mathscr{S}_\Gamma[\rho + \zeta](\boldsymbol{x}), \tag{32}$$

The unknown density functions ρ and ζ are determined as follows. The role of ρ is to match the given force and torque (Eq. (2)d & e) on each body. It has been shown [85, 70] that this can be achieved by setting ρ_j on each rigid body j as:

$$\rho_j(\boldsymbol{x}) = \frac{\boldsymbol{F}_j}{|\Gamma_j|} + \boldsymbol{\tau}_j^{-1}\boldsymbol{T}_j \times (\boldsymbol{x} - \boldsymbol{c}_j), \tag{33}$$

where $\boldsymbol{\tau}_j$ is the moment of inertia tensor [70] and $|\Gamma_j|$ is the surface area defined for particle j with surface Γ_j. For the remaining unknown ζ, an integral equation can be obtained by enforcing that the internal stress generated by $\rho + \zeta$ on each rigid body vanishes. This is accomplished by:

$$\left(\frac{1}{2}\mathscr{I} + \mathscr{K}_\Gamma + \mathscr{L}_\Gamma\right)[\zeta](\boldsymbol{x}) = -\left(\frac{1}{2}\mathscr{I} + \mathscr{K}_\Gamma\right)[\rho](\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \Gamma, \tag{34}$$

where for each Γ_j, $\mathscr{L}_{\Gamma_j}$ is a local operator defined for each particle independently, designed to remove the 6 dimensional null space of the operator $\frac{1}{2}\mathscr{I} + \mathscr{K}_\Gamma$: $\mathscr{L}_{\Gamma_j}[\zeta](\boldsymbol{x}) = \frac{1}{|\Gamma_j|}\int_{\Gamma_j}\zeta_j(\boldsymbol{y})dS_y + \boldsymbol{\tau}_j^{-1}\left(\int_{\Gamma_j}(\boldsymbol{y} - \boldsymbol{c}_j) \times \zeta_j(\boldsymbol{y})dS_y\right) \times (\boldsymbol{x} - \boldsymbol{c}_j)$. We refer to [70] for more details.

In summary, the mobility problem is solved by first computing ρ using given forces and torques, then solving Eq. (34) for ζ, and then computing the fluid velocity $\boldsymbol{u}$ where needed with Eq. (32). The velocities $\boldsymbol{U}_j, \boldsymbol{\Omega}_j$ for each rigid body can be computed by averaging $\boldsymbol{u}$ over the surface Γ_j. In this approach, the operation $\mathscr{U} = \mathscr{M}\mathscr{F}$ reqires an iterative solution to Eq. (34), instead of a simple explicit matrix-vector multiplication.

4.2. Vectorial spherical harmonics discretization

The BI formulation Eq. (32) is applicable to arbitrarily-shaped rigid bodies, but the accuracy relies on the accurate evaluation of the operators $\mathscr{K}$ and $\mathscr{S}$ for the given geometry. For well-separated rigid bodies this is straightforward via various standard surface discretization and smooth quadrature rules. However, for close pairs nearly in contact, special techniques must be employed because the Stokes kernels G and T become *nearly-singular*. While such techniques are well-developed for two-dimensional problems (see [86] and references therein), optimally handling arbitrary geometries in three dimensions is still an open problem and currently an active area of research [87, 88, 89]. For spherical geometries, however, an efficient method based on vectorial spherical harmonic (VSH) basis functions was recently developed in [64] by two of the co-authors. Here, we briefly summarize this VSH technique and apply it to develop a computational framework for spheres.

Any smooth vector field ρ, e.g., the hydrodynamic traction, defined on a spherical surface Γ can be represented as an expansion over the VSH basis functions $\boldsymbol{V}_n^m, \boldsymbol{W}_n^m, \boldsymbol{X}_n^m$:

$$\rho = \sum_{n\geq 0, -n\leq m\leq n}\left[\frac{1}{|\boldsymbol{V}_n^m|^2}\hat{\rho}_{n,m}^V \boldsymbol{V}_n^m + \frac{1}{|\boldsymbol{W}_n^m|^2}\hat{\rho}_{n,m}^W \boldsymbol{W}_n^m + \frac{1}{|\boldsymbol{X}_n^m|^2}\hat{\rho}_{n,m}^X \boldsymbol{X}_n^m\right], \tag{35}$$

where the basis functions V_n^m, W_n^m, X_n^m are generated from the scalar spherical harmonic functions Y_n^m:

$$G_n^m = \nabla_\Gamma Y_n^m, \tag{36}$$

$$V_n^m = G_n^m - (n+1)Y_n^m e_r, \tag{37}$$

$$W_n^m = G_n^m + nY_n^m e_r, \tag{38}$$

$$X_n^m = e_r \times G_n^m, \tag{39}$$

where $\nabla_\Gamma = \dfrac{\partial}{\partial \theta} e_\theta + \dfrac{1}{\sin \theta} \dfrac{\partial}{\partial \phi} e_\phi$ is the surface gradient operator. Note that when $n = 0$, $V_0^0 = -Y_0^0 e_r$ does not vanish, being just a vector field pointing inward on the unit sphere surface.

The coefficients $\hat{\rho}_{n,m}^V, \hat{\rho}_{n,m}^W, \hat{\rho}_{n,m}^X$ are the inner product of ρ and the basis functions on the spherical surface:

$$\hat{\rho}_{n,m}^V = \langle \rho, V_n^m \rangle, \quad \hat{\rho}_{n,m}^W = \langle \rho, W_n^m \rangle, \quad \hat{\rho}_{n,m}^X = \langle \rho, X_n^m \rangle, \tag{40}$$

where

$$\langle u, v \rangle = \int_S u\bar{v} dS. \tag{41}$$

For real-valued ρ, the n, m and $n, -m$ coefficients are complex conjugates to each other. In this work we follow the notational convention for spherical harmonics, and write Y_n^m as

$$Y_n^m(\theta, \phi) = \sqrt{\frac{2n+1}{4\pi}} \sqrt{\frac{(n-m)!}{(n+m)!}} P_n^m(\cos\theta) e^{im\phi}, \tag{42}$$

where

$$P_n^m(x) = \frac{1}{2^n n!} (-1)^m (1-x^2)^{m/2} \frac{\partial^{n+m}}{\partial x^{n+m}} (x^2-1)^n, \tag{43}$$

are Legendre polynomials. The $P_n^{\pm m}$ satisfy:

$$P_n^{-m}(x) = (-1)^m \frac{(n-m)!}{(n+m)!} P_n^m(x). \tag{44}$$

The expansion Eq. (35) is spectrally convergent for a smooth density ρ. Similarly, for a target point not on the spherical surface Γ, the value of the BI operators evaluated at this point $\mathscr{S}_\Gamma[\rho](x)$, $\mathscr{D}_\Gamma[\rho](x)$, and $\mathscr{K}_\Gamma[\rho](x)$ can all be expanded as a series summation over the VSH basis. For example, for a point x outside Γ, we first write $x = (r, \theta, \phi)$ in the spherical coordinate system of Γ. Then, the velocity $u(r, \theta, \phi) = \mathscr{S}_\Gamma[\rho](x)$ and the corresponding fluid pressure $p(r, \theta, \phi)$ governed by Stokes equation can be expressed by:

$$u(r, \theta, \phi) = \sum_{n,m} f_{n,m}^V(r) V_n^m + f_{n,m}^W(r) W_n^m + f_{n,m}^X(r) X_n^m, \tag{45}$$

$$p(r, \theta, \phi) = \sum_{n,m} g_{n,m}(r) Y_n^m. \tag{46}$$

The mapping from the VSH coefficients for ρ in Eq. (35) to $f_{n,m}^V(r), f_{n,m}^W(r), f_{n,m}^X(r), g_{n,m}(r)$ is linear due to the linearity of Stokes equation, and is diagonal:

$$f_{n,m}^V(r) = \frac{n}{(2n+1)(2n+3)} r^{-(n+2)} \frac{1}{\left|V_n^m\right|^2} \hat{\rho}_{n,m}^V + \frac{n+1}{4n+2} \left(r^{-(n+2)} - r^{-n}\right) \frac{1}{\left|W_n^m\right|^2} \hat{\rho}_{n,m}^W, \tag{47a}$$

$$f_{n,m}^W(r) = \frac{n+1}{(2n+1)(2n-1)} r^{-n} \frac{1}{\left|W_n^m\right|^2} \hat{\rho}_{n,m}^W, \tag{47b}$$

$$f_{n,m}^X(r) = \frac{1}{2n+1} r^{-(n+1)} \frac{1}{\left|X_n^m\right|^2} \hat{\rho}_{n,m}^X, \tag{47c}$$

$$g_{n,m}(r) = nr^{-(n+1)} \frac{1}{\left|W_n^m\right|^2} \hat{\rho}_{n,m}^W. \tag{47d}$$

13

Similar diagonalized relations have been derived by Corona and Veerapaneni [64] for both $\mathscr{S}$ and $\mathscr{D}$ operators, and for $\boldsymbol{x}$ both inside and outside Γ.

For the traction operator $\mathscr{K}$, this mapping is no longer diagonal. In other words, the mapping from the VSH coefficients for $\boldsymbol{\rho}$ to the coefficients of $\mathscr{K}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$ is a dense matrix. In this work, we derive a general analytical relation, for the target point $\boldsymbol{x}$ both inside and outside Γ, of this full dense mapping using Eq. (45). The traction $\boldsymbol{t}$ at each target point $\boldsymbol{x}$ is defined as:

$$\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n} = \left(-p\boldsymbol{I} + \nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right) \cdot \boldsymbol{n}, \tag{48}$$

where p has been given in Eq. (45) and (47). The velocity gradient tensor $\nabla \boldsymbol{u}$ has 9 components in the (r, θ, ϕ) spherical coordinate system:

$$\nabla \boldsymbol{u} = \sum_{n,m} \begin{bmatrix} g_{nm}^{rr} & g_{nm}^{r\theta} & g_{nm}^{r\phi} \\ g_{nm}^{\theta r} & g_{nm}^{\theta\theta} & g_{nm}^{\theta\phi} \\ g_{nm}^{\phi r} & g_{nm}^{\phi\theta} & g_{nm}^{\phi\phi} \end{bmatrix}. \tag{49}$$

Each g_{nm} can be analytically computed with the functions $f_{nm}^V, f_{nm}^W, f_{nm}^X$ in Eq. (47). The detailed expressions are given in Appendix A. The velocity gradient $\nabla \boldsymbol{u}$ is then transformed from the spherical coordinate system r, θ, ϕ to the Cartesian coordinate system by standard tensor rotation rules.

For a smooth density $\boldsymbol{\rho}$ defined on a spherical surface Γ, $\mathscr{S}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$, $\mathscr{D}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$, and $\mathscr{K}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$ are again spectrally convergent. This important feature allows us to represent lubrication effects efficiently with only a few spherical harmonic modes. In simulations of many spheres, the operators $\mathscr{S}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$, $\mathscr{D}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$, and $\mathscr{K}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$ are first directly evaluated on discretized spherical harmonics grid points [64] defined on the surface of each sphere by Kernel Independent Fast Multipole Method (KIFMM). However, the results evaluated by KIFMM are inaccurate when spheres are close to each other, because the Stokes kernels G_{ij} and T_{ijk} are singular To overcome this inaccuracy, for each close pair of spheres i, j we first transform the density $\boldsymbol{\rho}$ from the surface grid points to the VSH coefficients $\hat{\rho}_{n,m}^V$, $\hat{\rho}_{n,m}^W$, and $\hat{\rho}_{n,m}^V$. Then, the VSH representation is used to evaluate the accurate values of the boundary integral operators $\mathscr{S}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$, $\mathscr{D}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$, and $\mathscr{K}_\Gamma[\boldsymbol{\rho}](\boldsymbol{x})$. The close pairs are detected by checking the center-to-center distance $\left| \boldsymbol{c}_i - \boldsymbol{c}_j \right|$. If this distance is smaller than a threshold value $\beta(R_i + R_j)$, the VSH representation is used. In this work, we usually choose $\beta \in (1.5, 2)$. We observed no benefits in accuracy for larger β, and the cost of close-pair VSH corrections quickly increases with increasing β.

5. Implementation

Proper implementation is necessary to achieve a scalable computational framework with modern `MPI+OpenMP` parallelism that maximizes the efficiency on high-core-count CPUs. In this section we describe the four major components in our implementation.

5.1. KIFMM

The mobility problem is solved via GMRES iteration of the BI Eq. (34), where the operator $\mathscr{K}$ must be evaluated once every GMRES iteration. The operator $\mathscr{K}$ is an all-to-all operator, where the traction kernel $\boldsymbol{T}$ is evaluated between every pair of the spherical harmonic grid points on all spheres. In total, there are $(p+1)(2p+1)N$ points for N spheres with order p spherical harmonics. The operators $\mathscr{S}$ and $\mathscr{D}$ are also evaluated in this all-to-all style. These are standard operations and can be computed by KIFMM with $O(N)$ cost. In this work, we use the fully parallelized KIFMM package PVFMM [90], and code the kernel functions as optimized `AVX2` intrinsic instructions to fully utilize the 256-bit `SIMD` capability of modern CPUs. The developed code is open-sourced as STKFMM on GitHub.[2] We verified the accuracy of the operator evaluations to machine precision, and benchmarked the scalability to thousands of cores on a CPU cluster interconnected by Intel Omni-Path fabrics.

---

[2] https://github.com/wenyan4work/STKFMM

5.2. Near neighbor detection

Once the BI operators are evaluated with KIFMM, near-field corrections must be performed for close-to-contact pairs of spheres with the VSH representation. This step requires efficient detection of all close-to-contact pairs. This is a standard neighbor detection operation and can be efficiently completed with algorithms such as cell list or k-D tree. However, although there are a few high-performance libraries publicly available such as FDPS [91], DataTransferKit [92], and LibGeoDecomp [93], there are some important features still missing. For example, the necessary VSH data, including grid point coordinates, values, and expansion coefficients, needs to be efficiently migrated between the `MPI` ranks, and customizable serialization and de-serialization are necessary to allow VSH data with different order p to be sent and received as `MPI` messages.

Therefore we built a custom near neighbor detection module based on a Morton-coded octree. This near neighbor module is fully parallelized with both `OpenMP` and `MPI`. Once the near neighbor pairs are detected, the necessary data is transferred in `msgpack` binary format with the `msgpack-c` library.[3] Each pair is dispatched to one `OpenMP` thread to compute the near corrections, implemented in a thread-safe way. This near neighbor detection module is also used to identify the possibly colliding pairs to construct the set for all constraints $\mathscr{A}$. Then the sparse matrix $\mathscr{D}$ is constructed for the LCP (20), distributed on all `MPI` ranks.

5.3. Load-balancing

Proper load balancing is necessary to ensure scalability over `MPI` parallelism, and the balancing of both the LCP and the mobility problem must be considered. The balancing of the mobility problem Eq. (34) is handled through the KIFMM routine and the near neighbor detection routine. An adaptive octree with 2:1 balancing is built inside the PVFMM library and decomposed to each `MPI` rank to allow efficient KIFMM evaluation. Similar decomposition is also used in the near neighbor detection module so that each `MPI` rank handles roughly the same number of near neighbors pairs to perform VSH corrections. The decomposition for the LCP is slightly different. In LCP (20), each contact pair appears only once in the vector $\boldsymbol{\gamma}$ and the geometric matrix $\mathscr{D}$. We use a simple but effective strategy to partition $\boldsymbol{\gamma}$ and $\mathscr{D}$. Each particle is labeled with an index i, which is globally unique across all `MPI` ranks and randomly initialized in the beginning of simulations, so that i is uncorrelated with its spatial location $\boldsymbol{c}_i$. When the pair of particles i, j is detected to be possibly colliding, the minimum separation Φ_{ij}, the sparse colliding geometry column vector $\boldsymbol{D}_{ij}$, and the unknown collision force γ_{ij}, are determined to be owned by the `MPI` rank which owns the smaller index between i, j. The transposed matrix $\mathscr{D}^T$ is then constructed in compressed-row-storage (CRS) format where the rows are partitioned to each `MPI` rank. $\boldsymbol{\Phi}$ and $\boldsymbol{\gamma}$ are partitioned in the same way as $\mathscr{D}^T$. Compared to the matrix-free implementation where the matrix $\mathscr{D}^T$ and $\mathscr{D}$ are not explicitly constructed [47], our implementation takes more storage space but allows the utilization of the highly optimized sparse linear algebra functions implemented in the libraries `Tpetra` and `Kokkos` in `Trilinos`.[4]

5.4. Vector spherical harmonics

For spherical harmonics of order p, we compute the VSH expansion coefficients from function values on the $(p+1)(2p+1)$ spherical harmonic grid points using a spherical harmonic transform (SHT). We do this by computing $2p+1$ discrete Fourier transforms (DFT) in the ϕ direction and a Legendre transform in the θ direction for each sphere. The DFT is computed using the FFTW3 functions [94] and requires $O\left(p^2 \log p\right)$ work per sphere. We compute the Legendre transform using matrix-vector products, and this requires $O\left(p^3\right)$ work per sphere. We do not use a fast Legendre transform (FLT) since it is advantageous only for very large p. For multiple spheres, we parallelize using OpenMP by partitioning the spheres across threads. When the number of spheres is greater than the number of threads, we use blocking to compute multiple transforms together. This allows us to use matrix-matrix products for the Legendre transform, and these are computed efficiently using an optimized BLAS[5] implementation. We use Intel Math Kernel Library for both the FFTW3 and BLAS functions. We can similarly compute function values on the spherical harmonic grid from the spherical harmonic coefficients using the inverse SHT. For vector valued functions, we can compute the coefficients for the representation in the

---

[3]https://github.com/msgpack/msgpack-c
[4]https://trilinos.org
[5]http://www.netlib.org/blas

VSH basis by computing three scalar SHTs for each sphere as discussed in [64]. Similarly, we can compute the grid values from the VSH coefficients using three inverse SHTs.

Once the VSH coefficients for the density function ρ have been computed for each sphere, we can then compute the coefficients for the velocity $\boldsymbol{u}$, the pressure p and the traction $\boldsymbol{t}$ for each sphere-target pair using the diagonal operators discussed in Section 4.2. Finally, we evaluate the VSH expansions to get the potentials. This requires evaluating the VSH basis functions and has $O(p^2)$ computational cost for each sphere-target pair. Since the positions of the spheres only change between time-steps, we could precompute these basis functions at each time step and reuse them during the linear solve. However, this would require $O(p^2)$ memory for each sphere and near target pair; therefore, we do not do this precomputation in our current implementation.

6. Results

In this section we report the numerical accuracy and performance results for suspensions of spherical particles in unbounded Stokes flow.

6.1. Static lubrication benchmark

In this section we probe the accuracy of the mobility solver discussed in Section §4, for a few static configurations. The collision resolution algorithm is not used.

6.1.1. Two spheres

We begin with a convergence test for a static configuration where lubrication forces are important. A pair of nearby spheres with radius a are driven by the same force $\boldsymbol{F}_g$, i.e. sedimentation, or torque $\boldsymbol{T}$ as illustrated by the schematics in Figs. 2 and 3, respectively. To evaluate convergence, U and $\boldsymbol{\Omega}$ are solved for both spheres, for each case, at various gap separations ϵ. The errors are shown in Fig. 2 for the sedimentation case and in Fig. 3 for the rotation case. In each figure, the left panel shows the convergence error (absolute value) relative to the results generated by $p = 24$: $\left|U_{g,error}(p)\right| = \left|U_g(p) - U_g(24)\right|$ and $|\Omega_{error}| = |\Omega(p) - \Omega(24)|$, of the particle on the left.

It is also well-known that spherical harmonic grids are significantly denser around the poles in comparison to areas close to the equator. This non-uniformity across the sphere surface may affect the capability to resolve the highly non-uniform distribution of hydrodynamic force $\boldsymbol{f}$ induced by lubrication effects. Because of this, the spherical harmonics grid is randomly oriented for each sphere at different ϵ. This randomness induces some asymmetry error in the computed U and $\boldsymbol{\Omega}$, defined as the difference of computed velocity for the two particles $\Delta U_g = U_{g,1} - U_{g,2}$ and $\Delta\Omega = \Omega_1 - \Omega_2$, as shown in the right panels of Figs. 2 and 3.

The results show that the asymmetry error is always on the same order as the convergence error. Therefore, the user does not need to pick a particular pole orientation to 'better resolve' the lubrication effect. Further, spectral convergence holds until the separation ϵ/a is comparable with $1/p$. This is because for such small gaps the hydrodynamic traction has a large peak in the near-contact region due to lubrication effects. Also, the error for the rotation case is larger than for the sedimentation case, because when the two spheres are rotating in the same direction, the fluid in the near-contact region has a very large shear rate due to the relative motion of the two sphere surfaces.

6.1.2. Three spheres

Accurate benchmark data for lubrication effects is surprisingly hard to find because the dominating method in this setting was asymptotic expansions [18, 26] with questionable convergence beyond 3 digits of accuracy [95]. To our knowledge the most accurate lubrication benchmark is given by Wilson [95] to 10 digits of accuracy for a few particular geometries of several spheres. We use these results to evaluate the accuracy of our mobility solver based on KIFMM and VSH for three equidistant spheres of equal radii each driven by a constant force $\boldsymbol{F}$ perpendicular to their common plane. The results for the relative error of the translational velocity U and angular velocity $\boldsymbol{\Omega}$ are shown in Fig. 4 as a function of the gap separation distance ϵ between each pair of spheres.

The results in Fig. 4 show accuracy similar to the two-particle benchmarks shown previously in Fig. 2 and Fig. 3. It is clear that the spectral convergence with increasing p is kept until ϵ/a is comparable to $1/p$, which is roughly at $\epsilon/a \approx 0.05$ for $p = 24$. When ϵ further decreases, the error decreases slowly with increasing p.
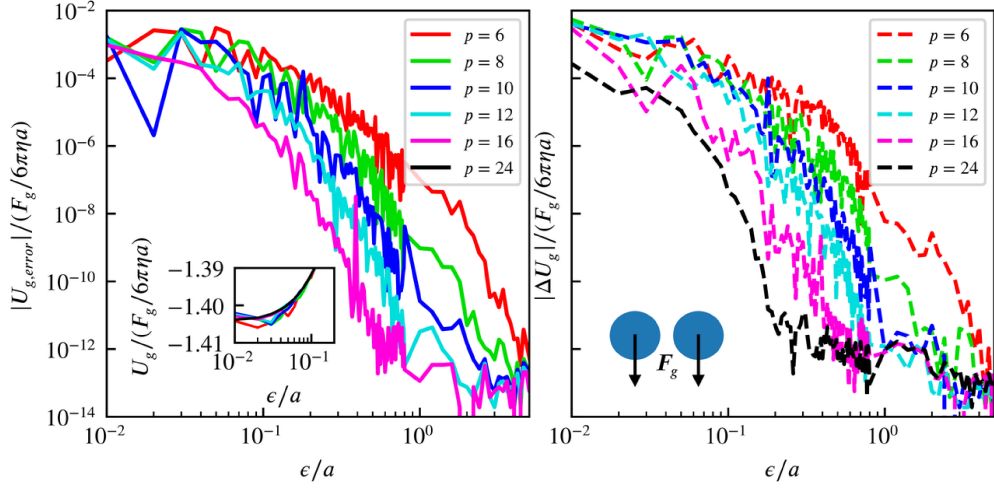
Figure 2: The convergence error and asymmetry error for the sedimentation velocity U_g. The inset plot shows the sedimentation velocity in comparison to the single sphere Stokes solution $F_g/(6\pi\eta a)$.
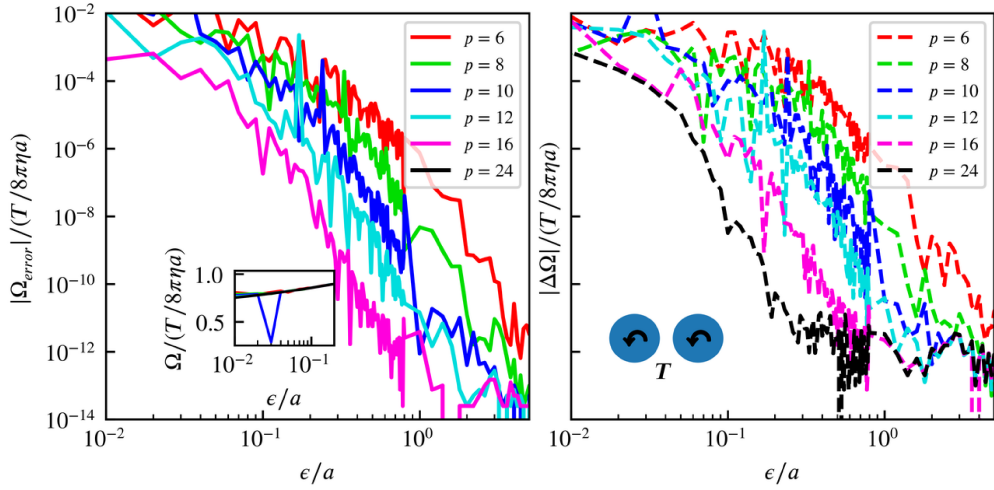

Figure 3: The convergence error and asymmetry error for the angular velocity Ω. The inset plot shows the angular velocity in comparison to the single sphere Stokes solution $\Omega = T/(8\pi\eta a^3)$.
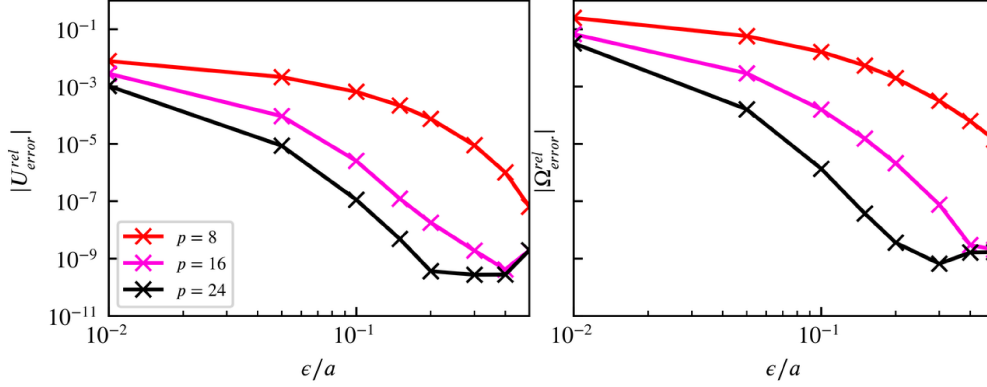
Figure 4: The relative error of translational and angular velocity in comparison to high accuracy reference data [95]. The error is limited to 10^{-10} as this reference data has 10 digits of accuracy.

Lubrication effects are added explicitly in the popular Stokesian Dynamics method [18, 21], where the asymptotic analytical functions for particle hydrodynamic traction multipoles are tabulated and added to a far-field expansion of the mobility matrix. The analytic functions for multipoles are truncated at the stresslet level. Therefore even at $\epsilon/a = 0.1$, at most two digits of accuracy is achieved [95]. Even with a low order grid $p = 8$, the VSH approach has accuracy on par with Stokesian Dynamics.

For dynamic simulations with many possible collisions, to reduce computation cost we generally use a low order grid with $p = 6, 8$, etc., because when particles are close to contact the benefits of increasing p are not significant enough to justify the extra cost. In particular, we set the collision radius a_c to be slightly larger than the particle radius a, and resolve collisions for each particle at the collision radius. Usually $a_c/a = 1.05$. This is a common strategy [19, 96, 97] in particle-tracking simulations in suspension mechanics, where a soft repulsive pairwise potential which acts at the length-scale a_c is used to prevent particle overlaps. For example, in Stokesian Dynamics an exponentially decaying pairwise repulsive force is usually added over the length-scale $\epsilon \approx (0.01 \sim 0.05)a$ to prevent the spheres from overlapping [19, 96]. In this work, we actually choose a_c based on the choice of p, for example, $a_c = 1.01a$ when $p = 24$ and $a_c = 1.05a$ when $p = 6$ or 8.

6.2. Two spheres approaching each other

In this section we probe the behavior of the mobility solver and the LCP solver with a dynamic problem, where two spheres of equal radius a are dragged by constant equal and opposite external forces $\pm \boldsymbol{F}$ along the horizontal $\boldsymbol{x}$-axis, with the spheres starting with a vertical offset a from each other. The two spheres then approach and roll over each other. The collision radius is set to be $a_c = 1.01a$, i.e., the collision force is non-zero when the separation $\epsilon/a = 0.02$.

The orientation of the spherical harmonics grid with $p = 24$ is randomly chosen for each sphere, and different random orientations are used for computing $\mathscr{U}_{nc}$ and $\mathscr{U}_c$. The time-step $\Delta t = 0.1\eta a^2/F$ is fixed for the results reported in Fig. 5, and the simulation remains stable if Δt is increased by a factor of 10. The residual tolerance of BBPGD is set to 10^{-5}. The simulation shown in Fig. 5 includes 4 stages, as illustrated by the snapshots (A), (B), (C), and (D). In each snapshot, the left panel shows the hydrodynamic traction induced by the external force $\boldsymbol{F}$, with the grey arrows showing the total velocity $\boldsymbol{U}_{nc} + \boldsymbol{U}_c$ for each sphere. The right panel shows the hydrodynamic traction induced by collision forces. The performance of BBPGD and APGD solvers for the LCP are compared in Fig. 6.

In Fig. 5(A), the particles are close enough and are determined to be possibly in contact. The LCP (20) is constructed as a scalar problem, because there is only one possible contact. The LCP solver then determines that no actual collision happens, as shown by the zero hydrodynamic traction in the right panel of Fig. 5(A). In Fig. 5(B), the LCP is constructed similarly, but the LCP solver finds that the collision force is non-zero. This is shown by the non-zero hydrodynamic traction induced by collision in Fig. 5(B). In Fig. 5(C), the particles are

18

about to roll over each other so the collision force is tiny. In Fig. 5(D), the collision force is zero because the spheres are instantaneously touching each other and are moving towards a collision-free configuration.
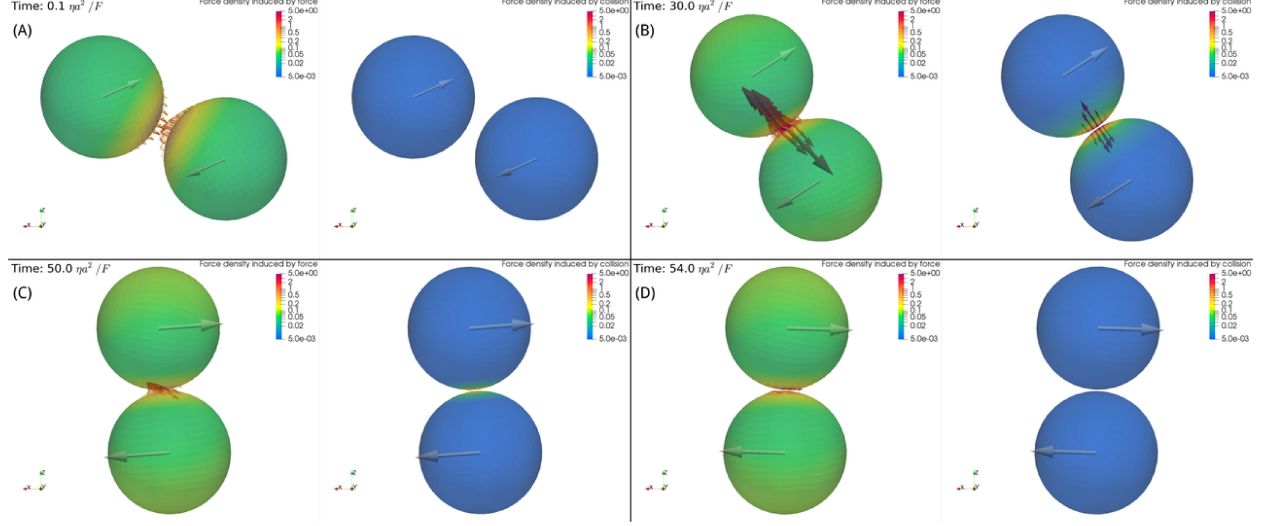

Figure 5: A pair of particles dragged by equal and opposite external forces to roll over each other. In (A), (B), (C), and (D), the left panel shows the hydrodynamic traction induced by the external force $\boldsymbol{F}$ with colored surface and vectors on the spherical harmonic grid. The grey arrows show the total velocity $\boldsymbol{U}_{nc} + \boldsymbol{U}_c$ of each particle. The right panels show the hydrodynamic traction induced by the collision force.

The performance history of BBPGD and APGD, shown in Fig. 6, shows the difference between the four stages corresponding to the snapshots. The 'steps' count shows how many GD steps are used in the solver, and the 'MVOPs' number shows how many evaluations of $\boldsymbol{M}\boldsymbol{\gamma}$ are invoked during the GD steps, for each time-step. The actual cost (and running time) scales with the number of 'MVOPs' because one mobility problem is solved within each evaluation of $\boldsymbol{M}\boldsymbol{\gamma}$. For stages (A) and (D), only 1 MVOP is necessary for BBPGD because the solver finds that the zero initial guess of collision force already solves the problem, without the necessity of computing GD steps. In stage (B) and (C0), BBPGD is able to converge with only 1 GD step for this scalar LCP. Recall in Algorithm 1 that the first BBPGD step takes 2 MVOPs, and then only 1 MVOP per step. It is clear that in all cases BBPGD has much lower cost compared to APGD. We observed similar performance advantages of BBPGD in many-body simulations, and all the rest results reported in this paper are computed with BBPGD only.
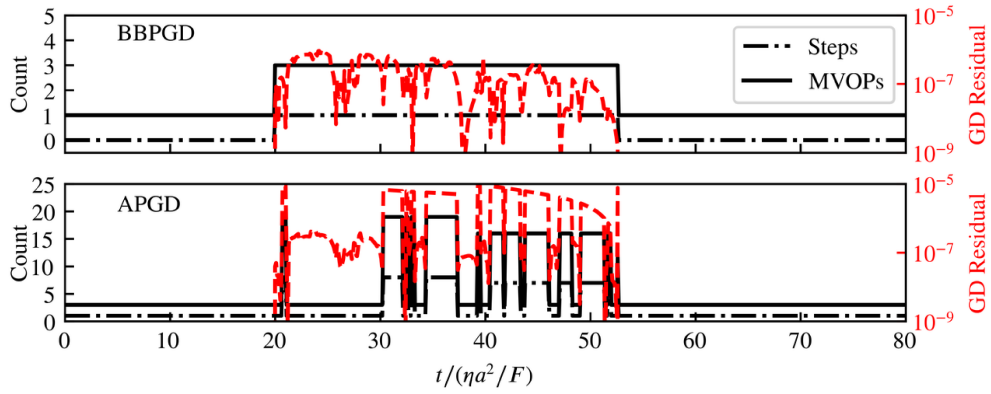
Figure 6: The performance of BBPGD and APGD solvers at each time-step for the simulation shown in Fig. 5. The residual is set to 10^{-5} for both cases. BBPGD significantly outperforms APGD.

6.3. Sedimentation of a dense cluster

The sedimentation of 1000 monodisperse spheres is simulated as a more demanding and realistic testing problem. The spheres are initially placed in a dense spherical cluster of volume fraction $\approx 20\%$, and the dynamics are tracked in an unbounded fluid domain. Each sphere is of the same radius a and sediments due to the same gravitational force $\boldsymbol{F}$ pointing towards $-z$ direction. The collision radius for each particle is set to 5% larger than a. The time-step is set to $\Delta t = 0.05\eta a^2/F$. $p = 6$ is used for the spherical harmonic representation, and $m = 8$ (equivalent point density along each octree box edge) is used for KIFMM evaluations.

The snapshots of this sedimentation process are shown in Fig. 7, where for snapshots (A), (B) and (C) the four panels show the hydrodynamic traction induced by gravitational force, the hydrodynamic traction induced by collision forces in the cluster, the collision force network corresponding to the set of constraints $\mathscr{A}$, and the velocity of each particle. In general, particles in a crowded environment sediment faster than isolated particles, because their close neighbors drag fluid together with them, effectively reducing the relative friction between particle and fluid. Therefore, the particles close to the cluster surface sediment much slower than the particles close to the cluster core. These slower particles tend to lag behind and accumulate at the trailing point of the cluster. At that point, these slow particles form a dilute structure locally, and become even slower as shown in snapshots (B) and (C) of Fig. 7. Eventually, a tail of slow particles form behind the sedimenting cluster, as illustrated in snapshot (D) of Fig. 7 showing the structure at the end of the simulation $t = 200\eta a^2/F$, where each particle is colored by the hydrodynamic traction induced by collisions. This is a well-known phenomena [98], and has also been observed in sedimenting clouds of fibers [99].

Because of this phenomena, the cluster of spheres becomes less dense during sedimentation. Consequently, the number of BBPGD steps necessary to resolve collisions gradually decreases. The performance of the BBPGD solver is shown in Fig. 8, where the convergence tolerance of BBPGD is set to $\epsilon_{tol} = 10^{-4}$.

6.4. Assessing the effect of numerical parameters

In our computational framework there are several tunable parameters, including the order p of spherical harmonic representation, the time-step Δt, the BBPGD solver tolerance ϵ_{tol}, and the collision radius a_c. In this section we report the effect of these four parameters on the accuracy of the solver by repeating the sedimentation cluster simulation of 1000 spheres done in the previous subsection but for a short amount of time $t = 10\eta a^2/F$. The simulation error is measured by the relative L_2 error of the vector $\mathscr{U} \in R^{6000}$ at the end of the simulation, relative to a more accurate reference $\mathscr{U}_{ref}$:

$$\epsilon_2 = \frac{\left\| \mathscr{U} - \mathscr{U}_{ref} \right\|_2}{\left\| \mathscr{U}_{ref} \right\|_2}. \tag{50}$$

We fix $m = 10$ (equivalent point density along each octree box edge) for KIFMM evaluations, and the GMRES convergence residual to 10^{-6}.

We first vary Δt over the range $(0.02-1.0)\eta a^2/F$, while fixing $p = 12$, $a_c = 1.01a$, and $\epsilon_{tol} = 10^{-5}$. The time-stepping as implemented in Eq. (16) has first order accuracy, and the result shown in Table 1 shows a consistent behavior at the smaller time-steps. We note that the largest time-step, $\Delta t = 1.0\eta a^2/F$ is 20 times larger than the time-step used in the previous section, and for this large time-step each particle may move $2 \sim 3$ times of particle radius during each time-step. Nonetheless, this simulation remains stable and still reasonably accurate in its average error.

$\Delta t/(\eta a^2/F)$	1.0	0.2	0.1	0.05
ϵ_2	0.00744	0.00198	0.000931	0.000372

Table 1: The effect of varying the time-step Δt upon the relative error ϵ_2. The reference case uses $\Delta t = 0.02\eta a^2/F$, with $p = 12$, $a_c = 1.01a$, and $\epsilon_{tol} = 10^{-5}$.

We next study the effect of varying the BBPGD tolerance ϵ_{tol} between 10^{-3} to 10^{-6}, while fixing other parameters to $p = 12$, $a_c = 1.01a$, and $\Delta t = 0.1\eta a^2/F$. The results, reported in Table 2 , show that tuning ϵ_{tol}
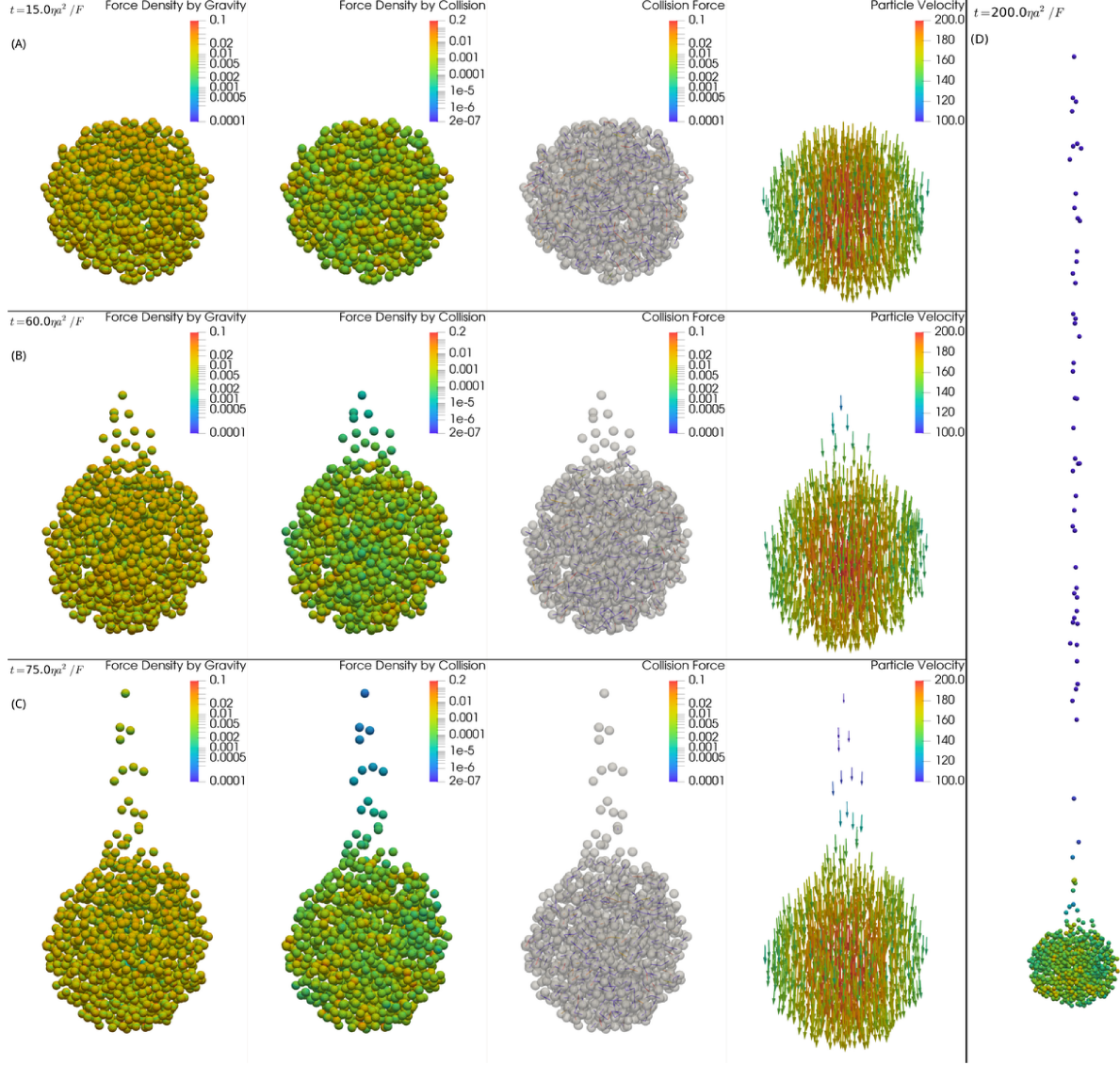
Figure 7: The sedimentation of a dense cluster of 1000 monodisperse spheres with volume fraction $\approx$ 20%. In the snapshots (A), (B) and (C), the four panels show the hydrodynamic traction induced by the gravitational force $\boldsymbol{F}$ pointing toward $-z$ direction, the hydrodynamic traction induced by collision forces in the cluster, the collision force network corresponding to the set of constraints $\mathscr{A}$, and the velocity of each particle. The snapshot (D) shows the structure at the end of the simulation, at $t = 200\eta a^2/F$, where each particle is colored by the hydrodynamic traction induced by collisions. A video of this sedimentation process is available in the Supplemental Material.
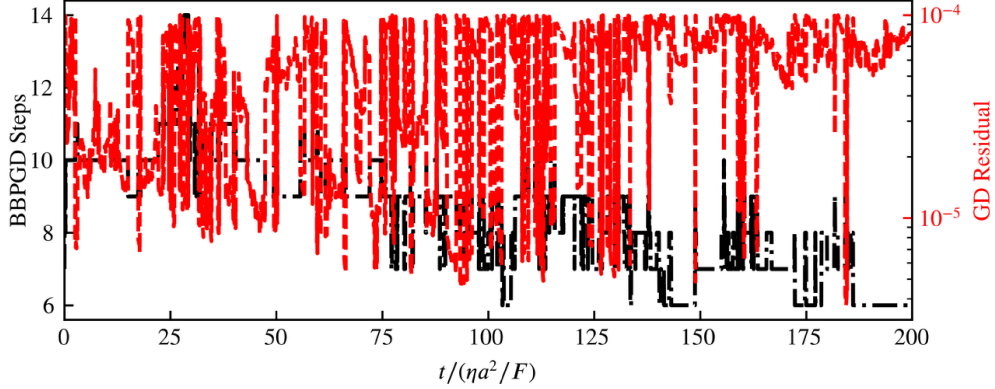
Figure 8: The performance of the BBPGD solver for the sedimentation simulation shown in Fig. 7. The simulation starts from a collision-free configuration. The time-step is $\Delta t = 0.05\eta a^2/F$. The convergence tolerance of BBPGD is set to $\epsilon_{tol} = 10^{-4}$.

has little effect on the simulation accuracy. This is because the most difficult part of the BBPGD solver is the identification of the active set $\mathscr{A}_c$, i.e., determining where are the actual collisions. Once this set is identified, the BBPGD solver converges quickly, as demonstrated by the average number of BBPGD steps in Table 2, where decreasing ϵ_{tol} by an order of magnitude only requires 2 or 3 more BBPGD steps. Dai and Fletcher [76] have thoroughly analyzed this behavior of BBPGD algorithm.

ϵ_{tol}	10^{-3}	10^{-4}	10^{-5}
ϵ_2	2.13×10^{-7}	3.52×10^{-8}	0
Average BBPGD Steps	9	12	14

Table 2: The effect of varying the tolerance ϵ_{tol} of BBPGD on the relative error ϵ_2. The reference case uses $\epsilon_{tol} = 10^{-6}$. Other parameters are fixed to $p = 12$, $a_c = 1.01a$, and $\Delta t = 0.1\eta a^2/F$. $\epsilon_2 = 0$ means the error has fallen below machine precision.

The effect of changing the order of the spherical harmonics expansion is shown in Table 3 , where p is varied from 8 to 20, with $p = 20$ used as the reference case, and with other parameters held fixed at $a_c = 1.01a$, $\epsilon_{tol} = 10^{-5}$, and $\Delta t = 0.1\eta a^2/F$. The error ϵ_2 shows slow convergence with increasing p. This is because particles are very close to each other in this dense cluster simulation, and as we have shown in Fig. 4, the error of the mobility solver decreases slowly in this regime. This is the intrinsic difficulty of resolving lubrication effects efficiently in dynamic simulations, and remains an open problem. The reference data used in Fig. 4 is computed by a highly accurate method which is not generally feasible in dynamic simulations. Because of the cost in increasing the order of the spherical harmonics expansion, we do not use high order expansions in the large scale simulations of the next section .

p	8	12	16
ϵ_2	0.00697	0.00268	0.00114

Table 3: The effect of varying p, the order of the spherical harmonics expansion, upon the relative error ϵ_2. The reference case uses $p = 20$, and other parameters are fixed to $a_c = 1.01a$, $\epsilon_{tol} = 10^{-5}$, $\Delta t = 0.1\eta a^2/F$.

Finally, we investigate the effect of the collision radius a_c. Table 4 reports the relative error ϵ_2 for a_c varying from $1.001a$ to $1.05a$, and fixing $p = 12$, $\epsilon_{tol} = 10^{-5}$, and $\Delta t = 0.1\eta a^2/F$. ϵ_2 decreases monotonically with decreasing a_c, but the convergence is slow. This is again related to the difficulty in resolving lubrication effects when particles are close to each other, because changing a_c only affects those particles that may get closer. In this regime the accuracy of spherical harmonics expansion is only modestly accurate, even with a high order

22

expansion. Therefore, there is generally no need to pick a very small value of a_c, and we will use fairly large values of a_c in the next section for large scale simulations.

$a_c/a - 1$	0.05	0.02	0.01	0.005
ϵ_2	0.0108	0.00656	0.00521	0.00352

Table 4: The effect of varying the collision radius a_c upon the relative error ϵ_2. The reference case uses $a_c = 1.001a$, with other parameters fixed to $p = 12$, $\epsilon_{tol} = 10^{-5}$, and $\Delta t = 0.1\eta a^2/F$.

6.5. Scaling benchmark

The sedimentation of particles randomly placed in a cubic box in an unbounded fluid is also used to benchmark the scaling of our implementation, using up to 80000 spheres on 1792 cores (64 nodes). Weak and strong scaling is measured on a cluster where each node is equipped with two 14-core Intel Xeon CPU E5-2680 v4 running at 2.40GHz. Hyper-Threading is turned off. One `MPI` rank is launched for each CPU socket, and every `MPI` rank launches 14 `OpenMP` threads. In total, 28 cores are used on each node, and the nodes are interconnected by a 100 Gb/s Intel Omni-Path (OPA) network. Intel MPI compilers and libraries are used, together with Intel MKL libraries for BLAS, LAPACK and FFTW3 functions. The radius a of each particle is randomly generated from a log-normal distribution with standard parameters $\mu = a$ and $\sigma = 0.3a$. The collision radius $a_c = 1.05a$ is taken for each sphere. The time-step is fixed at $\Delta t = 0.1\eta a^2/F$, where $\boldsymbol{F}$ is the sedimentation force applied on each particle. $p = 8$ is used for the spherical harmonic grid and $m = 10$ (equivalent point density along each octree box edge) is used for the KIFMM evaluations, throughout these benchmarks. The largest system of 8×10^4 spheres has 486 degrees of freedom for hydrodynamic traction on each sphere and $\sim 3.9 \times 10^7$ degrees of freedom in total.

The running time for the five major operations are measured and reported in Fig. 9, 10, and 11. 'Far Setup' refers to the time to compute the coordinates of spherical harmonic grid points and to build an adaptive octree for KIFMM evaluations. 'Far Traction' refers to the evaluation of the traction operator $\mathcal{K}$ in the BI equation Eq. (34). 'Far SL' refers to the evaluation of the single layer operator $\mathcal{S}$ to evaluate the fluid velocity $\boldsymbol{u}$ on grid points after the traction ρ is found by solving Eq. (34). 'Near Setup' refers to the time to perform near neighbor detection to prepare the VSH evaluations. 'Near Correction' refers to the time for VSH corrections. The cost of constructing the LCP sparse matrix $\mathcal{D}$ and applying the sparse matrix-vector multiplications during LCP solution is not shown here because it is negligible compared to the total cost of mobility solutions.

Fig. 9 shows the strong scaling for 10,000 spheres on up to 224 cores (8 nodes) and Fig. 10 shows the strong scaling for 80,000 spheres on up to 1792 cores (64 nodes). In both cases the same initial configuration at approximately 6% volume fraction is used throughout all runs on different numbers of cores. The left panels of Fig. 9 and 10 show the total wall time for different operations for 11 time-steps. The right panels show the average wall time for performing each operation once. The black dashed line denotes the ideal scaling with 100% parallel efficiency for reference. In these scaling tests, spherical harmonic grids with the same order p but different orientation are used for mobility and collision problems. Therefore, at each time-step 'Far Setup' and 'Near Setup' are both executed twice, i.e., once for the mobility problem and once for the collision problem. The 'Far Traction' is evaluated once at each GMRES iteration of solving the mobility problem Eq. (34), but the 'Far SL' is evaluated only when GMRES converges. In general, 'Far Setup' and 'Far SL' are executed the same number of times.

6.5.1. Strong scaling

The parallel efficiency of the 'Near Correction' part is close to ideal except for the largest case on 1792 cores, shown in Fig. 10, where we suspect there is some load-unbalancing due to domain decomposition. The efficiency of 'Far Traction' and 'Far SL' are a bit lower than the near correction part, but they are consistent with the results demonstrated for the library PVFMM [90], which we used to perform KIFMM evaluations. The efficiency of 'Far Setup' is even lower, but since it runs only once per time-step and takes only a small fraction of the total running time, there is almost no noticeable benefit in optimizing this part. The wall time for 'Near Setup' is negligible and invisible in both Fig. 9 and Fig. 11.
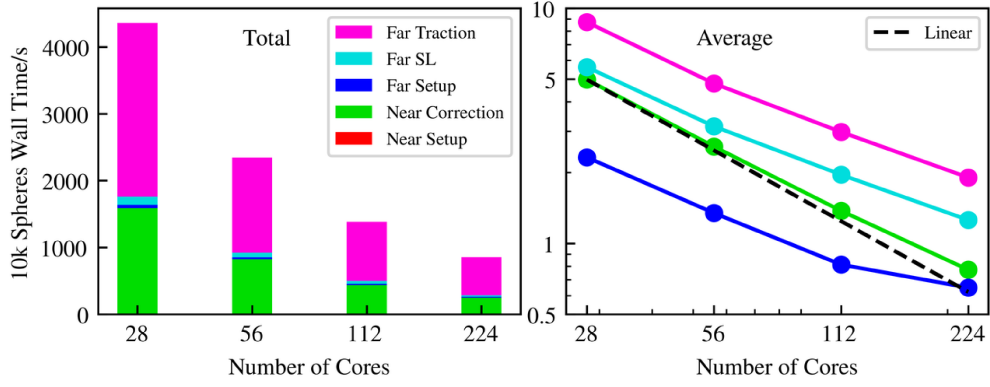
23

Figure 9: The strong scaling of $10,000$ spheres for 11 time-steps on up to 224 cores. The left panel shows the total time for each operation. 'Far Setup' and 'Near Setup' both run 22 times. 'Far SL' runs 21 times and 'Far Traction' runs 298 times. 'Near Correction' runs $21 + 298 = 319$ times for both SL and Traction corrections. The right panel shows the average time for each operation.
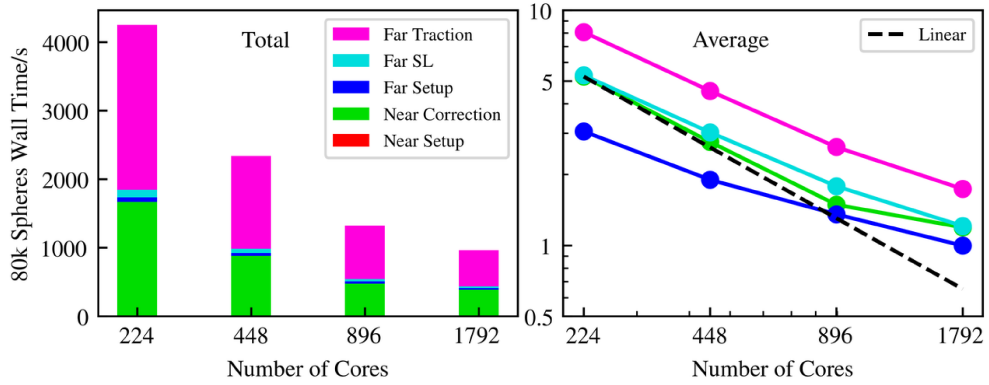

Figure 10: The strong scaling of 8×10^4 spheres for 11 time-steps on up to 1792 cores. The left panel shows the total time for each operation. 'Far Setup' and 'Near Setup' both run 22 times. 'Far SL' runs 21 times and 'Far Traction' runs 298 times. 'Near Correction' runs $21 + 298 = 319$ times for both SL and Traction corrections. The number of times each operation is executed coincides with the previous smaller scale test of $10,000$ spheres because the configuration is generated at similar radius distribution and volume fraction. The right panel shows the average time for each operation.

6.5.2. Weak scaling

Since the cost of performing near corrections grows with the number of near pairs, we keep the system volume fraction approximately constant by adjusting the box size according to the number of particles. Fig. 11 shows the average time for performing each operation once during a 10-time-step simulation, for two volume fractions 3% and 6%. Ideally a flat line is expected if the parallel efficiency is 100%. Here the running time of 'Far Setup' grows faster but other operations are not significantly far from the ideal case. This non-ideal scaling of 'Far Setup' does not matter in real simulations because this operation is performed only once when particles move, that is, once for each time-step. Therefore the net cost of this 'Far Setup' is far less than the total cost in other operations, as can be seen from the left panels in Fig. 9 and Fig. 10.
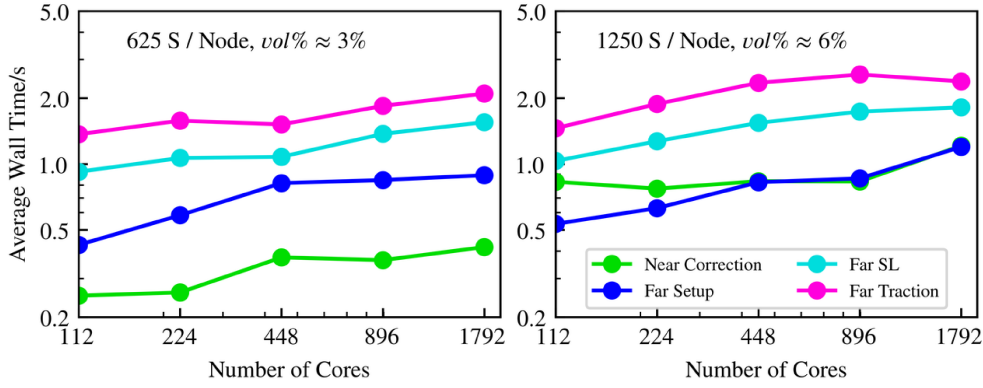
Figure 11: The weak scaling at different volume fractions. The time for performing each operation once is measured. The left panel shows the scaling with 625 spheres per node, ranging from 2500 spheres (4 nodes) to 4×10^4 spheres (64 nodes). The right panel shows the scaling with spheres per node, ranging from 5000 spheres (4 nodes) to 80,000 spheres (64 nodes).

7. An active-matter case study: Suspensions of Stokes rotors

Recently, suspensions of active particles have been intensely investigated as real-world realizations of active matter [5]. Active matter refers to multiscale materials whose microscopic constituents perform directed work on the system, leading to large-scale dynamics such as self-organization. A canonical example of an active suspension is a bacterial bath, wherein microorganisms interact through the flow-fields created by their self-propulsion [100]. These systems can evince instabilities towards alignment and unpredictable large-scale flows sometimes termed "bacterial turbulence" [101]. Similarly complex dynamics appears in suspensions of microtubules which are "polarity sorted" by the directed motion of cross-linking motor-proteins [102, 7, 8].

A very different kind of active suspension consists of immersed particles that are driven to rotate, say rather than swim or sort, with that rotation again creating flow fields that can create large-scale coupling and dynamics. Such rotor systems are typically – but not always [103, 66] – driven by external means, such as a rotating magnetic field. Such systems can show activity-induced phase separation [65], crystallization [66], odd surface dynamics and rheology [67], and forms of active turbulence [68].

Here we use the methods developed here to study the dynamics of closely packed rotor systems, showing the development of large-scale dynamics. In each example we assume the same external torque $\boldsymbol{T}$ is applied to every particle. Within each time-step, we first solve a mobility problem to compute the velocity $\mathscr{U}_{nc}$ driven by $\boldsymbol{T}$. This is followed by applying the collision resolution method described in Section §3 to compute the collision velocity $\mathscr{U}_c$. A fixed time-step Δt is used throughout, and taken large enough so that each particle may move about 20% of its radius at each time-step. This time-step is roughly two to three orders of magnitude larger than the usual choices in Stokes suspension simulations [9, 21]. We do this to demonstrate the stability of our collision resolution algorithm. For all simulations in this section, $p = 6$ is used for the spherical harmonic grid and $m = 8$ (equivalent point density along each octree box edge) is used for KIFMM evaluations.

25

7.1. A cluster of 10,000 *rotors*

We first probe the dynamics of a spherical cluster of 10,000 polydisperse (in diameter) spherical rotors in an unbounded fluid. A constant torque T along the z-axis is applied to each sphere. The radius of each particle is randomly generated from a log-normal distribution with standard parameters $\mu = a$ and $\sigma = 0.3a$, where the probability density function is defined as $p(x) = \frac{1}{x\sigma\sqrt{2\pi}}\exp\left(-\frac{(\ln x - \mu)^2}{2\sigma^2}\right)$. The cluster is approximately spherical and the volume fraction of spheres is approximately 10%. A fixed time-step, $\Delta t = 1.0\eta a^3/T$, is used. The collision radius a_c is set to 10% larger than the radius for each particle.

In the absence of hydrodynamic interactions, each particle would rotate at constant rate about its $z-$axis, with larger particles rotating more slowly than smaller ones (given the constant driving torque). Figure 12, at $t = 300\eta a^3/T$, shows the effect of hydrodynamic and steric coupling in creating large-scale rotation of the cluster. The left panel shows the hydrodynamic traction magnitude across each rotor surface. The middle panel shows the instantaneous velocity magnitude of each particle, where blue ones are slow and green ones are fast. The right panel shows the trajectory of each particle, starting from $t = 0$, and colored by each particle's velocity magnitude at each time-step. The trajectories show that this cluster is rotating relative to a common axis in the z direction through the cluster center. The simulation runs till $t = 500\eta a^3/T$, and no visible expansion or shrinking of this cluster is observed.
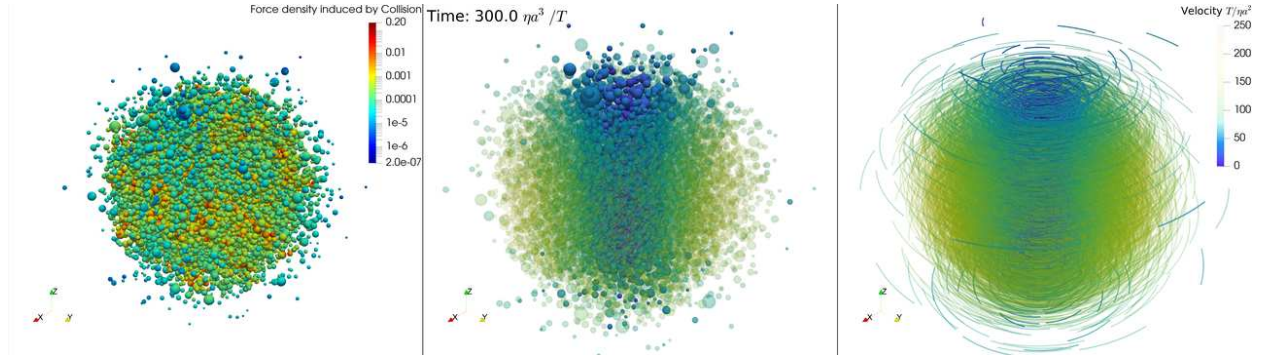
Figure 12: Snapshots at $t = 300\eta a^3/T$ of a cluster driven by a fixed torque T on each sphere. The left panel shows the hydrodynamic traction magnitude across each rotor surface. The middle and the right panels are both colored by the instantaneous center velocity magnitude of each sphere at $t = 300\eta a^3/T$, showing the particle structure at this time-step and the particle trajectories starting from the initial configuration, respectively. A video of this simulation is available in the Supplemental Material.

To analyze this global rotation, we set up a cylindrical coordinate system (r, θ, z), where $r, z = 0$ is fixed at the geometric center of the cluster. We project the velocity U of each particle onto this cylindrical coordinate system and take the angular component U_θ. Then we estimate the normalized distribution $P(U_\theta, r)$ using data accumulated over this entire simulation. The result is shown as a two dimensional histogram in Fig. 13. The distribution $P(U_\theta, r)$ shows that the overall motion is close to that of a rigid body rotation $U_\theta \propto r$, where particles within the cluster all rotate about the central z-axis with roughly the same angular velocity. Only relatively few particles, far from the cluster center with approximately $r/a > 45$, moves more slowly than the cluster's global rotation. The expectation is that collectively induced velocities will decay as r^{-2} for $r >> 1$, as the cluster will appear as a rotlet singularity in the far-field.

In this system, for each time step approximately 2000 possible collisions are included in the collision resolution solver, and approximately 250 collisions actually happen. That is, the size of the set $\mathscr{A}$ is roughly 2000 and the size of the set $\mathscr{A}_c$ is roughly 250. The performance of the collision resolution algorithm over the course of the simulation is shown in Fig. 14. The simulation starts from a collision-free configuration, and then collisions are generated and reaches a steady state as the cluster remains close to a rigid body rotation, as shown above. In the beginning roughly 14 BBPGD steps are necessary to resolve the collisions and later this number increases to roughly 20. In comparison to the sedimentation case reported in Section §6, this rotor simulation involves 10 times the number of particles but the number of necessary BBPGD steps only slightly increases. Empirically, the number of BBPGD steps scales much slower than the number of particles. This feature makes this collision
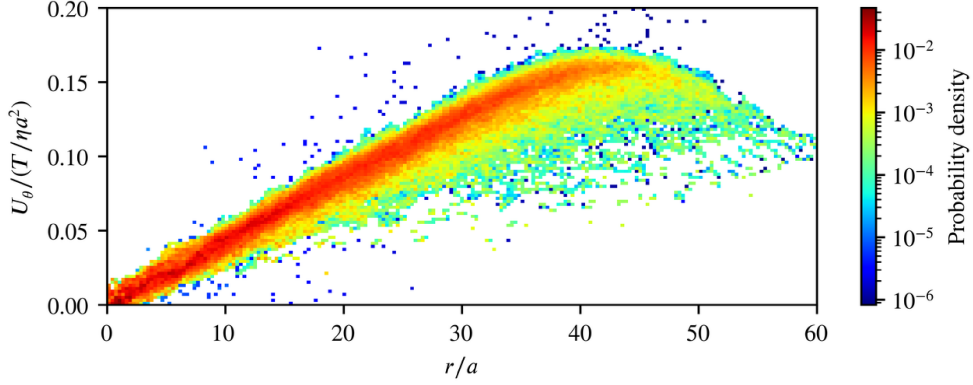
26

Figure 13: The normalized velocity distribution $P(U_\theta, r)$ for rotors in the cluster. The distribution P is normalized so that $\int_0^\infty \int_0^\infty P(U_\theta, r) 2\pi r \, dr \, dU = 1$. The data is accumulated over the entire simulation of $t = 500 \eta a^3/T$.

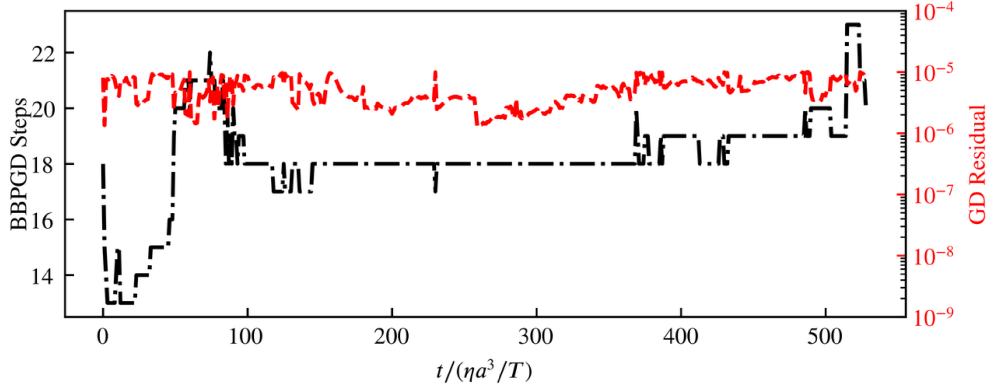resolution algorithm suitable for large scale simulations.

Figure 14: The performance of the BBPGD solver for 10,000 rotors in a spherical cluster. The simulation starts from a configuration with many collisions. The time-step $\Delta t = 1.0 \eta a^3/T$. The convergence tolerance of BBPGD is set to $\epsilon_{tol} = 10^{-5}$.

7.2. The dynamics within a monolayer of 20,000 rotors

Recent experiments and simulations have studied the dynamics of rotors confined to a surface within a fluid [65], or atop a solid substrate [67], with the driving external torque perpendicular to the surface or substrate. We use this setup to analyze the internal processes of a cluster of rotors, most especially the evolution of the collision network, using a disk of 20,000 monodisperse rotors confined in a monolayer, as shown in Fig. 1. We set $\Delta t = 0.5 \eta a^3/T$. The area fraction of particles is approximately 60% for this simulation, much denser than the previous example. As the collision radius of each particle is also set to $a_c = 1.1a$, the effective area fraction for collision resolution is around 70%. As a result, the rotors show a good deal of hexagonal ordering, as shown in Fig. 15. Even at such high densities, the BBPGD collision resolution solver takes about 20 descent steps per time step, similar to the spherical cluster example shown in Fig. 14.

The dynamics of the rotors is detailed in Fig. 15, where the hydrodynamic force density, i.e. the traction $\boldsymbol{f}$, the net collision force $\boldsymbol{\mathcal{F}}_c$ on each particle, and the collision force magnitude γ for each contact constraint are shown in the left, center, and right panels, respectively. Figure 15 (A) and (B) show a snapshot at times $t = 200 \eta a^3/T$ and $t = 350 \eta a^3/T$, respectively. The comparison between (A) and (B) clearly indicates that there are many small-scale motions caused by collisions, while the entire cluster is rotating collectively and differentially.

To examine this rotation, we compute the normalized distribution $P(U_\theta, r)$ as in Fig. 13. This is shown in Fig. 16. The radius of the disk is $R = 195a$, and $U_\theta \propto r$ holds only near the center of the disk, and at the disk
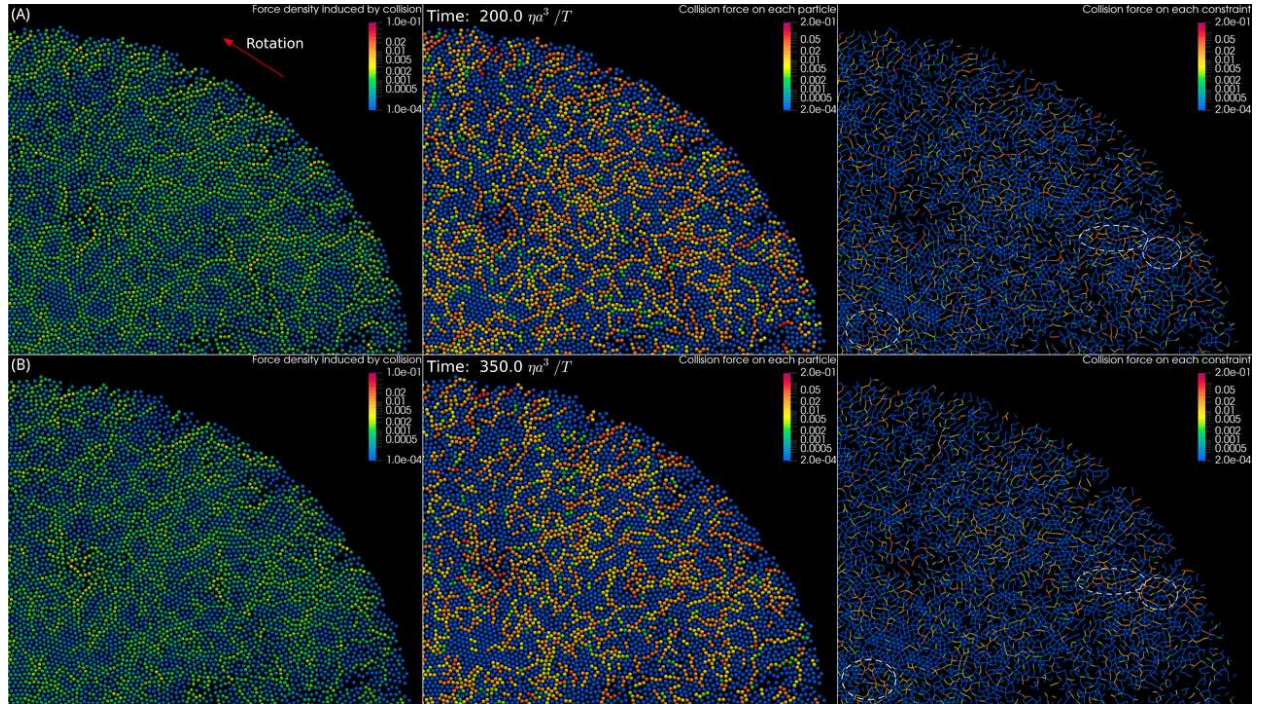
27

Figure 15: Snapshots of a quarter of the rotor disk at times $t = 200\eta a^3/T$ (A) and $350\eta a^3/T$ (B). Counterclockwise global rotation is driven by the torque $\boldsymbol{T}$ on each rotor, perpendicular to the disk. For (A) and (B), the left panels show the magnitude of hydrodynamic traction induced by collisions on each rotor surface, the middle panel shows the net collision force on each particle, and the right panel shows the collision force γ_ℓ on each contact constraint ℓ. The rightmost dashed circle marks a set of rotors that form a transient collision chain appearing in (B) only. Each of the other two white dashed circles on the left in (A) and (B) mark a single persistent collision chain. A video of this simulation is available in the Supplemental Material.

edge we observe a rapid increase in angular velocity. To understand this, we constructed a simple continuum model of the rotor assembly as an infinite number of rotlets at uniform density. This continuum model gives, for example, an angular velocity $U_\theta(r)$ with a logarithmic divergence at the disk edge:

$$\frac{U_\theta}{T/(\eta a^2)} \approx A \log(R-r) + B, \quad \text{as} \quad r \to R. \tag{51}$$

This calculation is detailed in Appendix B for the reader's interest. In Fig. 16 we show a fit (black dashed curve) to this form, i.e. for A and B, from our numerical data in the range $r > 170a$. It appears that the simulation has achieved sufficient scale to describe well a continuum of forced particles.
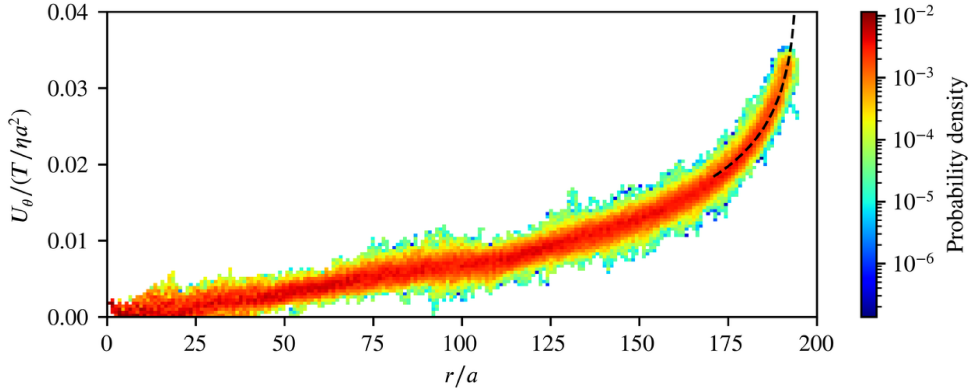
Figure 16: The normalized velocity distribution $P(U_\theta, r)$ for rotors in the monolayer disk. The distribution P is normalized so that $\int_0^\infty \int_0^\infty P(U_\theta, r) 2\pi r \, dr \, dU = 1$. The data is accumulated over the entire simulation of $t = 350\eta a^3/T$. The dashed curve is the function $A \log(R-r) + B$, where A and B are estimated by fitting to the simulation data in the range $r/a > 170$.

To quantify the time-scales induced by collisions, we analyze the lifetime of each collision constraint with collision force γ_ℓ, for both the set $\mathscr{A}$ for all constraints, and its subset $\mathscr{A}_c$ for active constraints. For the set $\mathscr{A}$ the lifetime of a constraint is defined as the number of time-steps k during which the constraint remain included in the LCP solver. For the subset $\mathscr{A}_c$, the lifetime is defined as the number of time-steps during which the solution $\gamma_\ell > 0$ for this constraint ℓ. The normalized distributions for these two cases are shown in Fig. 17.
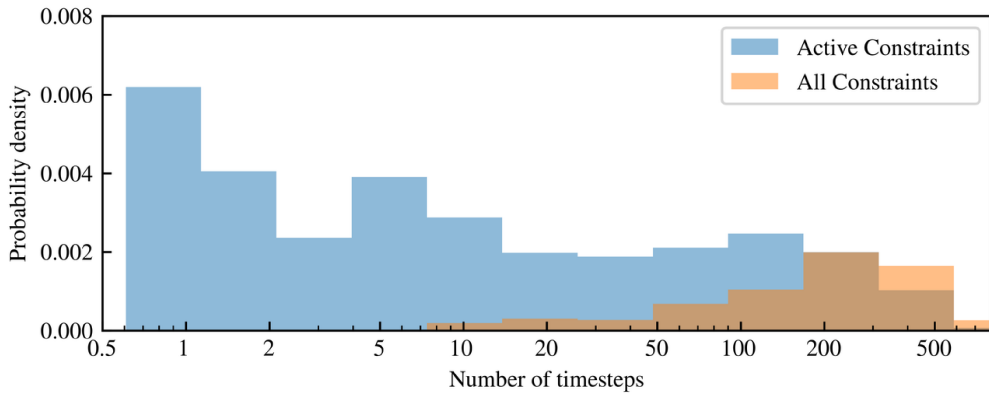
Figure 17: The normalized distribution $P(k)$ of constraint lifetimes. $P(k)$ is normalized so that $\sum_0^{k_{\max}} P(k) = 1$, where k is the number of time-steps, and $k_{\max}$ is the total number of time-steps. The blue and yellow distributions show $P(k)$ for the active constraint set $\mathscr{A}_c$ and the all constraint set $\mathscr{A}$, respectively.

Figure 17 shows that while many constraints in the active set $\mathscr{A}_c$ last for only 1 time-step, there is a broad region of short-lived constraints lasting between 2 and 20 time-steps. These longer-lived active constraints are of

29

relatively lower probability, but are not rare. However, almost no constraints in the set $\mathscr{A}$ have a lifetime shorter than 10 time-steps. The majority of them last more than 200 time-steps. This is because the set $\mathscr{A}$ depends solely on the particles' geometric configuration. While the rotation rate is differential in this simulation, neighboring pairs of particles rarely change their relative positions, and so the set $\mathscr{A}$ barely changes from step to step. This is shown in the right panels of Fig. 15(A) and (B). In contrast, the active constraint set $\mathscr{A}_c$ depends not only on geometry, but also on the velocity $\mathscr{U}_{nc}$, i.e., the particles' tendency to collide with each other. For this many-body problem, slight relative displacements of particles may induce significant changes in relative velocities, and generate very different $\mathscr{A}_c$ at different time-steps. As a final note, collisions may form 'collision force chains' when a few close-by particles run into one another like a chain. The two white dashed circles on the left in Fig. 15(A) and (B) mark persisting collision chains, while the rightmost white dashed circles mark a transient collision chain appearing in (B) only.

7.3. Dynamics of a tangentially forced layer of 20,000 rotors

In a last example, we place a square sheet of 20,000 rotors on the $z = 0$ plane, and again apply a torque T on each particle but now along the y-axis. Having the putative rotation axis aligned with the layer is a very different kind of forcing from the previous examples, and is conceptually akin to a vortex layer or sheet immersed in an inviscid fluid. We set $\Delta t = \eta a^3 / T$. The initial area fraction is still $\approx 60\%$ and we set the collision radius $a_c = 1.1a$ as in the previous examples.

The top and side views of simulation snapshots are shown in Fig. 18 and 19, where subfigures (A) - (D) show the magnitude of hydrodynamic traction, the collision force on each particle, and the collision force γ_ℓ on each constraint, at times $t = 54\eta a^3 / T$, $170\eta a^3 / T$, $300\eta a^3 / T$ and $689\eta a^3 / T$.

Fig. 18(A) shows the initial stage of system evolution, where particles driven by the torque tend to roll over each other and generate many collisions within the sheet, but with the activated constraints (right panel) being rather isolated instead of forming chains. At $t = 170\eta a^3 / T$, (B) shows the peak time of collision force in this simulated process and, as shown in the right panel, that many force chains are forming. Later, as shown in subfigure (C), the collision force decreases because narrow regions void of particles start to form. The most striking feature is the formation of strings of particles, or rollers in the vortical dynamics parlance, along the direction of the torque T, as shown in Fig. 18(D). These are reminiscent, perhaps, of the Kelvin-Helmholtz rolls that form from flat vortex sheets and layers [104, 105].

The formation of these rollers can also be seen from the side view Fig. 20, where the arrangement is similar to Fig. 19 but only 10% of particles close to the edge are shown. Inside each roller, the hydrodynamic traction distributed across the rotor surface induced by the torque T and collisions can be clearly seen. The formation of these chains of rotors are related to the flow generated by the torques. Since all rotors are rotating along the same $+y$ direction, a global flow is induced towards the $+x$ direction above the sheet, and towards the $-x$ direction below the sheet. This causes a jump of fluid velocity across the sheet. Further, because collectively the sheet maintains the thin layer geometry as shown in the side view Fig. 19, this jump of fluid velocity persists for a long time, and keeps driving the formation of chains. Again, this is similar to the Kelvin-Helmholtz instability, where the fluid velocity jumps across a vortical layer separating two fluids. However, for our rotor system the Reynolds number is zero, rather than infinity, and reflects the precise balance of drive and dissipation, rather than of dissipationless conservation laws and Hamiltonian structure (though see [106]). A complete investigation of these analogies is beyond the scope of this work on numerical methods , and we leave it for a future study.
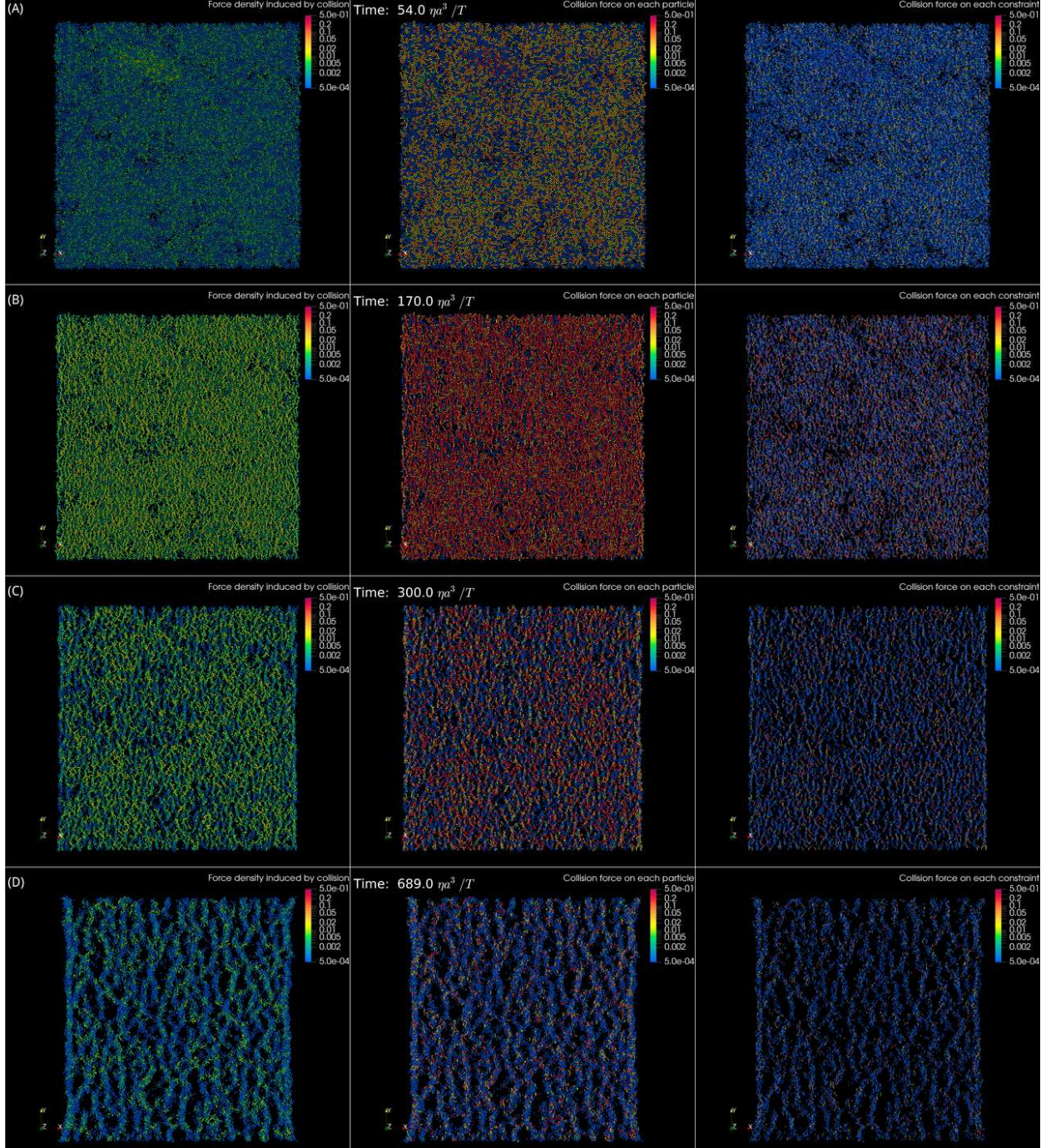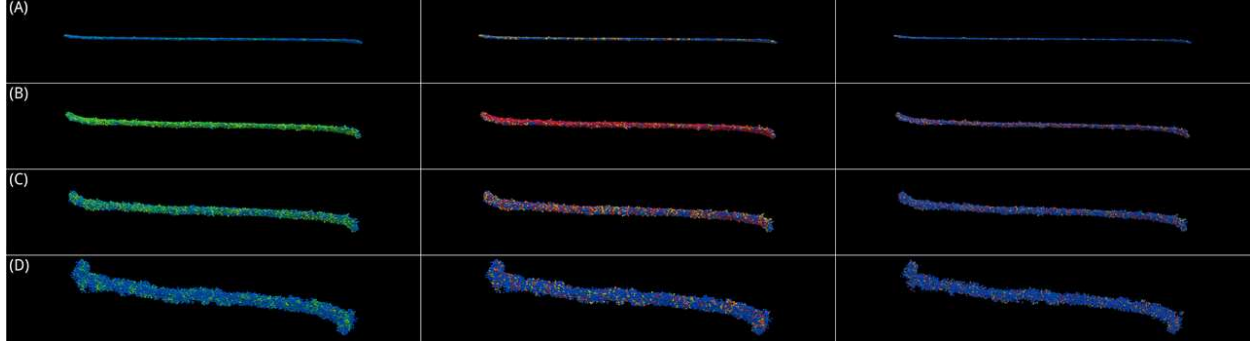
Figure 18: The top view of simulation snapshots for $20,000$ rotors on the $z = 0$ plane driven by torque $\boldsymbol{T}$ aligned with the y-axis. (A), (B), (C), and (D) are taken at $t = 54\eta a^3/T$, $170\eta a^3/T$, $300\eta a^3/T$ and $689\eta a^3/T$, respectively. The left panels show the magnitude of hydrodynamic traction induced by collisions on each rotor surface, the middle panels show the net collision force on each particle, and the right panels show the collision force γ_ℓ on each contact constraint ℓ. This arrangement is the same as the previous example Fig. 15. A video of this simulation is available in the Supplemental Material.

31

Figure 19: A side view of simulation snapshots for $20,000$ rotors on the $z = 0$ plane driven by torque $\boldsymbol{T}$ aligned with the y-axis. The three panels of (A), (B), (C), and (D) show the same visualizations as in Fig. 18, viewed from the negative y-axis.
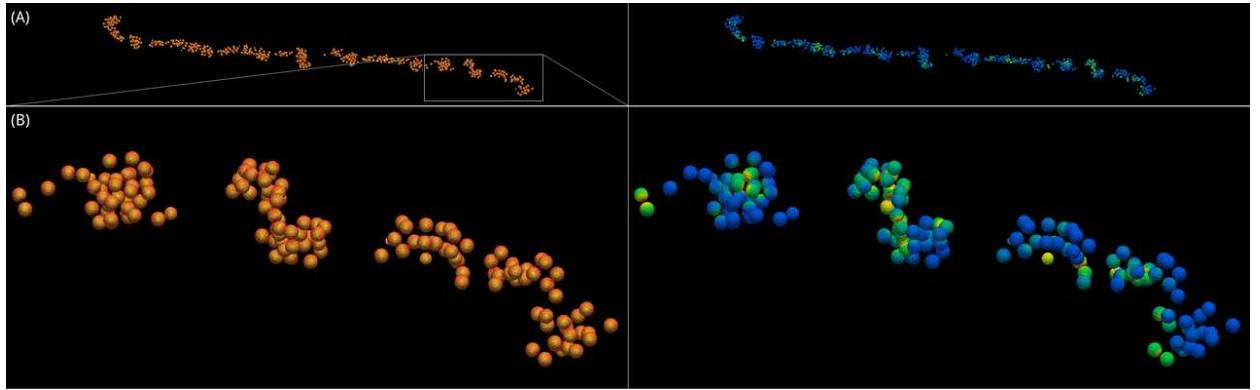
Figure 20: The side view of simulation snapshots at $t = 600\eta a^3/T$, showing only those particles close to the bottom edge of the structure shown in Fig. 18. The left panels in (A) and (B) show the hydrodynamic traction induced by the torque $\boldsymbol{T}$ on each particle, and the right panels show the hydrodynamic traction induced by collisions. The colormap is the same as the left panels in Fig. 18.

32

8. Conclusions

In this work we described a computational framework for simulating particulate Stokes suspensions. A key component is the collision resolution algorithm we extended from the LCP method for underdamped (inertial) granular flow [35, 39] to overdamped Stokes suspensions. The LCP is constructed at every time-step based on the non-overlapping geometric constraints and coupled to an explicit time-stepping scheme. The LCP is then converted to a CQP, utilizing the fact that the mobility matrix $\mathcal{M}$ is SPD, which is efficiently solved with the BBPGD method. This collision resolution algorithm addresses two important drawbacks in traditional collision resolution methods based on pairwise repulsive potentials: (i) the temporal stiffness induced by repulsive potentials, and (ii) the particles becoming effectively soft since the repulsive potentials cannot be infinitely stiff. This collision resolution method does not require explicit construction of the mobility matrix $\mathcal{M}$. Any mobility solver that is able to compute $\mathcal{U}$ with given force $\mathcal{F}$ can be used within this collision resolution algorithm as long as $\mathcal{M}$ is kept SPD. Further, the particles do not have to be spherical [63].

We then demonstrated the application of this method to suspensions of spherical particles, where the mobility solver is based on a new second-kind BI equation [70]. In particular, VSH expansions [64] are utilized to maintain high accuracy of the BI operators for close pairs of spheres. Consequently, this specialized mobility solver is well-conditioned in all test cases, even when particles are very close. The stability and scalability of our algorithm is demonstrated in Section §6, where we implemented these methods with full MPI and OpenMP parallelism. In the sedimentation and rotation tests, time-stepping remains stable although very large time-steps are purposefully used to demonstrate stability, even when particles move about $10 \sim 20\%$ of their radii within one time-step. In scalability benchmarks, systems of up to 8×10^4 spheres on 1792 cores are demonstrated. We believe the code can be successfully scaled to much larger systems on larger machines. Finally, in Section §7, we demonstrated the application of our method to suspensions of driven rotors, which illustrate the collectively-induced system-scale rotational motions, intricate microscopic collision networks, and a Kelvin-Helmholtz-like instability.

In comparison to a few other methods based on geometric constraints [29, 61], a key advantage of our method is that the collision force between each collision pair is individually computed and recorded. This preserves the entire collision force network information, which is necessary for computing the system collision stress. Our approach requires computing the collision force as a separate mobility problem, instead of embedding the minimization problem into the mobility solver, as done by Lu et al. [61, 62] in their work on deformable bodies (note that while the LCP collision resolution method is derived for rigid particles, it does allow the particles to deform outside this collision resolution stage). Therefore, our method has a higher cost because the dense mobility matrix $\mathcal{M}$ appears in the matrix A of the LCP (20). This extra cost can potentially be reduced by the matrix-splitting subspace optimization method [107], where in most stages of the minimization process only a block-diagonal part of $\mathcal{M}$ is used. We leave this to future work.

In other work we have demonstrated the applicability of our collision resolution scheme to evolving assemblies of Brownian spherocylinders [63]. There, we simulated dynamics of a system of growing and dividing cells, where cell sizes increase when they grow and decrease when they divide, and steric interactions are central to how the "colony" grows. Similarly, this computational framework can be directly applied to many other interesting physics and engineering problems, such as confined suspensions of swimmers [108] and cell packing in biofilms [109].

9. Acknowledgements

Our implementation of this framework will be released on GitHub (https://github.com/wenyan4work/SphereSimulator) as an open-source software following the publication of this article.

Appendix A. The VSH expansion of traction operator

The first row of :

$$g_{nm}^{rr} = \left(n f_{nm}^{W}{}'(r) - (n+1) f_{nm}^{V}{}'(r)\right) Y_n^m(\theta, \phi) \tag{A.1}$$

$$g_{nm}^{r\theta} = \frac{Y_n^m(\theta, \phi)(-m(n+2) f_{nm}^{V}(r)\cot(\theta) + m(n-1) f_{nm}^{W}(r)\cot(\theta) + im f_{nm}^{X}(r)\csc(\theta))}{r}$$
$$- \frac{e^{-i\phi}\sqrt{-m(m+1) + n^2 + n}((n+2) f_{nm}^{V}(r) - n f_{nm}^{W}(r) + f_{nm}^{W}(r)) Y_n^{m+1}(\theta, \phi)}{r} \tag{A.2}$$

$$g_{nm}^{r\phi} = \frac{m e^{-i\phi}\csc(\theta)(\sin(\phi) - i\cos(\phi)) Y_n^m(\theta, \phi)((n+2) f_{nm}^{V}(r) - n f_{nm}^{W}(r) + f_{nm}^{W}(r) - i f_{nm}^{X}(r)\cos(\theta))}{r}$$
$$- \frac{e^{-i\phi} f_{nm}^{X}(r)\sqrt{-m(m+1) + n^2 + n} Y_n^{m+1}(\theta, \phi)}{r} \tag{A.3}$$

The second row:

$$g_{nm}^{\theta r} = m\csc(\theta) Y_n^m(\theta, \phi)\left(\cos(\theta)\left(f_{nm}^{V}{}'(r) + f_{nm}^{W}{}'(r)\right) - i f_{nm}^{X}{}'(r)\right)$$
$$+ e^{-i\phi} Y_n^{m+1}(\theta, \phi)\left(\sqrt{(n-m)(m+n+1)} f_{nm}^{V}{}'(r) + \sqrt{(n-m)(m+n+1)} f_{nm}^{W}{}'(r)\right) \tag{A.4}$$

$$g_{nm}^{\theta\theta} = \left[\frac{f_{nm}^{V}(r)\left(-m\left(\csc^2(\theta) - m\cot^2(\theta)\right) - n - 1\right)}{r} + \frac{f_{nm}^{W}(r)\left(m^2\cot^2(\theta) - m\csc^2(\theta) + n\right)}{r}\right.$$
$$\left. - \frac{i(m-1)m f_{nm}^{X}(r)\cot(\theta)\csc(\theta)}{r}\right] Y_n^m(\theta, \phi)$$
$$+ \left(\frac{e^{-i\phi}\sqrt{-m(m+1) + n^2 + n}\csc(\theta)((2m+1)\cos(\theta)(f_{nm}^{V}(r) + f_{nm}^{W}(r)) - im f_{nm}^{X}(r))}{r}\right) Y_n^{m+1}(\theta, \phi)$$
$$+ \left(\frac{e^{-2i\phi}\sqrt{(m-n)(m-n+1)(m+n+1)(m+n+2)}(f_{nm}^{V}(r) + f_{nm}^{W}(r))}{r}\right) Y_n^{m+2}(\theta, \phi) \tag{A.5}$$

$$g_{nm}^{\theta\phi} = \frac{m Y_n^m(\theta, \phi)(f_{nm}^{X}(r) + (m-1)\csc(\theta)(f_{nm}^{X}(r)\csc(\theta) + i\cot(\theta)(f_{nm}^{V}(r) + f_{nm}^{W}(r))))}{r}$$
$$+ \left[\frac{im e^{-i\phi} f_{nm}^{V}(r)\sqrt{-m(m+1) + n^2 + n}\csc(\theta)}{r} + \frac{im e^{-i\phi} f_{nm}^{W}(r)\sqrt{-m(m+1) + n^2 + n}\csc(\theta)}{r}\right.$$
$$\left. - \frac{e^{-i\phi} f_{nm}^{X}(r)\sqrt{-m(m+1) + n^2 + n}\cot(\theta)}{r}\right] Y_n^{m+1}(\theta, \phi) \tag{A.6}$$

The third row:

$$g_{nm}^{\phi r} = m\csc(\theta)Y_n^m(\theta,\phi)\left(\cos(\theta)f_{nm}^{X}{}'(r) + i\left(f_{nm}^{V}{}'(r) + f_{nm}^{W}{}'(r)\right)\right)$$
$$+ e^{-i\phi}\sqrt{(n-m)(m+n+1)}f_{nm}^{X}{}'(r)Y_n^{m+1}(\theta,\phi) \tag{A.7}$$

$$g_{nm}^{\phi\theta} = \left[\frac{i(m-1)mf_{nm}^{V}(r)\cot(\theta)\csc(\theta)}{r} + \frac{i(m-1)mf_{nm}^{W}(r)\cot(\theta)\csc(\theta)}{r}\right.$$
$$\left.+ \frac{mf_{nm}^{X}(r)\csc^2(\theta)(m\cos(2\theta)+m-2)}{2r}\right]Y_n^m(\theta,\phi)$$

$$\left[\frac{ime^{-i\phi}f_{nm}^{V}(r)\sqrt{-m(m+1)+n^2+n}\,\csc(\theta)}{r} + \frac{ime^{-i\phi}f_{nm}^{W}(r)\sqrt{-m(m+1)+n^2+n}\,\csc(\theta)}{r}\right.$$
$$\left.+ \frac{(2m+1)e^{-i\phi}f_{nm}^{X}(r)\sqrt{-m(m+1)+n^2+n}\,\cot(\theta)}{r}\right]Y_n^{m+1}(\theta,\phi)$$

$$\left[\frac{ime^{-i\phi}f_{nm}^{V}(r)\sqrt{-m(m+1)+n^2+n}\,\csc(\theta)}{r} + \frac{ime^{-i\phi}f_{nm}^{W}(r)\sqrt{-m(m+1)+n^2+n}\,\csc(\theta)}{r}\right.$$
$$\left.+ \frac{(2m+1)e^{-i\phi}f_{nm}^{X}(r)\sqrt{-m(m+1)+n^2+n}\,\cot(\theta)}{r}\right]Y_n^{m+2}(\theta,\phi) \tag{A.8}$$

$$g_{nm}^{\phi\phi} = \left[\frac{f_{nm}^{V}(r)\left(-(m-1)m\csc^2(\theta)-m-n-1\right)}{r} + \frac{f_{nm}^{W}(r)\left(m\left(\cot^2(\theta)-m\csc^2(\theta)\right)+n\right)}{r}\right.$$
$$\left.+ \frac{i(m-1)mf_{nm}^{X}(r)\cot(\theta)\csc(\theta)}{r}\right]Y_n^m(\theta,\phi)$$

$$\left[\frac{e^{-i\phi}f_{nm}^{V}(r)\sqrt{-m(m+1)+n^2+n}\,\cot(\theta)}{r} + \frac{e^{-i\phi}f_{nm}^{W}(r)\sqrt{-m(m+1)+n^2+n}\,\cot(\theta)}{r}\right.$$
$$\left.+ \frac{ime^{-i\phi}f_{nm}^{X}(r)\sqrt{-m(m+1)+n^2+n}\,\csc(\theta)}{r}\right]Y_n^{m+1}(\theta,\phi) \tag{A.9}$$

Appendix B. The singularity of velocity close to the disk edge of Stokes rotlets

Appendix B.1. Geometry and setup

We consider a domain D of a disk on $z = 0$ plane:

$$D = \{(x,y)\,|\,x^2 + y^2 < R^2\} \tag{B.1}$$

We denote the number density of particles within this disk as n. A constant torque $\boldsymbol{T}$ toward $+z$ direction is exerted on each particle. We assume that each particle follows the fluid velocity induced by other particles. Each particle induces a rotational fluid flow u_j, as given by the Stokes rotlet velocity field:

$$u_j = \frac{\epsilon_{jlm}}{8\pi\mu}\frac{r_m}{r^3}T_l, \tag{B.2}$$

where ϵ_{jlm} is the Levi-Civita tensor. $\boldsymbol{r}$ is the vector pointing from the rotlet to the point where u_j is evaluated.

Due to the symmetry of this disk D, we consider a point $(s,0)$ on the x-axis. The velocity $\boldsymbol{u}$ at this point is aligned with the y-axis, given by an integral over the rotlets on this entire disk:

$$\frac{8\pi\mu}{nT}u(s) = \int_0^R 2\pi r\int_0^{2\pi}\frac{s-r\cos\theta}{\left((s-r\cos\theta)^2+r^2\sin^2\theta\right)^3}d\theta\,dr, \tag{B.3}$$

where r, θ is the cylindrical coordinate system used to denote the rotlet point in this disk of radius R. $\fint$ denotes the principal value of this integral, because this integral involves a high order of singularity at the point $(r, 0)$.

The integral over θ can be computed to get the closed form:

$$F(r) = 2\left(\frac{K\left(\frac{4rs}{(r+s)^2}\right)}{s(r+s)} + \frac{E\left(\frac{4rs}{(r+s)^2}\right)}{s(s-r)} \right), \tag{B.4}$$

where $K(x)$ is the complete elliptic integral of the first kind [110], and $E(x)$ is the complete elliptic integral of the second kind [111]. $K(x)$ is singular at $x = 1$, i.e., $r = s$. So the two terms of $F(r)$ are both singular at $r = s$. As a result, the velocity $u(s)$ becomes:

$$\frac{8\pi\mu}{nT}u(s) = \fint_0^R F(r)2\pi r dr, \tag{B.5}$$

where $0 < s < R$. The principal value can be computed as this limit as $\delta \to 0^+$:

$$\frac{8\pi\mu}{nT}u(s) = \lim_{\delta \to 0+}\left(\int_0^{s-\delta} F(r)2\pi r dr + \int_{s+\delta}^R F(r)2\pi r dr \right). \tag{B.6}$$

Appendix B.2. Principal value and singularity

Now we discuss the singularity in a small region $[s-a, s+b]$ around s, where $s-a < s-\delta < s < s+\delta < s+b$. In this small region, we use the asymptotic expansion of the integrand $rF(r)$ around $r = s$ [112]:

$$rF(r) \approx \frac{-2\log(s-r) - \log\left(\frac{1}{4s^2}\right) - 4 + 4\log(2)}{2s} - \frac{2}{r-s} \quad \text{when } r < s \tag{B.7}$$

$$\approx \frac{-2\log(r-s) - \log\left(\frac{1}{4s^2}\right) - 4 + 4\log(2)}{2s} - \frac{2}{r-s} \quad \text{when } r > s \tag{B.8}$$

In the small region $[s-a, s+b]$, the integral can be asymptotically computed:

$$\fint_{s-a}^{s+b} rF(r)dr = \int_{s-a}^{s-\delta} F(r)rdr + \int_{s+\delta}^{s+b} F(r)rdr \tag{B.9}$$

$$\approx \frac{(\delta - a)\log\left(\frac{1}{s^2}\right) + 4s\log\left(\frac{a}{\delta}\right) + (\log(64) - 2)(a - \delta) - 2a\log(a) + 2\delta\log(\delta)}{2s}$$

$$+ \frac{(b - \delta)\left(-\log\left(\frac{1}{s^2}\right) - 2 + \log(64)\right) + 4s\log\left(\frac{\delta}{b}\right) - 2b\log(b) + 2\delta\log(\delta)}{2s} \tag{B.10}$$

$$= \frac{(\log(64) - 2)(a + b - 2\delta) - (a + b)\log\left(\frac{1}{s^2}\right) + 4s\log\left(\frac{a}{b}\right) - 2a\log(a) - 2b\log(b) + 2\delta\log\left(\frac{\delta^2}{s^2}\right)}{2s} \tag{B.11}$$

The limit as $\delta \to 0$ exists:

$$\lim_{\delta \to 0}\fint_{s-a}^{s+b} rF(r)dr = 2\log\left(\frac{a}{b}\right)$$

$$+ \frac{-\frac{1}{2}a\log\left(\frac{1}{s^2}\right) - a + \frac{1}{2}a\log(64) - a\log(a) - \frac{1}{2}b\log\left(\frac{1}{s^2}\right) - b + \frac{1}{2}b\log(64) - b\log(b)}{s} \tag{B.12}$$

This gives the behavior of the singularity around the point $(s, 0)$.

When the point $(s, 0)$ is close to the edge of the disk, i.e., $s \to R^-$:

$$b = R - s \to 0^+ \tag{B.13}$$

36

The behavior of this pole can be calculated as the limit of $b \to 0^+$:

$$\left(\lim_{\delta \to 0} \int_{s-a}^{s+b} r F(r) dr \right) = -2 \log b - \frac{1}{R} b \log b + O(b) + C(a) \quad \text{as } b \to 0^+ \tag{B.14}$$

where we have used $s \to R^-$. $O(b)$ is of order b or higher. $C(a)$ is a function involving a and R only.

In sum, for a target point at $(s, 0)$ close to the disk edge, the velocity shows a logarithm singularity dominated by the $\log b$ term:

$$u = A \log(R - s) + B + O\left(\frac{R - s}{R} \log(R - s) \right), \tag{B.15}$$

where A and B are two constants to be determined.

References

References

[1] N. J. Wagner, J. F. Brady, Shear thickening in colloidal dispersions, Physics Today 62 (2009) 27–32. doi:10.1063/1.3248476.

[2] Y. S. Lee, E. D. Wetzel, N. J. Wagner, The ballistic impact characteristics of kevlar (r) woven fabrics impregnated with a colloidal shear thickening fluid, Journal of Materials Science 38 (2003) 2825–2833. doi:Doi10.1023/A:1024424200221.

[3] V. Trappe, V. Prasad, L. Cipelletti, P. N. Segre, D. A. Weitz, Jamming phase diagram for attractive particles, Nature 411 (2001) 772–775. doi:10.1038/35081021.

[4] Anderson, Lekkerkerker, Insights into phase transition kinetics from colloid science, Nature (2002).

[5] D. Saintillan, M. J. Shelley, Active suspensions and their nonlinear models, Comptes Rendus Physique 14 (2013) 497–517. doi:10.1016/j.crhy.2013.04.001.

[6] M. C. Marchetti, J.-F. Joanny, S. Ramaswamy, T. B. Liverpool, J. Prost, M. Rao, R. A. Simha, Hydrodynamics of soft active matter, Reviews of Modern Physics 85 (2013) 1143.

[7] M. J. Shelley, The dynamics of microtubule/motor-protein assemblies in biology and physics, Annual Review of Fluid Mechanics 48 (2016) 487–506.

[8] D. Needleman, Z. Dogic, Active matter at the interface between materials science and cell biology, Nature Reviews Materials 2 (2017) 17048.

[9] D. R. Foss, J. F. Brady, Brownian dynamics simulation of hard-sphere colloidal dispersions, Journal of Rheology 44 (2000) 629–651. doi:Doi10.1122/1.551104.

[10] J. Rotne, S. Prager, Variational treatment of hydrodynamic interaction in polymers, The Journal of Chemical Physics 50 (1969) 4831–4837. doi:10.1063/1.1670977.

[11] H. Yamakawa, Transport properties of polymer chains in dilute solution: Hydrodynamic interaction, The Journal of Chemical Physics 53 (1970) 436–443. doi:10.1063/1.1673799.

[12] P. J. Zuk, E. Wajnryb, K. A. Mizerski, P. Szymczak, Rotne-Prager-Yamakawa approximation for different-sized particles in application to macromolecular bead models, Journal of Fluid Mechanics 741 (2014). URL: http://journals.cambridge.org/article_S002211201300668X. doi:10.1017/jfm.2013.668.

[13] E. Wajnryb, K. A. Mizerski, P. J. Zuk, P. Szymczak, Generalization of the Rotne-Prager-Yamakawa mobility and shear disturbance tensors, Journal of Fluid Mechanics 731 (2013). URL: http://journals.cambridge.org/article_S0022112013004023. doi:10.1017/jfm.2013.402.

[14] K. A. Mizerski, E. Wajnryb, P. J. Zuk, P. Szymczak, The Rotne-Prager-Yamakawa approximation for periodic systems in a shear flow, The Journal of Chemical Physics 140 (2014) 184103. doi:10.1063/1.4871113.

[15] C. W. J. Beenakker, Ewald sum of the Rotne-Prager tensor, Journal of Chemical Physics 85 (1986) 1581–1582. doi:Doi10.1063/1.451199.

[16] Z. Liang, Z. Gimbutas, L. Greengard, J. Huang, S. Jiang, A fast multipole method for the Rotne-Prager-Yamakawa tensor and its applications, Journal of Computational Physics 234 (2013) 133–139. doi:10.1016/j.jcp.2012.09.021.

[17] W. Guan, X. Cheng, J. Huang, G. Huber, W. Li, J. A. McCammon, B. Zhang, RPYFMM: Parallel adaptive fast multipole method for Rotne-Prager-Yamakawa tensor in biomolecular hydrodynamics simulations, Computer Physics Communications 227 (2018) 99–108. doi:10.1016/j.cpc.2018.02.005.

[18] L. Durlofsky, J. F. Brady, G. Bossis, Dynamic simulation of hydrodynamically interacting particles, Journal of Fluid Mechanics 180 (1987) 21–49. doi:10.1017/S002211208700171x.

[19] J. F. Brady, G. Bossis, Stokesian dynamics, Annual Review of Fluid Mechanics 20 (1988) 111–157. doi:10.1146/annurev.fl.20.010188.000551.

[20] T. N. Phung, J. F. Brady, G. Bossis, Stokesian dynamics simulation of brownian suspensions, Journal of Fluid Mechanics 313 (1996) 181–207. doi:10.1017/S0022112096002170.

[21] A. Sierou, J. F. Brady, Accelerated stokesian dynamics simulations, Journal of Fluid Mechanics 448 (2001) 115–146. doi:10.1017/s0022112001005912.

[22] M. Wang, J. F. Brady, Spectral ewald acceleration of stokesian dynamics for polydisperse suspensions, Journal of Computational Physics 306 (2016) 443–477.

[23] R. T. Bonnecaze, J. F. Brady, Yield stresses in electrorheological fluids, Journal of Rheology 36 (1992) 73–115. doi:Doi10.1122/1.550343.

[24] R. T. Bonnecaze, J. F. Brady, Dynamic simulation of an electrorheological fluid, Journal of Chemical Physics 96 (1992) 2183–2202. doi:Doi10.1063/1.462070.

[25] I. L. Claeys, J. F. Brady, Suspensions of prolate spheroids in stokes flow. part 1. dynamics of a finite number of particles in an unbounded fluid, Journal of Fluid Mechanics 251 (1993-06) 411–442. doi:10.1017/S0022112093003465.

[26] S. Kim, S. J. Karrila, Microhydrodynamics: Principles and Selected Applications, Courier Corporation, 2005.

[27] D. J. Pine, J. P. Gollub, J. F. Brady, A. M. Leshansky, Chaos and threshold for irreversibility in sheared suspensions, Nature 438 (2005) 997.

[28] J. a. Janela, A. Lefebvre, B. Maury, A penalty method for the simulation of fluid - rigid body interaction, ESAIM: Proceedings 14 (2005) 115–123. doi:10.1051/proc:2005010.

[29] B. Maury, A time-stepping scheme for inelastic collisions, Numerische Mathematik 102 (2006) 649–679. doi:10.1007/s00211-005-0666-6.

[30] P. A. Cundall, O. D. Strack, A discrete numerical model for granular assemblies, geotechnique 29 (1979) 47–65.

[31] P. Lötstedt, Mechanical Systems of Rigid Bodies Subject to Unilateral Constraints, SIAM Journal on Applied Mathematics 42 (1982) 281–296. doi:10.1137/0142022.

[32] Chunsheng Cai, B. Roth, On the spatial motion of a rigid body with point contact, in: 1987 IEEE International Conference on Robotics and Automation Proceedings, volume 4, 1987, pp. 686–695. doi:10.1109/ROBOT.1987.1087971.

[33] D. J. Montana, The Kinematics of Contact and Grasp, The International Journal of Robotics Research 7 (1988) 17–32. doi:10.1177/027836498800700302.

[34] D. Baraff, Issues in computing contact forces for non-penetrating rigid bodies, Algorithmica 10 (1993) 292. doi:10.1007/BF01891843.

[35] M. Anitescu, J. F. Cremer, F. A. Potra, Formulating Three-Dimensional Contact Dynamics Problems, Mechanics of Structures and Machines 24 (1996) 405–437. doi:10.1080/08905459608905271.

[36] D. E. Stewart, J. C. Trinkle, An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction, International Journal for Numerical Methods in Engineering 39 (1996) 2673–2691.

[37] M. Anitescu, F. A. Potra, Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems, Nonlinear Dynamics 14 (1997) 231–247. doi:10.1023/A:1008292328909.

[38] D. E. Stewart, Convergence of a Time-Stepping Scheme for Rigid-Body Dynamics and Resolution of Painlevé's Problem, Archive for Rational Mechanics and Analysis 145 (1998) 215–260. doi:10.1007/s002050050129.

[39] M. Anitescu, F. A. Potra, D. E. Stewart, Time-stepping for three-dimensional rigid body dynamics, Computer Methods in Applied Mechanics and Engineering 177 (1999) 183–197. doi:10.1016/S0045-7825(98)00380-6.

[40] D. Stewart, Rigid-Body Dynamics with Friction and Impact, SIAM Review 42 (2000) 3–39. doi:10.1137/S0036144599360110.

[41] M. Anitescu, F. A. Potra, A time-stepping method for stiff multibody dynamics with contact and friction, International Journal for Numerical Methods in Engineering 55 (2002) 753–784. doi:10.1002/nme.512.

[42] M. Anitescu, G. D. Hart, A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction, International Journal for Numerical Methods in Engineering 60 (2004) 2335–2371. doi:10.1002/nme.1047.

[43] M. Anitescu, Optimization-based simulation of nonsmooth rigid multibody dynamics, Mathematical Programming 105 (2006) 113–143. doi:10.1007/s10107-005-0590-7.

[44] A. Tasora, D. Negrut, M. Anitescu, Large-scale parallel multi-body dynamics with frictional contact on the graphical processing unit, Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics 222 (2008) 315–326. doi:10.1243/14644193JMBD154.

[45] A. Tasora, M. Anitescu, A Fast NCP Solver for Large Rigid-Body Problems with Contacts, Friction, and Joints, in: Multibody Dynamics, Computational Methods in Applied Sciences, Springer, Dordrecht, 2009, pp. 45–55. 10.1007/978-1-4020-8829-2_3.

[46] A. Tasora, M. Anitescu, A Convex Complementarity Approach for Simulating Large Granular Flows, Journal of Computational and Nonlinear Dynamics 5 (2010) 031004–031004–10. doi:10.1115/1.4001371.

[47] A. Tasora, M. Anitescu, A matrix-free cone complementarity approach for solving large-scale, nonsmooth, rigid body dynamics, Computer Methods in Applied Mechanics and Engineering 200 (2011) 439–453. doi:10.1016/j.cma.2010.06.030.

[48] A. Tasora, D. Negrut, M. Anitescu, GPU-Based Parallel Computing for the Simulation of Complex Multibody Systems with Unilateral and Bilateral Constraints: An Overview, in: Multibody Dynamics, Computational Methods in Applied Sciences, Springer, Dordrecht, 2011, pp. 283–307. 10.1007/978-90-481-9971-6_14.

[49] D. Negrut, A. Tasora, H. Mazhar, T. Heyn, P. Hahn, Leveraging parallel computing in multibody dynamics, Multibody System Dynamics 27 (2012) 95–117. doi:10.1007/s11044-011-9262-y.

[50] A. Tasora, M. Anitescu, A complementarity-based rolling friction model for rigid contacts, Meccanica 48 (2013) 1643–1659. doi:10.1007/s11012-013-9694-y.

[51] T. Heyn, H. Mazhar, A. Pazouki, D. Melanz, A. Seidl, J. Madsen, A. Bartholomew, D. Negrut, D. Lamb, A. Tasora, Chrono: A Parallel Physics Library for Rigid-Body, Flexible-Body, and Fluid Dynamics, ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference 7B (2013) V07BT10A050. doi:10.1115/DETC2013-13239.

[52] H. Mazhar, T. Heyn, A. Pazouki, D. Melanz, A. Seidl, A. Bartholomew, A. Tasora, D. Negrut, CHRONO: A parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics, Mechanical Sciences 4 (2013) 49–64. doi:10.5194/ms-4-49-2013.

[53] A. Tasora, M. Anitescu, S. Negrini, D. Negrut, A compliant visco-plastic particle contact model based on differential variational inequalities, International Journal of Non-Linear Mechanics 53 (2013) 2–12. doi:10.1016/j.ijnonlinmec.2013.01.010.

[54] A. Pazouki, M. Kwarta, K. Williams, W. Likos, R. Serban, P. Jayakumar, D. Negrut, Compliant contact versus rigid contact: A comparison

in the context of granular dynamics, Physical Review E 96 (2017) 042905. doi:10.1103/PhysRevE.96.042905.

[55] M. Anitescu, A. Tasora, An iterative approach for cone complementarity problems for nonsmooth dynamics, Computational Optimization and Applications 47 (2010) 207–235. doi:10.1007/s10589-008-9223-4.

[56] T. Heyn, M. Anitescu, A. Tasora, D. Negrut, Using Krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation, International Journal for Numerical Methods in Engineering 95 (2013) 541–561. doi:10.1002/nme.4513.

[57] H. Mazhar, T. Heyn, D. Negrut, A. Tasora, Using Nesterov's Method to Accelerate Multibody Dynamics with Friction and Contact, ACM Trans. Graph. 34 (2015) 32:1–32:14. doi:10.1145/2735627.

[58] D. Melanz, L. Fang, P. Jayakumar, D. Negrut, A comparison of numerical methods for solving multibody dynamics problems with frictional contact modeled via differential variational inequalities, Computer Methods in Applied Mechanics and Engineering 320 (2017) 668–693. doi:10.1016/j.cma.2017.03.010.

[59] E. Corona, D. Gorsich, P. Jayakumar, S. Veerapaneni, Tensor train accelerated solvers for nonsmooth rigid body dynamics, arXiv preprint arXiv:1808.02558 (2018).

[60] S. De, E. Corona, P. Jayakumar, S. Veerapaneni, Scalable solvers for cone complementarity problems in frictional multibody dynamics, Proceedings of IEEE Conference on High Performance Extreme Computing To appear (2019).

[61] L. Lu, A. Rahimian, D. Zorin, Contact-aware simulations of particulate Stokesian suspensions, Journal of Computational Physics 347 (2017) 160–182. URL: http://www.sciencedirect.com/science/article/pii/S002199911730493X. doi:10.1016/j.jcp.2017.06.039.

[62] L. Lu, A. Rahimian, D. Zorin, Parallel contact-aware simulations of deformable particles in 3d stokes flow, arXiv preprint arXiv:1812.04719 (2018).

[63] W. Yan, H. Zhang, M. Shelley, Computing collision stress in assemblies of active spherocylinders: applications of a fast and generic geometric method, The Journal of Chemical Physics 150 (2019). doi:arXiv:1811.04736.

[64] E. Corona, S. Veerapaneni, Boundary integral equation analysis for suspension of spheres in stokes flow, Journal of Computational Physics 362 (2018-06) 327–345. doi:10.1016/j.jcp.2018.02.017.

[65] K. Yeo, E. Lushi, P. M. Vlahovska, Collective Dynamics in a Binary Mixture of Hydrodynamically Coupled Microrotors, Physical Review Letters 114 (2015) 188301. doi:10.1103/PhysRevLett.114.188301.

[66] N. Oppenheimer, D. B. Stein, M. J. Shelley, Hydrosteric crystallization of rotating membrane inclusions, to appear, Physical Review Letters (2019).

[67] V. Soni, E. Bililign, S. Magkiriadou, S. Sacanna, D. Bartolo, M. J. Shelley, W. T. M. Irvine, The odd free surface flows of a colloidal chiral fluid, to appear, Nature Physics (2019).

[68] G. Kokot, S. Das, R. G. Winkler, G. Gompper, I. S. Aranson, A. Snezhko, Active turbulence in a gas of self-assembled spinners, Proceedings of the National Academy of Sciences 114 (2017) 12870–12875. doi:10.1073/pnas.1710188114.

[69] H. Power, G. Miranda, Second kind integral equation formulation of stokes' flows past a particle of arbitrary shape, SIAM Journal on Applied Mathematics 47 (1987) 689–698.

[70] 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 (2017-03) 504–519. doi:10.1016/j.jcp.2016.12.018.

[71] A.-K. Tornberg, K. Gustavsson, A numerical method for simulations of rigid fiber suspensions, Journal of Computational Physics 215 (2006) 172–196. URL: http://www.sciencedirect.com/science/article/pii/S0021999105004845. doi:10.1016/j.jcp.2005.10.028.

[72] K. Gustavsson, A.-K. Tornberg, Gravity induced sedimentation of slender fibers, Physics of Fluids (1994-present) 21 (2009) 123301. URL: http://scitation.aip.org/content/aip/journal/pof2/21/12/10.1063/1.3273091. doi:10.1063/1.3273091.

[73] S. Delong, F. B. Usabiaga, A. Donev, Brownian dynamics of confined rigid bodies, The Journal of Chemical Physics 143 (2015) 144107. doi:10.1063/1.4932062.

[74] S. Niebe, K. Erleben, Numerical methods for linear complementarity problems in physics-based animation, Morgan & Claypool Publishers, 2015.

[75] R. Fletcher, On the Barzilai-Borwein method, in: L. Qi, K. Teo, X. Yang (Eds.), Optimization and Control with Applications, Springer US, Boston, MA, 2005, pp. 235–256.

[76] Y.-H. Dai, R. Fletcher, Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming, Numerische Mathematik 100 (2005) 21–47. doi:10.1007/s00211-004-0569-y.

[77] Y.-H. Dai, W. W. Hager, K. Schittkowski, H. Zhang, The cyclic Barzilai-Borwein method for unconstrained optimization, IMA Journal of Numerical Analysis 26 (2006) 604–627. doi:10.1093/imanum/drl006.

[78] O. L. Mangasarian, T. H. Shiau, Error bounds for monotone linear complementarity problems, Mathematical Programming 36 (1986) 81–89. doi:10.1007/BF02591991.

[79] Y. Lin, J. Pang, Iterative Methods for Large Convex Quadratic Programs: A Survey, SIAM Journal on Control and Optimization 25 (1987) 383–411. doi:10.1137/0325023.

[80] S. Jiang, Z. Liang, J. Huang, A fast algorithm for Brownian dynamics simulation with hydrodynamic interactions, Mathematics of Computation 82 (2013) 1631–1645. doi:10.1090/S0025-5718-2013-02672-5.

[81] H. C. Brinkman, A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles, Flow, Turbulence and Combustion 1 (1949) 27. URL: https://doi.org/10.1007/BF02120313. doi:10.1007/BF02120313.

[82] J. R. Blake, A spherical envelope approach to ciliary propulsion, Journal of Fluid Mechanics 46 (1971) 199–208.

[83] T. Ishikawa, T. Pedley, Coherent structures in monolayers of swimming particles, Physical review letters 100 (2008) 088103.

[84] C. Pozrikidis, Boundary Integral and Singularity Methods for Linearized Viscous Flow, Cambridge Texts in Applied Mathematics, Cambridge University Press, 1992.

[85] M. Rachh, L. Greengard, Integral equation methods for elastance and mobility problems in two dimensions, SIAM Journal on

Numerical Analysis 54 (2016) 2889–2909.

[86] B. Wu, H. Zhu, A. Barnett, S. Veerapaneni, Solution of stokes flow in complex nonsmooth 2d geometries via a linear-scaling high-order adaptive integral equation scheme, Journal of Computational Physics Under review (2019).

[87] M. Siegel, A.-K. Tornberg, A local target specific quadrature by expansion method for evaluation of layer potentials in 3D, Journal of Computational Physics 364 (2018) 365–392. URL: http://www.sciencedirect.com/science/article/pii/S002199911830158X. doi:10.1016/j.jcp.2018.03.006.

[88] M. Wala, A. Klöckner, A fast algorithm for Quadrature by Expansion in three dimensions, Journal of Computational Physics 388 (2019) 655–689. doi:10.1016/j.jcp.2019.03.024.

[89] C. Pérez-Arancibia, L. M. Faria, C. Turc, Harmonic density interpolation methods for high-order evaluation of Laplace layer potentials in 2D and 3D, Journal of Computational Physics 376 (2019) 411–434. doi:10.1016/j.jcp.2018.10.002.

[90] D. Malhotra, G. Biros, PVFMM: A Parallel Kernel Independent FMM for Particle and Volume Potentials, Communications in Computational Physics 18 (2015) 808–830. URL: http://journals.cambridge.org/article_S181524061500081X. doi:10.4208/cicp.020215.150515sw.

[91] M. Iwasawa, A. Tanikawa, N. Hosono, K. Nitadori, T. Muranushi, J. Makino, Implementation and performance of FDPS: A framework for developing parallel particle simulation codes, Publications of the Astronomical Society of Japan 68 (2016) 54. URL: http://pasj.oxfordjournals.org/lookup/doi/10.1093/pasj/psw053. doi:10.1093/pasj/psw053.

[92] S. R. Slattery, Mesh-free data transfer algorithms for partitioned multiphysics problems: Conservation, accuracy, and parallelism, Journal of Computational Physics 307 (2016) 164–188. URL: http://www.sciencedirect.com/science/article/pii/S0021999115008037. doi:10.1016/j.jcp.2015.11.055.

[93] A. Schäfer, D. Fey, LibGeoDecomp: A Grid-Enabled Library for Geometric Decomposition Codes, in: A. Lastovetsky, T. Kechadi, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 285–294.

[94] M. Frigo, S. Johnson, The design and implementation of FFTW3, Proceedings of the IEEE 93 (2005) 216 – 231. doi:10.1109/jproc.2004.840301.

[95] H. J. Wilson, Stokes flow past three spheres, Journal of Computational Physics 245 (2013) 302–316. URL: http://www.sciencedirect.com/science/article/pii/S0021999113001988. doi:10.1016/j.jcp.2013.03.020.

[96] A. Sierou, J. F. Brady, Rheology and microstructure in concentrated noncolloidal suspensions, Journal of Rheology 46 (2002) 1031–1056. URL: http://sor.scitation.org/doi/abs/10.1122/1.1501925. doi:10.1122/1.1501925.

[97] A. M. Fiore, J. W. Swan, Fast Stokesian dynamics, Journal of Fluid Mechanics 878 (2019) 544–597. doi:10.1017/jfm.2019.640.

[98] É. Guazzelli, Sedimentation of small particles: how can such a simple problem be so difficult?, Comptes Rendus Mécanique 334 (2006) 539–544. doi:10.1016/j.crme.2006.07.009.

[99] J. Park, B. Metzger, É. Guazzelli, J. E. Butler, A cloud of rigid fibres sedimenting in a viscous fluid, Journal of Fluid Mechanics 648 (2010) 351–362. doi:10.1017/S0022112009993909.

[100] C. Dombrowski, L. Cisneros, S. Chatkaew, R. E. Goldstein, J. O. Kessler, Self-concentration and large-scale coherence in bacterial dynamics, Physical review letters 93 (2004) 098103.

[101] D. Saintillan, M. J. Shelley, Instabilities and Pattern Formation in Active Particle Suspensions: Kinetic Theory and Continuum Simulations, Physical Review Letters 100 (2008) 178103. doi:10.1103/PhysRevLett.100.178103, 00248.

[102] T. Sanchez, D. T. Chen, S. J. DeCamp, M. Heymann, Z. Dogic, Spontaneous motion in hierarchically assembled active matter, Nature 491 (2012) 431.

[103] M. K. Kuimova, Mapping viscosity in cells using molecular rotors, Physical Chemistry Chemical Physics 14 (2012) 12671–12686. doi:10.1039/C2CP41674C.

[104] R. Krasny, Desingularization of periodic vortex sheet roll-up, Journal of Computational Physics 65 (1986) 292–313.

[105] G. Baker, M. Shelley, On the connection between thin vortex layers and vortex sheets, Journal of Fluid Mechanics 215 (1990) 161–194.

[106] E. Lushi, P. M. Vlahovska, Periodic and Chaotic Orbits of Plane-Confined Micro-rotors in Creeping Flows, Journal of Nonlinear Science 25 (2015) 1111–1123. doi:10.1007/s00332-015-9254-9.

[107] D. Robinson, L. Feng, J. Nocedal, J. Pang, Subspace Accelerated Matrix Splitting Algorithms for Asymmetric and Symmetric Linear Complementarity Problems, SIAM Journal on Optimization 23 (2013) 1371–1397. URL: http://epubs.siam.org/doi/abs/10.1137/110845094. doi:10.1137/110845094.

[108] S. Y. Reigh, L. Zhu, F. Gallaire, E. Lauga, Swimming with a cage: Low-Reynolds-number locomotion inside a droplet, Soft Matter 13 (2017) 3161–3173. doi:10.1039/C6SM01636G.

[109] R. Hartmann, P. K. Singh, P. Pearce, R. Mok, B. Song, F. Díaz-Pascual, J. Dunkel, K. Drescher, Emergence of three-dimensional order and structure in growing biofilms, Nature Physics (2018) 1. doi:10.1038/s41567-018-0356-9.

[110] E. W. Weisstein, Complete elliptic integral of the first kind, http://mathworld.wolfram.com/CompleteEllipticIntegraloftheFirstKind.html, ????.

[111] E. W. Weisstein, Complete elliptic integral of the second kind, http://mathworld.wolfram.com/CompleteEllipticIntegraloftheSecondKind.html, ????.

[112] NIST digital library of mathematical functions, https://dlmf.nist.gov/, 2019.