Fast algorithms for large dense matrices with applications to biofluids*

Minghao W. Rostami[1]

Department of Mathematics, Syracuse University, 215 Carnegie Bldg., Syracuse, NY 13244.

Sarah D. Olson[2]

Department of Mathematical Sciences, Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA 01609, USA.

Abstract

Numerical simulation of biofluids entails solving equations of fluid-structure interactions. At zero Reynolds number, solvers such as the Method of Regularized Stokeslets (MRS) give rise to large and dense matrices in practical applications where the number of structures immersed in the fluid is large. Building on previous work for an unbounded fluid domain, we first extend the Kernel-Independent Fast Multipole Method (KIFMM) for MRS to compute the matrix-vector products for the fluid flow induced by point forces above a stationary wall. In this case, the use of a regularized image system introduces additional terms to the solution which cause the matrix-vector multiplication to be quite challenging. In addition, we study the case where a linear system needs to be solved for the unknown forces that structures with known velocities exert on the fluid. Our main contribution is proposing several preconditioning techniques for the matrices associated with a few variants of MRS, including the case where a force-free, torque-free condition is imposed. They take advantage of the data-sparsity of FMM matrices as well as properties of Krylov subspaces. Our approach is memory efficient, capable of handling non-uniformly distributed structures and applicable to all FMM matrices. It enables efficient computation of the flow field surrounding a large group of dynamic micro-structures; in particular, we study the effects of fluid mixing caused

*May 26th, 2019.

[1]Corresponding author. The work of M. W. Rostami (mwrostam@syr.edu) was supported in part by the Simons Foundation under award 527247, the Oak Ridge Associated Universities under a Ralph E. Powe Junior Faculty Enhancement Award, and the National Science Foundation under grant DMS-1818833.

[2]The work of S. D. Olson (sdolson@wpi.edu) was supported in part by the National Science Foundation under grant DMS-1455270 and the Fulbright Research Scholar Program.

by the periodic beating of a dense carpet of lung cilia.

Keywords: preconditioner, GMRES, Krylov subspace recycling, kernel-independent fast multipole method, regularized Stokeslets, cilia

---

1. Introduction

At high enough density, dynamic micro-structures such as bacteria [1–4], cilia, sperm [5, 6], algae, and microtubules [7, 8] exhibit remarkable collective motions which bear significant biological implications. For example, sperm swim both competitively and collaboratively to reach the egg, and cilia in the airways beat collectively to propel mucus and foreign particles out of the lung. These phenomena can be modeled by several methods, such as the immersed boundary method [9], the boundary integral equation method [10], Stokesian dynamics [11–13] the Method of Regularized Stokeslets (MRS) [14, 15] and through the use of the Rotne-Prager-Yamakawa (RPY) tensor [16, 17]. At zero Reynolds number, the fluid flow is linearly related to the force via a mobility tensor; the MRS utilizes a tensor based on the Stokeslet or single layer potential whereas the RPY tensor is based on both the Stokeslet and Stresslet. Both MRS and RPY are regularized methods for solving the incompressible Stokes equations; they have the advantage of not requiring an Eulerian grid for the entire fluid domain but the computations required by both methods entail the use of dense matrices. The size of the linear system grows and becomes progressively more difficult to work with as the number of point forces associated with the micro-structures increases. The main goal of this paper is to develop efficient methods for solving large-scale linear systems that involve the aforementioned coefficient matrices, which are large and dense, and are generally determined by particular kernel functions.

Two "ingredients" are essential in the efficient solution of the linear systems of interest. One is a fast algorithm for matrix-vector multiplications to determine the resulting fluid flow when the forces are known. The second ingredient is an effective preconditioner when solving for the forces in the case of known velocities. The Fast Multipole Method (FMM) involves an analytic factorization and can be utilized for fast matrix-vector products [18–22]; this method has been extended to matrices arising from the RPY tensor and the MRS, reducing the cost of computing a matrix-vector product from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ for $N \times N$ matrices. The standard Barnes-Hut treecode also breaks up the calculations into a near and far field calculation, resulting in a method that is $\mathcal{O}(N \log N)$ [23–25], where Taylor series or Lagrange barycentric interpolation could be used to represent the kernels [26, 27]. In particular, due to the difficulties with factorizing certain kernel functions, we have implemented the Kernel-Independent FMM (KIFMM) [28, 29] in [30] for the MRS. One limitation of [30] is that only unbounded fluid domains were considered.

It is well-known that surface interactions play an important role in the swimming motion of

micro-organisms [31]. The interactions between a small number of sperm and solid or elastic walls have been studied in [6, 32–38]. The collective behavior of micro-organisms in geometric confinement has been investigated by many papers such as [4–6, 39–44]. Most models that account for the flow above a stationary, planar wall have utilized an image system that cleverly places image points below the wall to enforce the no-slip condition at the wall [45]. Recently, the KIFMM has been extended for the case of flow above a no-slip wall in both non-periodic and periodic domains [46, 47]. However, the image system used different kernel functions [48] than the ones commonly used in MRS and the study did not focus on the case of solving for the unknown forces. Therefore, in this paper, we further extend the KIFMM to domains with a stationary, planar wall within the framework of MRS using the common image system of Blake [45] and then focus on solving the linear systems involving KIFMM matrices.

We emphasize that mathematically, FMM is equivalent to approximating the original matrix with a hierarchical matrix (or an $\mathcal{H}$-matrix) with nested bases, known as the $\mathcal{H}^2$-matrix (see [49, Section 2.11], [50, Chapter 8], and [51–56]). A hierarchical matrix is, roughly speaking, a matrix that can be partitioned into a hierarchy of smaller blocks, the vast majority of which are low-rank (see [50, Section 6.1]). It is therefore "data-sparse" and allows for efficient storage and matrix-vector multiplication with linear or almost linear complexity (see [50, Chapter 6] and [57–59]). In addition, fast direct solvers have been developed for various categories of $\mathcal{H}$-matrices including the Hierarchically Off-Diagonal Low-Rank (HODLR) matrices [60, 61], the Hierarchically SemiSeparable (HSS) matrices [62–67], and the $\mathcal{H}^2$-matrices [68, 69]. Direct solvers enjoy many advantages such as linear or almost linear complexity; they are often more efficient when used as preconditioners for an iterative solver. However, constructing these preconditioners can be expensive both computationally and storage-wise. Alternatively, many papers [70–75] use the sparse approximate inverse preconditioner [76, Section 10.5], which refers to a sparse approximation to the inverse of the coefficient matrix. As the user can specify the sparsity pattern of the preconditioner, storing it is easy; however, computing it requires the solution of N least-squares problems for an $N \times N$ matrix and can still be costly. The overhead incurred by both veins of methods is mitigated in applications where a sequence of linear systems with the *same* coefficient matrix and multiple right-hand sides need to be solved. Unfortunately, in many models for biofluids, we wish to understand the movement of dynamic micro-structures in time. This results in a *sequence* of linear systems with *different* coefficient matrices, making the cost for these methods prohibitive.

The preconditioner that we propose to use is the matrix associated with a version of KIFMM that is less accurate than the version applied to compute the matrix-vector products. This idea is not new; for example, it has been used in [73] for FMM matrices. It gives rise to a sequence of challenging subsidiary linear systems. By exploiting properties of Krylov subspaces, we propose a few techniques that both expedite each individual solve and reduce the number of solves needed. Compared to the inverse FMM preconditioner [68, 69], our approach is highly economical in terms of memory as there is no need to explicitly form the preconditioner; its implementation is also insensitive to whether the distribution of the swimmers is uniform.

Although our focus is on dense matrices, for completeness, we point out that FMM-based preconditioners have recently been developed for the sparse matrices arising from spatial discretization of elliptic partial differential equations [77, 78]. They are competitive alternatives to the conventional preconditioners for these matrices such as multigrid.

The rest of the paper is organized as follows. In Section 2, we describe the structure of the matrices arising from the MRS in various settings and provide a brief review of KIFMM. In Section 3, we explain in detail how we plan to solve the linear systems involving large-scale MRS matrices. Numerical results of the proposed methods are presented in Section 4, where several types of dynamic micro-structures are considered. We conclude the paper with a few remarks in Section 5.

2. Matrix-vector multiplication

In Section 2.1, we first review the Method of Regularized Stokeslets (MRS) with emphasis on the structure of the matrices associated with it. As we shall see, they are dense and will grow in size as the number of point forces increases. In Section 2.2, we summarize the Kernel-Independent Fast Multipole Method (KIFMM) for performing the matrix-vector multiplication efficiently.

Throughout the rest of the paper, the fluid domain is three-dimensional and either unbounded or half-bounded by a stationary, planar wall; and $\|\cdot\|$ denotes the standard Euclidean norm (2 norm).

2.1. The Method of Regularized Stokeslets (MRS)

The MRS is a widely applicable method to simulate zero Reynolds number flows driven by forces from active or moving structures. Micro-swimmers are often represented in a space dimension one lower than the fluid, which generally leads to singular or nearly singular kernel fluid velocity evaluations when using fundamental solutions. Integrable kernels and finite velocity fields on the structures are obtained in the MRS by first regularizing the forces [14, 15]. Forces can be regularized with support regions mainly corresponding to the width or size of the structure, which has been shown to match well with experiments [79–81] and capture dynamics as well (or better) than resistive force theory and slender body theory [80, 82]. Additionally, the MRS can handle structures of different sizes and shapes via the choice of different regularization functions, discretizations, and force models to capture flexible (elastic) structures.

Let a group of micro-swimmers or structures be represented by N points located at $\{\mathbf{x}_j\}_{j=1}^N$. For instance, they can be grid points corresponding to a Lagrangian discretization of the structures or can each be an individual swimmer. Suppose that these points exert forces $\{\mathbf{f}_j\}_{j=1}^N$ on the surrounding fluid and move at the local fluid velocities (no-slip condition). Since the swimmers are at the scale of zero Reynolds number where inertia is negligible and viscous effects dominate, the fluid dynamics can be modeled by the incompressible Stokes equations:

$$\begin{aligned} -\mu\Delta\mathbf{u} + \nabla p &= \sum_{j=1}^{N} \mathbf{f}_j \delta(r_j), \\ \nabla\cdot\mathbf{u} &= 0, \end{aligned} \tag{1}$$

where $r_j = \|\mathbf{r}_j\| = \|\mathbf{x} - \mathbf{x}_j\|$, $\delta(r)$ is the Dirac delta distribution, μ is the viscosity, $p = p(\mathbf{x})$ is the pressure and $\mathbf{u} = \mathbf{u}(\mathbf{x})$ is the velocity at $\mathbf{x} = [x, y, z]^T$. In free-space, the solution $\mathbf{u}(\mathbf{x})$ to (1) can be written as $\sum_{j=1}^{N} S(\mathbf{x}, \mathbf{x}_j)\mathbf{f}_j$, where $S(\mathbf{x}, \mathbf{x}_j) : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ $(j = 1, \ldots, N)$ is defined at every $\mathbf{x}$ *except for* $\mathbf{x}_j$. In order to evaluate the velocities of the micro-organisms, which is necessary for updating their locations in a time-dependent simulation, MRS [14, 15] replaces $\delta(r)$ in (1) with a smooth, radially symmetric "blob" function $\psi_\varepsilon(r)$, where $\varepsilon > 0$ is a parameter that controls the spreading of every point force $\mathbf{f}_j$. An example of such a function is

$$\psi_\varepsilon(r) = \frac{15\varepsilon^4}{8\pi(\varepsilon^2 + r^2)^{7/2}}. \tag{2}$$

Although (2) has an infinite support, it spreads the majority of $\mathbf{f}_j$ within the ball centered at $\mathbf{x}_j$ with radius ε. Thus, in our applications, the parameter ε can be viewed as the radius or thickness of the micro-organisms. The solution to (1) with the regularized forcing terms $\{\mathbf{f}_j\psi_\varepsilon(r_j)\}_{j=1}^N$ is $\mathbf{u}(\mathbf{x}) = \sum_{j=1}^{N} S_\varepsilon(\mathbf{x}, \mathbf{x}_j)\mathbf{f}_j$, where the regularized Stokeslet $S_\varepsilon(\mathbf{x}, \mathbf{x}_j) : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ $(j = 1, \ldots, N)$ is defined at every $\mathbf{x}$ *including* $\mathbf{x}_j$. As an alternative to (2), we can also use a compactly supported blob function [83].

Two additional cases of MRS will be considered as well. In the free-space case, we can also account for point torques, capturing the twisting or rotation of micro-organisms. Hence, in addition to the point forces $\{\mathbf{f}_j\}_{j=1}^N$, we also need to include the torques $\{\mathbf{n}_j\}_{j=1}^N$ at $\{\mathbf{x}_j\}_{j=1}^N$ in the right hand side of (1) as $\sum_{j=1}^{N} \nabla \times \mathbf{n}_j\psi_\varepsilon(r_j)$ for the regularized case [84–87]. As shown in [87], the solution to (1) with both the regularized forces and torques incorporated is

$$\begin{bmatrix} \mathbf{u}(\mathbf{x}) \\ \mathbf{w}(\mathbf{x}) \end{bmatrix} = \sum_{j=1}^{N} K_\varepsilon(\mathbf{x}, \mathbf{x}_j) \begin{bmatrix} \mathbf{f}_j \\ \mathbf{n}_j \end{bmatrix},$$

where $\mathbf{w}(\mathbf{x})$ is the angular velocity at $\mathbf{x}$ and $K_\varepsilon(\mathbf{x}, \mathbf{x}_j) : \mathbb{R}^3 \to \mathbb{R}^{6\times 6}$ is defined at every $\mathbf{x}$.

We will also consider the case of only point forces above a stationary, planar wall satisfying a no-slip boundary condition. The solution to the regularized version of (1) with $\sum_{j=1}^{N} \mathbf{f}_j\psi_\varepsilon(r_j)$ on the right hand side in a domain $\mathcal{D}_W$ half bounded by a wall was derived in [88] using the method of images of Blake [45]; it can be written as

$$\mathbf{u}(\mathbf{x}) = \sum_{j=1}^{N} W_\varepsilon(\mathbf{x}, \mathbf{x}_j)\mathbf{f}_j, \tag{3}$$

where $W_\varepsilon(\mathbf{x}, \mathbf{x}_j) : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ is defined at every $\mathbf{x} \in \mathcal{D}_W$.

In each of the three cases, the point forces (and torques) and the velocities at the N points due to their pairwise hydrodynamic interactions satisfy a linear system $\boldsymbol{\mathcal{U}} = A\boldsymbol{\mathcal{G}}$. Here, A is determined by a particular kernel function (S_ε, W_ε or K_ε) and is a $dN \times dN$ matrix consisting of $d \times d$ blocks ($S_\varepsilon(\mathbf{x}_i, \mathbf{x}_j)$, $W_\varepsilon(\mathbf{x}_i, \mathbf{x}_j)$ or $K_\varepsilon(\mathbf{x}_i, \mathbf{x}_j)$), where $d = 3$ if only forces are considered or $d = 6$ if torques are included as well. Let $\boldsymbol{\mathcal{G}}$ denote the $dN \times 1$ vector obtained by concatenating $\{\mathbf{f}_j\}_{j=1}^N$ or

$$\left\{ \begin{bmatrix} \mathbf{f}_j \\ \mathbf{n}_j \end{bmatrix} \right\}_{j=1}^{N}.$$

For $\mathbf{u}_j$ ($\mathbf{w}_j$) denoting the linear (angular) velocity at $\mathbf{x}_j$, we can define the $dN \times 1$ vector $\boldsymbol{\mathcal{U}}$ in a similar way by replacing $\mathbf{f}_j$ (and $\mathbf{n}_j$) in $\boldsymbol{\mathcal{G}}$ with $\mathbf{u}_j$ (and $\mathbf{w}_j$, respectively). When $\boldsymbol{\mathcal{G}}$ is known, computing the velocity $\boldsymbol{\mathcal{U}}$ is equivalent to evaluating a matrix-vector product. On the other hand, when $\boldsymbol{\mathcal{U}}$ is known and $\boldsymbol{\mathcal{G}}$ is wanted, we have to solve the linear system $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ instead.

We now take a closer look at the $d \times d$ kernel functions $S_\varepsilon(\mathbf{x}, \mathbf{y})$, $W_\varepsilon(\mathbf{x}, \mathbf{y})$ and $K_\varepsilon(\mathbf{x}, \mathbf{y})$. For any $\mathbf{x}$, $\mathbf{y}$ in the domain, let $\mathbf{r} = \mathbf{x} - \mathbf{y}$, $r = \|\mathbf{r}\|$, I_n denote the $n \times n$ identity matrix, and

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{4}$$

where v_k ($k = 1$, 2 or 3) is the kth component of a vector $\mathbf{v}$ in $\mathbb{R}^3$. (For any $\mathbf{u} \in \mathbb{R}^3$, the matrix-vector product $[\mathbf{v}]_\times \cdot \mathbf{u}$ is equivalent to the cross product $\mathbf{v} \times \mathbf{u}$.) Then, we can write the kernel functions for the free-space case of forces only or forces and torques as

$$S_\varepsilon(\mathbf{x}, \mathbf{y}) = \frac{1}{\mu}\left(H_1(r)I_3 + H_2(r)\mathbf{r}\mathbf{r}^T\right), \tag{5}$$

$$K_\varepsilon(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} S_\varepsilon(\mathbf{x}, \mathbf{y}) & \frac{1}{2\mu}Q(r)[\mathbf{r}]_\times^T \\ \frac{1}{2\mu}Q(r)[\mathbf{r}]_\times^T & \frac{1}{4\mu}\left(D_1(r)I_3 + \frac{1}{4}D_2(r)\mathbf{r}\mathbf{r}^T\right) \end{bmatrix}, \tag{6}$$

respectively, where $H_1(r)$, $H_2(r)$, $Q(r)$, $D_1(r)$ and $D_2(r)$ are smooth, radially symmetric functions. We note that the terms involving H_1, H_2 correspond to the regularized Stokeslet and account for the contributions of the point forces on the fluid flow, the terms involving Q are regularized rotlets due to the point forces and torques, and the terms involving D_1, D_2 are regularized dipoles due to the point torques. If a wall is located at $z = 0$ and the fluid domain is $\mathcal{D}_W = \{[x, y, z]^T \mid z \geq 0\}$, then

$$
\begin{aligned}
W_\varepsilon(\mathbf{x},\mathbf{y}) = S_\varepsilon(\mathbf{x},\mathbf{y}) - S_\varepsilon\left(\mathbf{x},\mathbf{y}^{\mathrm{im}}\right) - \frac{h^2}{\mu}\left[D_1^\dagger(r^{\mathrm{im}})I_3 - D_2^\dagger\left(r^{\mathrm{im}}\right)\mathbf{r}^{\mathrm{im}}\left(\mathbf{r}^{\mathrm{im}}\right)^T\right]\left(2\mathbf{e}_3\mathbf{e}_3^T - I_3\right) \\
+ \frac{2h}{\mu}\left[H_2\left(r^{\mathrm{im}}\right)\mathbf{r}^{\mathrm{im}}\mathbf{e}_3^T + H_2\left(r^{\mathrm{im}}\right)\left(\mathbf{r}^{\mathrm{im}}\cdot\mathbf{e}_3\right)I_3 + \frac{H_1'\left(r^{\mathrm{im}}\right)}{r^{\mathrm{im}}}\mathbf{e}_3\left(\mathbf{r}^{\mathrm{im}}\right)^T\right. \\
\left. + \frac{H_2'\left(r^{\mathrm{im}}\right)}{r^{\mathrm{im}}}\left(\mathbf{r}^{\mathrm{im}}\cdot\mathbf{e}_3\right)\mathbf{r}^{\mathrm{im}}\left(\mathbf{r}^{\mathrm{im}}\right)^T\right]\left(2\mathbf{e}_3\mathbf{e}_3^T - I_3\right) \\
+ \frac{2h}{\mu}\left(\frac{H_1'\left(r^{\mathrm{im}}\right)}{r^{\mathrm{im}}} + H_2\left(r^{\mathrm{im}}\right)\right)[\mathbf{y}^{\mathrm{im}}]_\times[\mathbf{e}_3]_\times,
\end{aligned}
\tag{7}
$$

where $\mathbf{y}^{\mathrm{im}}$ is the image of $\mathbf{y}$ with respect to the wall, $h > 0$ is the distance between $\mathbf{y}$ (or $\mathbf{y}^{\mathrm{im}}$) and the wall, $\mathbf{r}^{\mathrm{im}} = \mathbf{x} - \mathbf{y}^{\mathrm{im}}$, $r^{\mathrm{im}} = \|\mathbf{r}^{\mathrm{im}}\|$, $\mathbf{e}_3 = [0,0,1]^T$, prime denotes a derivative with respect to r, and all of the functions of r and r^{im} are again smooth, radially symmetric functions. The terms in (7) are similar to those in (5), (6) except that many are now with respect to the image points and are included to cancel the flow at the wall.

We note that the precise expressions for the functions $H_1(r)$, $H_2(r)$, $Q(r)$, $D_1(r)$, $D_2(r)$, $D_1^\dagger(r)$ and $D_2^\dagger(r)$ in (5)-(7) are dictated by the blob function(s) used. In the free-space case, only one blob function $\psi_\varepsilon(r)$ is needed and the choice (2) leads to the following:

$$
\begin{aligned}
H_1(r) &= \frac{2\varepsilon^2 + r^2}{8\pi(r^2+\varepsilon^2)^{3/2}},\ H_2(r) = \frac{1}{8\pi(r^2+\varepsilon^2)^{3/2}}, \\
Q(r) &= \frac{5\varepsilon^2 + 2r^2}{8\pi(r^2+\varepsilon^2)^{5/2}},\ D_1(r) = \frac{10\varepsilon^4 - 7\varepsilon^2 r^2 - 2r^4}{8\pi(r^2+\varepsilon^2)^{7/2}},\ D_2(r) = \frac{21\varepsilon^2 + 6r^2}{8\pi(r^2+\varepsilon^2)^{7/2}}.
\end{aligned}
\tag{8}
$$

For the case of flow bounded above a stationary, planar wall, the use of a second or companion blob function is necessary to impose the no-slip condition at the wall [88, 89]. Utilizing $\phi_\varepsilon(r) = 3\varepsilon^2/(4\pi(r^2+\varepsilon^2))^{5/2}$, we obtain the expressions for $D_1^\dagger$ and $D_2^\dagger$ as:

$$
D_1^\dagger(r) = \frac{-2\varepsilon^2 + r^2}{4\pi(r^2+\varepsilon^2)^{5/2}},\ D_2^\dagger(r) = -\frac{3}{4\pi(r^2+\varepsilon^2)^{5/2}}.
\tag{9}
$$

We conclude our review of MRS with a summary of the three kernel functions.

(i) $S_\varepsilon(\mathbf{x},\mathbf{y})$, $W_\varepsilon(\mathbf{x},\mathbf{y}) \in \mathbb{R}^{3\times 3}$ arise from the inclusion of regularized forces in (1) for different domains, whereas $K_\varepsilon(\mathbf{x},\mathbf{y}) \in \mathbb{R}^{6\times 6}$ also accounts for regularized torques.

(ii) $S_\varepsilon(\mathbf{x},\mathbf{y})$, $K_\varepsilon(\mathbf{x},\mathbf{y})$ correspond to an unbounded domain and $W_\varepsilon(\mathbf{x},\mathbf{y})$ to a half-bounded one.

(iii) $S_\varepsilon(\mathbf{x},\mathbf{y}) = S_\varepsilon(\mathbf{y},\mathbf{x})$, whereas in general, $K_\varepsilon(\mathbf{x},\mathbf{y}) \neq K_\varepsilon(\mathbf{y},\mathbf{x})$, $W_\varepsilon(\mathbf{x},\mathbf{y}) \neq W_\varepsilon(\mathbf{y},\mathbf{x})$. Additionally, $K_\varepsilon(\mathbf{x},\mathbf{y}) = (K_\varepsilon(\mathbf{y},\mathbf{x}))^T$ and $W_\varepsilon(\mathbf{x},\mathbf{y}) = (W_\varepsilon(\mathbf{y},\mathbf{x}))^T$.

(iv) $S_\varepsilon(\mathbf{x},\mathbf{y})$ and $K_\varepsilon(\mathbf{x},\mathbf{y})$ are completely determined by $\mathbf{x}-\mathbf{y}$, i.e., they are translation invariant; however, this is not true for $W_\varepsilon(\mathbf{x},\mathbf{y})$.

(v) For a fixed $\mathbf{y}$, the entries of $S_\varepsilon(\mathbf{x},\mathbf{y})$, $W_\varepsilon(\mathbf{x},\mathbf{y})$ and $K_\varepsilon(\mathbf{x},\mathbf{y})$ approach zero as the distance between $\mathbf{x}$ and $\mathbf{y}$ goes to infinity.

2.2. The Kernel-Independent Fast Multipole Method (KIFMM)

The naïve or direct computation of the matrix-vector product $A\boldsymbol{\mathcal{G}}$ to determine $\boldsymbol{\mathcal{U}}$ entails $\mathcal{O}(N^2)$ operations. In [30], by using KIFMM [28, 29], we are able to reduce this complexity to $\mathcal{O}(N)$ for the matrices generated by S_ε and K_ε. In order to study collective motion of micro-structures near a surface, we extend KIFMM to the matrices generated by W_ε in the current work. An outline of KIFMM is provided below where S_ε is the underlying kernel; due to its kernel-independent nature, the algorithm is similar for other kernels such as K_ε and W_ε.

FMM [18–22] is an $\mathcal{O}(N)$ method for calculating the interactions among N particles. In our applications, the N particles represent a group of hydrodynamically interacting micro-structures that are actively generating forces. A critical assumption of FMM is that the strength of the interaction between two particles decays as the distance between them grows. In this method, the particles are grouped into clusters based on their physical proximity; the interactions between a cluster and the "nearby" clusters are strong and calculated directly, whereas its interactions with the clusters in the "far field" are weak and thus only approximated.

Hierarchical partition of the computational domain. Suppose that all the particles $\{\mathbf{x}_j\}_{j=1}^N$ are contained in a box $\mathfrak{D}$ and s is a positive integer that does not vary with N. By uniformly subdividing each existing box into eight "child boxes", FMM creates an octant tree of boxes where $\mathfrak{D}$ is the root, the boxes on each level of the tree constitute a uniform partition of $\mathfrak{D}$, and each leaf box (i.e., the smallest box) contains no more than s particles. For each box $\mathfrak{B}$, the neighborhood of $\mathfrak{B}$, denoted by $\mathfrak{N}(\mathfrak{B})$, is the union of all the boxes on the same level in the tree as $\mathfrak{B}$ that share at least one boundary point with $\mathfrak{B}$; the far field of $\mathfrak{B}$, denoted by $\mathfrak{F}(\mathfrak{B})$, is the complement of $\mathfrak{N}(\mathfrak{B})$ in $\mathfrak{D}$.

Multipole and local expansions. In KIFMM, a multipole expansion associated with a box $\mathfrak{B}$ has the form

$$\sum_{j=1}^{n_q} \omega_j S_\varepsilon(\mathbf{x},\mathbf{q}_j)\mathbf{g}_j; \tag{10}$$

for any $\mathbf{x}$ in $\mathfrak{F}(\mathfrak{B})$, it estimates the part of $\mathbf{u}(\mathbf{x})$ induced by the particles in $\mathfrak{B}$. In (10), $\{\mathbf{q}_j\}_{j=1}^{n_q}$ are quadrature points serving as proxies of the particles in $\mathfrak{B}$, $\{\omega_j\}_{j=1}^{n_q}$ are the weights in an underlying quadrature rule, and $\{\mathbf{g}_j\}_{j=1}^{n_q}$ are properly chosen, fictitious (non-physical) forces at $\{\mathbf{q}_j\}_{j=1}^{n_q}$. A local expansion associated with $\mathfrak{B}$ admits a similar form, i.e.,

$$\sum_{j=1}^{n_q} \widetilde{\omega}_j S_\varepsilon(\mathbf{x}, \widetilde{\mathbf{q}}_j)\widetilde{\mathbf{g}}_j; \tag{11}$$

for any $\mathbf{x}$ in $\mathfrak{B}$, it estimates the part of $\mathbf{u}(\mathbf{x})$ induced by the particles in $\mathfrak{F}(\mathfrak{B})$. In (11), $\{\widetilde{\mathbf{q}}_j\}_{j=1}^{n_q}$ are quadrature points serving as proxies of the particles in $\mathfrak{F}(\mathfrak{B})$, $\{\widetilde{\omega}_j\}_{j=1}^{n_q}$ are again the weights in a quadrature rule, and $\{\widetilde{\mathbf{g}}_j\}_{j=1}^{n_q}$ are fictitious forces at $\{\widetilde{\mathbf{q}}_j\}_{j=1}^{n_q}$. (For the locations of both sets of quadrature points relative to $\mathfrak{B}$, see Fig. 10 in the Appendix.) We first compute $\{\mathbf{g}_j\}_{j=1}^{n_q}$ for each box while traversing the octant tree from the leaves to the root; utilizing the multipole expansions, we then find $\{\widetilde{\mathbf{g}}_j\}_{j=1}^{n_q}$ for each box while traversing the tree in the reverse order. The main cost of computing $\{\mathbf{g}_j\}_{j=1}^{n_q}$ or $\{\widetilde{\mathbf{g}}_j\}_{j=1}^{n_q}$ for a box is the numerical solution of an integral equation, which boils down to solving a linear system of $3n_q$ equations and $3n_q$ unknowns.

Evaluation. For every $\mathbf{x}$ in $\mathfrak{D}$, we first identify the leaf box $\mathfrak{B}$ containing $\mathbf{x}$. Then we directly calculate the part of $\mathbf{u}(\mathbf{x})$ induced by the particles in $\mathfrak{N}(\mathfrak{B})$ and estimate the other part of $\mathbf{u}(\mathbf{x})$ which is induced by the particles in $\mathfrak{F}(\mathfrak{B})$ using (11) instead.

We note that the main difference between the original FMM and KIFMM lies in how the multipole and local expansions are obtained. In the original FMM, they rely on an analytic expansion of the kernel function; KIFMM, on the other hand, does not require any analytic expansion and computes (10), (11) numerically. Due to the complexity of the three kernel functions shown in (5)-(8), how to implement the original FMM remains an open question.

A more algebraic interpretation of FMM is that it finds low-rank approximations to the submatrices in A corresponding to long-range interactions. The resulting approximation to A is an $\mathcal{H}$-matrix, which is "data-sparse" and allows for efficient storage and matrix-vector multiplication with linear or almost linear complexity (see [50, Chapter 6] and [57–59]). More specifically, the estimate to the matrix-vector product $A\boldsymbol{\mathcal{G}}$ produced by FMM is $A_{\mathcal{H}}\boldsymbol{\mathcal{G}}$, where $A_{\mathcal{H}}$ is an $\mathcal{H}$-matrix with nested bases known as the $\mathcal{H}^2$-matrix (see [49, Section 2.11], [50, Chapter 8], and [51–56]). Note that $A_{\mathcal{H}}$ is not explicitly formed in FMM.

The accuracy and runtimes of the KIFMM corresponding to various choices of quadrature points

are reported in the Appendix for S_ε, W_ε and K_ε.

3. Preconditioners

It is often hard to have a good understanding of the forces or torques exerted by the micro-swimmers, as direct measurements are extremely difficult to achieve. Instead, flow visualization techniques such as Particle Image Velocimetry (PIV) [90, 91] can be used to estimate velocities, based on which the forces (and torques) can be recovered. This requires solving the linear system $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ for $\boldsymbol{\mathcal{G}}$, whose computational complexity is $\mathcal{O}(N^3)$ if a direct method such as Gaussian elimination is applied. Once the forces (and torques) are found, the linear velocity (and angular velocity) at any point in the fluid can be calculated by MRS (see Section 2.1).

We will solve $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ using the preconditioned Generalized Minimal RESidual (GMRES) method [92]. This method first finds an "optimal" solution $\widehat{\boldsymbol{\mathcal{X}}}$ to $AP^{-1}\widehat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{U}}$ in a Krylov subspace

$$\mathcal{K}_m(AP^{-1}, \boldsymbol{\mathcal{U}}) = \text{span}\{\boldsymbol{\mathcal{U}}, AP^{-1}\boldsymbol{\mathcal{U}}, (AP^{-1})^2\boldsymbol{\mathcal{U}}, \cdots, (AP^{-1})^{m-1}\boldsymbol{\mathcal{U}}\} \tag{12}$$

where $P \in \mathbb{R}^{dN \times dN}$ is a preconditioner; and then it recovers an approximate solution to $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ as $\widehat{\boldsymbol{\mathcal{G}}} = P^{-1}\widehat{\boldsymbol{\mathcal{X}}}$. Ideally, the eigenvalues of AP^{-1} are highly clustered and the linear system $P\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ can be solved efficiently. In this work, the matrix-vector products involving A are estimated by KIFMM; consequently, the effective Krylov subspace is $\mathcal{K}_m(A_{\mathcal{H}}P^{-1}, \boldsymbol{\mathcal{U}})$, where $A_{\mathcal{H}}$ is the $\mathcal{H}^2$-matrix associated with KIFMM (see Section 2.2).

We consider two preconditioners: a sparse, crude preconditioner and a dense yet data-sparse, much more accurate preconditioner. Applying the second preconditioner gives rise to challenging subsidiary linear systems; in Section 3.2, we propose two techniques for expediting their solution. We emphasize that neither preconditioner needs to be formed explicitly. In Section 3.3, we also develop a preconditioner for the matrices formed by appending extra rows and columns to A. They emerge from problems where the swimmers exert zero net force and zero net torque.

3.1. The block-diagonal preconditioner

A sparse approximation to A can be obtained if we cut off most of the particle interactions. In particular, by allowing a particle to interact only with those contained in the same leaf box in

KIFMM, we get a block-diagonal approximation, denoted by $P_{\mathcal{D}}$, to A.[3] For example, let A be the 30000×30000 matrix that is generated by the kernel function S_ε and represents the pairwise interactions among 10000 random particles uniformly distributed in a cube $\mathfrak{D}$. Suppose that $\mathfrak{D}$ is uniformly partitioned twice in KIFMM. We display the sparsity pattern of $P_{\mathcal{D}}$ in the left panel of Fig. 1, where every small, dense block represents the interactions among the particles within one of the 64 leaf boxes.

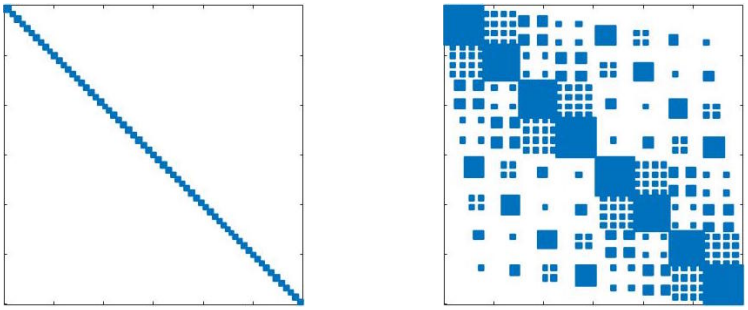


Figure 1: The sparsity patterns of $P_{\mathcal{D}}$ (left) and $P_{\mathcal{N}}$ (right), accounting for only interactions in the leaf cube and accounting for interactions in the leaf cube and its neighbors, respectively.

The block-diagonal matrix $P_{\mathcal{D}}$ is appealing as a preconditioner for the following two reasons: it captures some of the strongest interactions among the particles; and solving $P_{\mathcal{D}}\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ boils down to solving a small linear system for each leaf box, which is easy. If we also allow the interactions between a leaf cube and its neighbors, a more accurate sparse approximation to A, denoted by $P_{\mathcal{N}}$, can be obtained. The sparsity pattern of $P_{\mathcal{N}}$ is shown in the right panel of Fig. 1 for the same example considered earlier. However, due to the off-diagonal non-zero blocks in $P_{\mathcal{N}}$, the linear system $P_{\mathcal{N}}\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ is significantly more difficult to solve than $P_{\mathcal{D}}\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$. We note that block-diagonal preconditioners have also been used in [69, 93] for the large-scale, dense matrices generated by the RPY tensor and in [94] for those arising from a boundary integral formulation.

[3] Technically speaking, the particles in each leaf box must be assigned consecutive subscripts for this approximation to be block-diagonal. That is, if $\{\mathbf{x}_{j_i}\}_{i=1}^n$ are all the particles in the jth leaf box and $j_1 < j_2 < \cdots < j_n$, then $j_{i+1} = j_i + 1$ needs to be imposed for all $1 \le i \le n-1$. In practice, we do not form $P_{\mathcal{D}}$, so we assign subscripts to points on the structures in a way that makes sense for the modeling problem at hand, which may or may not correspond to consecutive subscripts in leaf boxes.

3.2. The data-sparser preconditioner

If the underlying quadrature rule remains fixed, the more quadrature points we use in the multipole expansion (10) and local expansion (11), the more accurate and computationally expensive KIFMM becomes. Let $A_{\mathcal{H}}$ correspond to the KIFMM when n_q quadrature points are used. The second preconditioner that we consider, denoted by $P_{\mathcal{H}}$, is the $\mathcal{H}^2$-matrix associated with the KIFMM when $n'_q < n_q$ quadrature points are used. Due to the smaller number of quadrature points, the blocks in $P_{\mathcal{H}}$ corresponding to long-range interactions have a lower rank, i.e., $P_{\mathcal{H}}$ is data-sparser than $A_{\mathcal{H}}$. Unlike the block-diagonal preconditioner $P_{\mathcal{D}}$, this preconditioner accounts for all the particle interactions and is thus a much more accurate preconditioner.

However, it is still challenging to solve the subsidiary linear systems whose coefficient matrix is $P_{\mathcal{H}}$. We propose to apply the GMRES preconditioned by $P_{\mathcal{D}}$ to them. This leads to a solver for $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ that has the following hierarchy: the outer iteration is the GMRES preconditioned by $P_{\mathcal{H}}$, and the inner iteration is the GMRES preconditioned by $P_{\mathcal{D}}$. It is not necessary to form $P_{\mathcal{H}}$: it can be multiplied to vectors efficiently using KIFMM. We propose the following two techniques for accelerating the solution of the subsidiary linear systems.

3.2.1. Recycling the Krylov subspaces

We need to solve a sequence of linear systems that share the same coefficient matrix $P_{\mathcal{H}}$ using GMRES. This is an ideal scenario where Krylov subspace recycling [95] can be applied to reduce computational cost. The basic idea is the following: instead of building a Krylov subspace from scratch for each linear system in the sequence, we retain some vectors from an existing Krylov subspace and construct a new Krylov subspace based on them.

Assume that we have already solved the kth subsidiary linear system $P_{\mathcal{H}}\boldsymbol{\mathcal{G}}_k = \boldsymbol{\mathcal{U}}_k$ using GMRES preconditioned by $P_{\mathcal{D}}$. If GMRES converges in ℓ steps, then we have at our disposal an Arnoldi decomposition

$$P_{\mathcal{H}}P_{\mathcal{D}}^{-1}V_\ell = V_\ell H_\ell + h_{\ell+1,\ell}\mathbf{v}_{\ell+1}\mathbf{e}_\ell^T = [V_\ell\ \mathbf{v}_{\ell+1}]\begin{bmatrix} H_\ell \\ h_{\ell+1,\ell}\mathbf{e}_\ell^T \end{bmatrix} = V_{\ell+1}\underline{H_\ell}, \tag{13}$$

where $H_\ell \in \mathbb{R}^{\ell\times\ell}$ is an upper Hessenberg matrix, $\mathbf{e}_\ell = [0,\cdots,0,1]^T \in \mathbb{R}^{\ell\times 1}$, and the columns of V_ℓ constitute an orthonormal basis for the Krylov subspace $\mathcal{K}_\ell(P_{\mathcal{H}}P_{\mathcal{D}}^{-1},\boldsymbol{\mathcal{U}}_k)$. Since the next linear system $P_{\mathcal{H}}\boldsymbol{\mathcal{G}}_{k+1} = \boldsymbol{\mathcal{U}}_{k+1}$ shares the same coefficient matrix, it is reasonable to further exploit $\mathcal{K}_\ell(P_{\mathcal{H}}P_{\mathcal{D}}^{-1},\boldsymbol{\mathcal{U}}_k)$. For example, we can retain the $1 \le \widetilde{\ell} \le \ell$ harmonic Ritz vectors of $P_{\mathcal{H}}P_{\mathcal{D}}^{-1}$

associated with its $\widetilde{\ell}$ dominant harmonic Ritz values, which tend to approximate the eigenvectors of $P_{\mathcal{H}}P_{\mathcal{D}}^{-1}$ corresponding to its $\widetilde{\ell}$ largest eigenvalues. Recall that a harmonic Ritz pair $(\theta, V_\ell \mathbf{z})$ is an approximate eigenpair, which satisfies the following Galerkin condition:

$$\left(P_{\mathcal{H}}P_{\mathcal{D}}^{-1}V_\ell\right)^T \left(P_{\mathcal{H}}P_{\mathcal{D}}^{-1}V_\ell \mathbf{z} - \theta V_\ell \mathbf{z}\right) = \mathbf{0}. \tag{14}$$

By (13) and (14), the harmonic Ritz vectors of $P_{\mathcal{H}}P_{\mathcal{D}}^{-1}$ in $\mathcal{K}_\ell(P_{\mathcal{H}}P_{\mathcal{D}}^{-1}, \boldsymbol{\mathcal{U}}_k)$ can be computed by solving an $\ell \times \ell$ generalized eigenvalue problem, namely,

$$\left(\underline{H_\ell}^T \underline{H_\ell}\right)\mathbf{z} = \theta \left(\underline{H_\ell}^T V_{\ell+1}^T V_\ell\right)\mathbf{z}.$$

When solving $P_{\mathcal{H}}\boldsymbol{\mathcal{G}}_{k+1} = \boldsymbol{\mathcal{U}}_{k+1}$, we keep extending the subspace spanned by the $\widetilde{\ell}$ recycled vectors until a sufficiently accurate solution is found.

3.2.2. Estimating the upper Hessenberg matrix

Krylov subspace recycling described in Section 3.2.1 aims to reduce the computational cost of each subsidiary solve after the first one has been performed. The technique proposed in this subsection reduces the number of such solves. Assume that solving $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ entails m iterations of GMRES preconditioned by $P_{\mathcal{H}}$. Then at the end of the solve, the Arnoldi decomposition

$$A_{\mathcal{H}}P_{\mathcal{H}}^{-1}V_m = V_m H_m + h_{m+1,m}\mathbf{v}_{m+1}\mathbf{e}_m^T = [V_m \ \mathbf{v}_{m+1}] \begin{bmatrix} H_m \\ h_{m+1,m}\mathbf{e}_m^T \end{bmatrix} = V_{m+1}\underline{H_m} \tag{15}$$

similar to (13) is available, and the approximate solution found is $\widehat{\boldsymbol{\mathcal{G}}}_m = \beta P_{\mathcal{H}}^{-1}(V_m \mathbf{y}_m)$ where $\beta = \|\boldsymbol{\mathcal{U}}\|$ and $\mathbf{y}_m$ is the least-squares solution to $\underline{H_m}\mathbf{y} = [1, 0, \cdots, 0]^T$.

In fact, thanks to the "nested" nature of the upper Hessenberg matrix, H_m can be written as

$$H_m = \left[\underline{H_{m-1}} \ \mathbf{b}\right]. \tag{16}$$

As a result, after $m-1$ GMRES iterations, the only "ingredients" that are missing for $\widehat{\boldsymbol{\mathcal{G}}}_m$ is the vector $\mathbf{b}$ in (16) and the number $h_{m+1,m}$ in (15). If we can somehow obtain good estimates to $\mathbf{b}$ and $h_{m+1,m}$, then we can compute an approximate solution close to $\widehat{\boldsymbol{\mathcal{G}}}_m$ in only $m-1$ GMRES iterations. A trivial yet crucial observation is that if $P_{\mathcal{H}} = A_{\mathcal{H}}$, then $H_m = V_m^T A_{\mathcal{H}} P_{\mathcal{H}}^{-1} V_m$ is simply the $m \times m$ identity matrix, in which case $\mathbf{b} = [0, \cdots, 0, 1]^T$ and $h_{m+1,m} = 0$. Unlike the block-diagonal preconditioner, $P_{\mathcal{H}}$ is a fairly good approximation to $A_{\mathcal{H}}$. We therefore propose to skip the mth GMRES iteration and approximate $\mathbf{b}$, $h_{m+1,m}$ with $[0, \cdots, 0, 1]^T$, 0, respectively. The number

of iterations, m, is small to begin with again because $P_{\mathcal{H}}$ approximates $A_{\mathcal{H}}$ well; thus, using one fewer iteration saves a large percentage of runtime.

We note that the closer $P_{\mathcal{H}}$ is to $A_{\mathcal{H}}$, the fewer outer iterations will be required, the better we can estimate the upper Hessenberg matrix, yet the more expensive the inner iteration (i.e., solving $P_{\mathcal{H}}\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$) will become. A good choice for $P_{\mathcal{H}}$ strikes a balance between the number of outer iterations and the cost of the inner iteration.

Both techniques described above are widely applicable. The coefficient matrix can be sparse or dense and does not need to be an FMM matrix. Krylov subspace recycling (section 3.2.1) can be applied whenever multiplying the inverse of the preconditioner and a vector also entails an iterative solve. The upper Hessenberg matrix estimation (section 3.2.2) can be performed whenever the preconditioner is a good approximation to the coefficient matrix.

3.3. The block-upper triangular preconditioner for augmented matrices

Free swimmers exert zero net force, zero net torque and undergo rigid translation and rotation. Assume that n free micro-swimmers are discretized by a total of N grid points located at $\{\mathbf{x}_j\}_{j=1}^{N} = \bigcup_{i=1}^{n}\{\mathbf{x}_{i_k}\}_{k=1}^{N_i}$, where $\{\mathbf{x}_{i_k}\}_{k=1}^{N_i}$ are on the ith swimmer and $\sum_{i=1}^{n} N_i = N$. Also assume that all the grid points on a swimmer are assigned consecutive subscripts. The forces $\{\mathbf{f}_j\}_{j=1}^{N}$ that they exert to the fluid must satisfy the following constraints:

$$\text{force-free:}\quad \sum_{k=1}^{N_i}\mathbf{f}_{i_k} = \mathbf{0}, \qquad \text{torque-free:}\quad \sum_{k=1}^{N_i}(\mathbf{x}_{i_k} - \mathbf{x}_c^i)\times\mathbf{f}_{i_k} = \mathbf{0},$$

where $\mathbf{x}_c^i$ is the center of mass of the ith swimmer.

Suppose that the emergent or total velocity $\mathbf{u}_{i_k}$ at a grid point $\mathbf{x}_{i_k}$ on the ith swimmer can be decomposed as

$$\mathbf{u}_{i_k} = \mathbf{v}_{i_k} + \boldsymbol{\Pi}_i + \boldsymbol{\Omega}_i\times(\mathbf{x}_{i_k} - \mathbf{x}_c^i),$$

where $\mathbf{v}_{i_k}$ is a prescribed velocity corresponding to the body deformations of the swimmer and $\boldsymbol{\Pi}_i$, $\boldsymbol{\Omega}_i$ are its translational and rotational velocities induced by $\{\mathbf{v}_j\}_{j=1}^{N}$. We can compute $\{\boldsymbol{\Pi}_i\}_{i=1}^{n}$, $\{\boldsymbol{\Omega}_i\}_{i=1}^{n}$ and $\{\mathbf{f}_j\}_{j=1}^{N}$ by solving a linear system $M\boldsymbol{\mathcal{G}}^{\star} = \boldsymbol{\mathcal{U}}^{\star}$, where

$$M = \begin{bmatrix} A & B^T \\ B & O_{6n\times 6n} \end{bmatrix} \in \mathbb{R}^{(3N+6n)\times(3N+6n)},\ \boldsymbol{\mathcal{G}}^\star = \begin{bmatrix} \boldsymbol{\mathcal{G}} \\ \boldsymbol{\Pi} \\ \boldsymbol{\Omega} \end{bmatrix} \in \mathbb{R}^{3N+6n},\ \boldsymbol{\mathcal{U}}^\star = \begin{bmatrix} \boldsymbol{\mathcal{U}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{3N+6n}. \tag{17}$$

In (17), the $(1,1)$-block $A \in \mathbb{R}^{3N\times 3N}$ in M is generated by the kernel function S_ε or W_ε depending on whether the fluid domain is bounded by a wall, $O_{m\times n}$ denotes the $m\times n$ zero matrix, $\boldsymbol{\mathcal{G}}$, $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{3N}$, $\boldsymbol{\Pi}$, $\boldsymbol{\Omega} \in \mathbb{R}^{3n}$ are formed by concatenating $\{\mathbf{f}_j\}_{j=1}^N$, $\{\mathbf{v}_j\}_{j=1}^N$, $\{\boldsymbol{\Pi}_i\}_{i=1}^n$ and $\{\boldsymbol{\Omega}_i\}_{i=1}^n$, respectively. In addition, the $(2,1)$-block $B \in \mathbb{R}^{6n\times 3N}$ in M is of the form

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_n \end{bmatrix} \tag{18}$$

where

$$B_i = \begin{bmatrix} I_3 & \cdots & I_3 & \cdots & I_3 \\ \left[\ \mathbf{x}_{i_1} - \mathbf{x}_c^i\ \right]_\times & \cdots & \left[\ \mathbf{x}_{i_k} - \mathbf{x}_c^i\ \right]_\times & \cdots & \left[\ \mathbf{x}_{i_{N_i}} - \mathbf{x}_c^i\ \right]_\times \end{bmatrix} \in \mathbb{R}^{6\times 3N_i}$$

corresponds to the ith swimmer and the 3×3 matrix $[\cdot]_\times$ has been defined in (4).

Assume that A is nonsingular. Then M has the following block LU decomposition:

$$M = \begin{bmatrix} A & B^T \\ B & O_{6n\times 6n} \end{bmatrix} = \begin{bmatrix} I_{3N} & O_{3N\times 6n} \\ BA^{-1} & I_{6n} \end{bmatrix} \begin{bmatrix} A & B^T \\ O_{6n\times 3N} & -BA^{-1}B^T \end{bmatrix} = LU,$$

which implies that all the eigenvalues of MU^{-1} equal 1. Consequently, the GMRES preconditioned by U would converge after only one iteration when applied to $M\boldsymbol{\mathcal{G}}^\star = \boldsymbol{\mathcal{U}}^\star$. However, using U as a preconditioner is impractical as it entails solving linear systems whose coefficient matrix is $BA^{-1}B^T$, which is too costly to form.

We propose the following block-upper triangular preconditioner for M:

$$P_M = \begin{bmatrix} P_A & B^T \\ O_{6n\times 3N} & P_S \end{bmatrix} \tag{19}$$

where P_A is a preconditioner for A and P_S is a preconditioner for $-BA^{-1}B^T$. Assume that both P_A and P_S are nonsingular. Then the inverse of P_M is

$$P_M^{-1} = \begin{bmatrix} P_A^{-1} & -P_A^{-1} B^T P_S^{-1} \\ O_{6n \times 3N} & P_S^{-1} \end{bmatrix}$$

and applying it to a vector entails solving two linear systems whose coefficient matrix is P_A and one linear system whose coefficient matrix is P_S.

We can choose P_A in (19) to be the block-diagonal preconditioner $P_{\mathcal{D}}$ (Section 3.1) or the data-sparser preconditioner $P_{\mathcal{H}}$ (Section 3.2). Again, $P_{\mathcal{D}}$ and $P_{\mathcal{H}}$ do not need to be formed.

Our construction of P_S in (19) is inspired by the so called "least-squares commutator" preconditioner [96, Chap. 9] developed for the sparse matrix arising from mixed finite element discretization of the incompressible Navier-Stokes equations. This matrix has a block structure very similar to that of M. The matrix $\widetilde{C}$ that minimizes $\|AB^T - B^T C\|$ over all $C \in \mathbb{R}^{6n \times 6n}$ is $\widetilde{C} = (BB^T)^{-1}(BAB^T)$. Therefore, loosely speaking, $AB^T \approx B^T \widetilde{C}$, which implies $A^{-1}B^T \approx B^T \widetilde{C}^{-1}$ and in turn leads to the following approximation:

$$BA^{-1}B^T \approx (BB^T)\widetilde{C}^{-1} = (BB^T)(BAB^T)^{-1}(BB^T). \tag{20}$$

Eq. (20) suggests $P_S = -(BB^T)(BAB^T)^{-1}(BB^T)$. Since $P_S^{-1} = -(BB^T)^{-1}(BAB^T)(BB^T)^{-1}$, the main cost of solving a linear system whose coefficient matrix is P_S is solving two subsidiary linear systems whose coefficient matrix is BB^T and multiplying A to a vector, the latter of which can again be accomplished by KIFMM. Constructing BB^T or applying its inverse is trivial: thanks to (18), $BB^T \in \mathbb{R}^{6n \times 6n}$ is block-diagonal with the 6×6 blocks $\{B_i B_i^T\}_{i=1}^n$ on its main diagonal.

4. Numerical experiments

We test the preconditioners described in Section 3 on the following three examples where the micro-structures are scattered particles, cilia, and free swimmers, respectively. The computing environment for all of the large-scale simulations is the same as specified in the Appendix.

4.1. A large group of uniformly random particles

We compare the performance of the following three versions of GMRES at solving $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$: GMRES with no preconditioner, GMRES preconditioned by the block-diagonal preconditioner $P_{\mathcal{D}}$ and GMRES preconditioned by the data-sparser preconditioner $P_{\mathcal{H}}$. We again consider three examples of A, namely, S240k, W240k and K480k, corresponding to the kernel functions S_ε, W_ε and K_ε and 80000 random particles uniformly distributed in $[0, 20] \times [0, 20] \times [0.01, 20.01]$ (see Appendix). The right-hand side $\boldsymbol{\mathcal{U}}$ is a random vector of unit norm, and the stopping criterion of GMRES is

$$\left\| A_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{U}} \right\| / \|\boldsymbol{\mathcal{U}}\| < 10^{-5}, \tag{21}$$

where $A_{\mathcal{H}}$ appears instead of A since the matrix-vector products involving A are approximated by KIFMM. In light of the triangle inequality

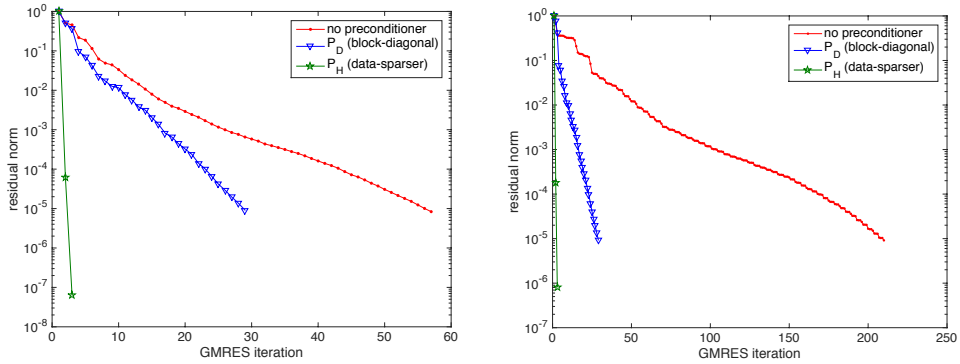
$$\left| \left\| A_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{U}} \right\| - \left\| A \widehat{\boldsymbol{\mathcal{G}}} - A_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}} \right\| \right| \leq \left\| A \widehat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{U}} \right\| \leq \left\| A_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{U}} \right\| + \left\| A \widehat{\boldsymbol{\mathcal{G}}} - A_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}} \right\|,$$

the accuracy of $\widehat{\boldsymbol{\mathcal{G}}}$ is limited by that of KIFMM; therefore, it is unnecessary to impose too stringent a stopping criterion on GMRES. In addition, when GMRES preconditioned by $P_{\mathcal{H}}$ is applied, the subsidiary linear systems in the form of $P_{\mathcal{H}} \boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$ are solved using GMRES preconditioned by $P_{\mathcal{D}}$ and its stopping criterion is

$$\left\| P_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{U}} \right\| / \|\boldsymbol{\mathcal{U}}\| < 10^{-5}.$$

We first compare the iteration counts of the three versions of GMRES. The two techniques described in Section 3.2 for enhancing the performance of $P_{\mathcal{H}}$ have not been implemented yet and will be considered later. Fig. 2 displays the decay of the residual norm $\left\| A_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{U}} \right\| / \|\boldsymbol{\mathcal{U}}\|$ as each version of GMRES progresses when A is S240k (left panel) or K480k (left panel). (The three curves obtained for W240k are highly similar to the ones corresponding to S240k.) For each of the three matrices, the numbers of iterations required by GMRES with no preconditioner and GMRES preconditioned by $P_{\mathcal{D}}$ as well as the numbers of quadrature points n_q, n_q' that $A_{\mathcal{H}}$, $P_{\mathcal{H}}$ correspond

to can be found in Table 1. We use fewer quadrature points for K480k since KIFMM produces much more accurate results for this matrix than for the other two when the same number of quadrature points are used (see Appendix). As illustrated in Fig. 2, the reduction in iteration count is dramatic when either preconditioner is used; in particular, GMRES preconditioned by $P_{\mathcal{H}}$ converges in only two iterations. If no preconditioner is used, GMRES converges in about 57 iterations when applied to S240k and W240k, yet it requires as many as 209 iterations to converge when applied to K480k. Based on the study of the condition numbers of the three types of MRS matrices, details in the Appendix, we do expect K480k to be much more challenging than the other two matrices for an iterative solver. Interestingly, for all three matrices, GMRES preconditioned by $P_{\mathcal{D}}$ converges in about 28 iterations. Further experiments that are not shown here also seem to support the same hypothesis: the number of iterations of GMRES preconditioned by $P_{\mathcal{D}}$ only depends on the quantity and distribution of the particles and does not depend on the kernel.



(a) S240k (b) K480k

Figure 2: Decay of the residual norm $\left\|A_{\mathcal{H}}\widehat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{U}}\right\| / \|\boldsymbol{\mathcal{U}}\|$ associated with the three versions of GMRES (no preconditioning, preconditioned by $P_{\mathcal{D}}$, and preconditioned by $P_{\mathcal{H}}$). S240k in (a) corresponds to the free-space, force-only case and K480k in (b) to the free-space case where torque is also accounted for (see Appendix).

Next, we re-run GMRES preconditioned by $P_{\mathcal{H}}$ with both Krylov subspace recycling and upper Hessenberg matrix estimation incorporated, and we compare the runtimes of the three versions of GMRES. Details of the two acceleration techniques are as follows. At the end of each subsidiary solve $P_{\mathcal{H}}\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$, we retain the harmonic Ritz vectors associated with the largest $\ell/2$ or $(\ell+1)/2$ (whichever one is an integer) harmonic Ritz values, where ℓ is the dimension of an existing Krylov subspace. Recall that GMRES preconditioned by $P_{\mathcal{H}}$ takes two iterations to converge. We now run

only one iteration and approximate, using the estimated 2×2 upper Hessenberg matrix, the solution that we would obtain if the second iteration were executed. This approach reduces the number of subsidiary solves from three to two; we also verify that it indeed produces a solution of essentially the same accuracy as before. As can be seen from Table 1, the runtime of GMRES preconditioned by $P_{\mathcal{H}}$ with acceleration ranges from 36% to 59% of that required by GMRES preconditioned by $P_{\mathcal{D}}$; we note that the efficiency of $P_{\mathcal{H}}$ is ultimately achieved by avoiding the matrix-vector multiplication involving $A_{\mathcal{H}}$, which is much more costly than that involving $P_{\mathcal{H}}$ or $P_{\mathcal{D}}^{-1}$.

Table 1: Comparison of the three versions of GMRES applied to S240k, W240k and K480k (W240k corresponds to the half-bounded, force-only case). GMRES preconditioned by $P_{\mathcal{H}}$ is accelerated by Krylov subspace recycling and upper Hessenberg matrix estimation. Number of processors: 16. "MVPs" = matrix-vector products, "iters." = iterations, and "sec." = second. n_q, n_q': numbers of quadrature points associated with $A_{\mathcal{H}}$, $P_{\mathcal{H}}$.

		no preconditioner			$P_{\mathcal{D}}$			$P_{\mathcal{H}}$			
matrix	n_q	# of iters.	# of MVPs ($A_{\mathcal{H}}$)	run-time (sec.)	# of iters.	# of MVPs ($A_{\mathcal{H}}/P_{\mathcal{D}}^{-1}$)	run-time (sec.)	# of iters.	# of MVPs ($A_{\mathcal{H}}/P_{\mathcal{D}}^{-1}/P_{\mathcal{H}}$)	run-time (sec.)	n_q'
S240k	208	56	56	2332	28	28/29	1168	1	1/50/48	483	64
W240k	208	58	58	5893	29	29/30	2923	1	1/54/52	1038	64
K480k	124	209	209	11209	28	28/29	1589	1	1/50/48	943	26

4.2. A dense mat of cilia

The respiratory tracts are lined with a dense mat of cilia which beat back and forth rhythmically to propel mucus out of the lung; this muco-ciliary clearance serves as a defense system against airborne infection. Cilia are also responsible for ovum transport in the oviduct of the female as well as sperm transport in the efferent ducts of the male. These motile cilia are elastic, flexible structures that are actively generating forces and will interact with each other due to their close proximity. When modeling cilia, viscoelastic effects might be important due to the relaxation time of the elastic flagellum as well as the surrounding fluid.

To illustrate the utility of our methods for a dense mat of cilia, we are ignoring viscoelastic effects and we do not account for feedback from the surrounding fluid on the beat cycle of the cilia. That is, we will only explore the resulting flow fields due to cilia with a prescribed, time-dependent beat

cycle. To achieve this, we utilize an accurate representation of the lung cilia beat cycle developed in [97]. In this model, a cilium is characterized by a 2D parametric curve: at time t, the location of any point on the cilium is given by the truncated Fourier series

$$\mathbf{x}(s,t) = \frac{1}{2}\mathbf{a}_0(s) + \sum_{n=1}^{N_0} \mathbf{a}_n(s)\cos(n\sigma t) + \mathbf{b}_n(s)\sin(n\sigma t) \tag{22}$$

in $\mathbb{R}^2$, where s is the arclength of the point measured from the base of the cilium and σ is the angular beat frequency. The constant vectors $\{\mathbf{a}_n\}_{n=0}^{N_0}, \{\mathbf{b}_n\}_{n=1}^{N_0} \subset \mathbb{R}^2$ are computed by applying Fourier analysis and the least-squares method to a series of cilia profiles traced using high speed cine-photography.

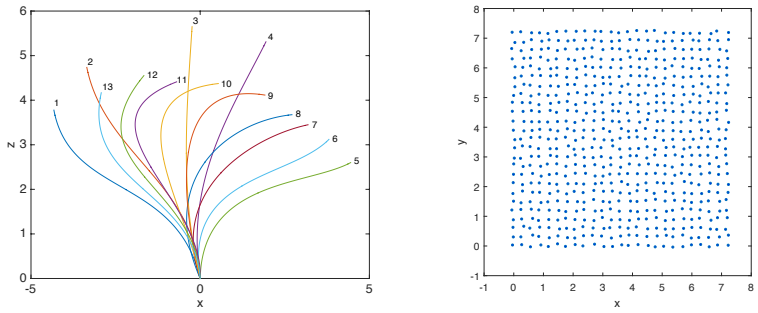
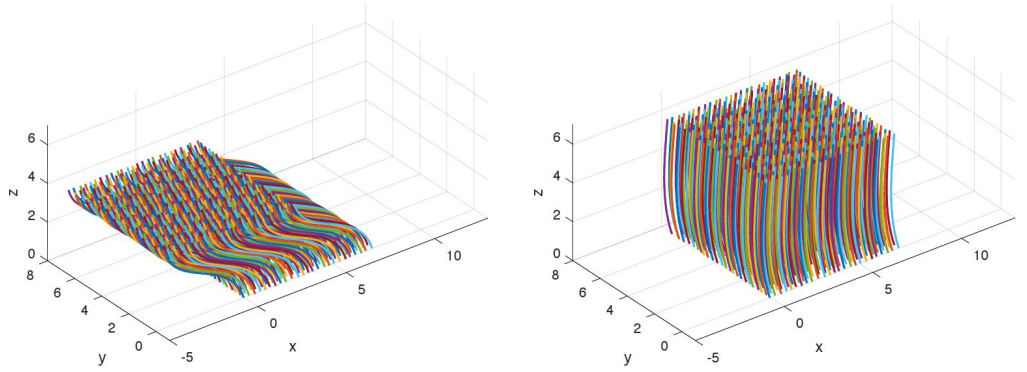


Figure 3: *Left:* 13 snapshots of a simulated cilium in one beat cycle. *Right:* The bases of a perturbed grid of cilia (cilia spacing is $h_c = 0.3$ and the level of perturbation is 20% at most in either the x or y direction).

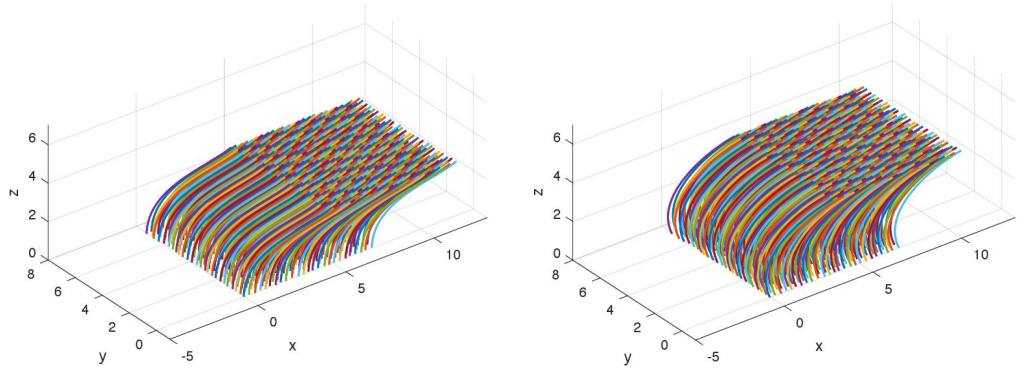
Using the Fourier coefficients computed in [98] based on the experimental data of [99], we simulate the motion of a planar cilium in one beat cycle and display 13 snapshots of it in the left panel of Fig. 3. (The larger its numerical label is, the later a profile appears in the cycle.) The elapsed time between two consecutive snapshots is the same. Snapshots 1 to 4 depict the power (effective) stroke whereas snapshots 5 to 13 capture the recovery stroke. In [88], the flow field induced by a single cilium attached to a stationary, planar wall has been studied using MRS: first, the velocities at the grid points on the cilium are found by differentiating Eq. (22) with respect to time; then the forces that induce the velocities of the grid points at each of the thirteen instants are recovered by solving a linear system. With the forces at our disposal, the velocity at *any* location in the fluid and at any one of the thirteen instants can be calculated by (3). The approach of [88] has been applied to compute the flow field induced by an array of cilia in [100–103]. When the number of cilia is large, however, it gives rise to a sequence of linear systems $\{A^{(n)}\boldsymbol{\mathcal{G}}^{(n)} = \boldsymbol{\mathcal{U}}^{(n)}\}_{n=1}^{13}$

whose coefficient matrices are large, dense and varying.



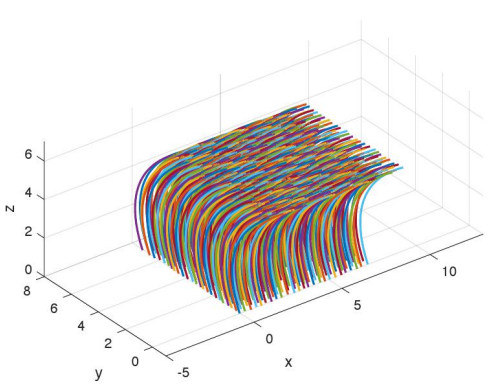
(a) Snapshot 1

(b) Snapshot 3

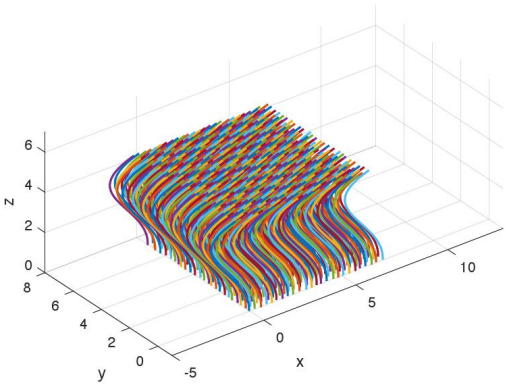


(c) Snapshot 5

(d) Snapshot 7



(e) Snapshot 9



(f) Snapshot 11

Figure 4: Snapshots 1, 3, 5, 7, 9 and 11 of the cilia carpet ($h_c = 0.3$).

We consider a dense carpet of cilia attached to a stationary, planar wall at $z = 0$. The bases of the cilia are the perturbed grid points of a 25×25 uniform grid on the wall (see the right panel of Fig. 3). To be more precise, if h_c denotes the grid spacing of the uniform grid, the base point of each cilium is located at $(x_0 + \xi h_c, y_0 + \zeta h_c, 0)$ where $(x_0, y_0, 0)$ is a grid point on the uniform grid and ξ, ζ are uniformly random between -20% and 20%. Thus, h_c is roughly the average grid spacing in the perturbed grid. Two values of h_c, 0.3 and 0.6, are considered in the numerical experiments. Although the cilia carpet is three-dimensional, every cilium is restricted to be two-dimensional: it is initially placed and will remain in the plane $y = y_b$, where y_b is the y coordinate of its base point. The x, z coordinates of the points on the cilium are determined by (22) as well as the x, z coordinates of the base point[4] and their y coordinates all equal y_b. Each cilium is of length 6 and uniformly discretized by 101 grid points. Note that (22) does not preserve the length of the cilium; for it to have constant length, we rescale the x, z coordinates of the grid points accordingly. Since the base points of the cilia are "tethered" to the wall, there are a total of $100 \times 25^2 = 62500$ particles in our calculation, resulting in coefficient matrices of dimension 187500×187500. In addition, we specify that the carpet of cilia beat in synchrony in these simulations, that is, their profiles are identical at each time point. In Fig. 4, six snapshots of the cilia carpet in one beat cycle are displayed. Unlike in Section 4.1 where the particles are uniformly random, the distribution of the grid points on the cilia can be far from uniform, as can be seen from Fig. 4. The beat frequency f of the cilia is 15Hz (or 15 beat cycles per second) and their angular beat frequency σ in (22) is $2\pi f = 30\pi$ accordingly. The regularization parameter ε in MRS is chosen to be 0.1 to match the radius of the cilia. The length and radius of the cilia as well as their beat frequency used in our study are the same as those in [98].

The values of various dimensional parameters for the cilia carpet, are summarized in Table 2. We first compare the performance of the block-diagonal preconditioner $P_{\mathcal{D}}$ and the data-sparser preconditioner $P_{\mathcal{H}}$. The cilia grid spacing h_c is set to 0.3, also matching with the value used in [98]. The x, z components of the velocity at each grid point are again found by differentiating (22) with respect to time, whereas the y component is set to 0 thus forcing the cilia to stay in their

[4]To be more precise, at time t, the x coordinate of the point with arclength s is given by the first component of $\mathbf{x}$ in (22) plus the x coordinate of the base point, and the z coordinate of the same point equals the second component of $\mathbf{x}$ plus the z coordinate of the base point, which is 0 in our simulations.

Table 2: Summary of parameter values used for the cilia carpet simulations.

Parameter	Value
Fluid viscosity	1 g μm s^{-1}
Number of cilia	$25 \times 25 = 625$
Number of grid points on each cilium	101
Number of grid points in the fluid (N)	$625 \times 100 = 62500$
Length of cilium	6 μm
Radius of cilium and ε	0.1 μm
Average cilia spacing (h_c)	0.3 or 0.6 μm
Beat frequency (f)	15 Hz
Angular beat frequency (σ)	30π
N_0, $\{\mathbf{a}_n\}_{n=0}^{N_0}$ and $\{\mathbf{b}_n\}_{n=1}^{N_0}$ in (22)	see Table 2(B), Eq. (2) in [97]

initial specified planes. Due to the higher particle density and larger ε, the linear systems arising from the cilia carpet are significantly more challenging to solve than the ones in Section 4.1 (see the Appendix for a summary of how the condition number varies with particle density and ε); therefore, we use the following milder stopping criteria for the preconditioned GMRES instead:

$$\left\| A_{\mathcal{H}}^{(n)} \widehat{\boldsymbol{\mathcal{G}}}^{(n)} - \boldsymbol{\mathcal{U}}^{(n)} \right\| / \left\| \boldsymbol{\mathcal{U}}^{(n)} \right\| < 10^{-3}, \quad \left\| P_{\mathcal{H}}^{(n)} \widehat{\boldsymbol{\mathcal{G}}}_k^{(n)} - \boldsymbol{\mathcal{U}}_k^{(n)} \right\| / \left\| \boldsymbol{\mathcal{U}}_k^{(n)} \right\| < 10^{-3}, \ n = 1, 2, \ldots, 13.$$

As in Section 4.1, we apply one iteration of the GMRES preconditioned by $P_{\mathcal{H}}$ that is accelerated by Krylov subspace recycling and upper Hessenberg matrix estimation. This approach outperforms GMRES preconditioned $P_{\mathcal{D}}$ in every case and requires 30% less runtime to solve all thirteen linear systems. The computational results for six linear systems are reported in Table 3. We observe that the advantage of $P_{\mathcal{H}}$ is the most pronounced when applied to the third linear system, which corresponds to the most uniformly distributed grid points (see Fig. 4). Although the less stringent stopping criteria (4.2) are used and the matrices are smaller than W240k considered in Section 4.1, the iteration counts and runtimes shown in Table 3 are still considerably higher than those reported for W240k in Table 1, indicating the difficulty of these problems.

As a carpet of cilia is simply a collection of points to KIFMM, whether they are synchronous or asynchronous, planar or three-dimensional does not affect how we solve the linear systems.

Table 3: Comparison of $P_{\mathcal{D}}$ and $P_{\mathcal{H}}$ in the case of a cilia carpet ($h_c = 0.3$) beating with a prescribed velocity. The snapshot numbers correspond to the cilia locations given in Fig. 4. Krylov subspace recycling and upper Hessenberg matrix estimation are applied in conjunction with $P_{\mathcal{H}}$. Number of quadrature points: $n_q = 208$ for $A_{\mathcal{H}}$ and $n_q' = 64$ for $P_{\mathcal{H}}$. Number of processors: 16.

	$P_{\mathcal{D}}$			$P_{\mathcal{H}}$		
snapshot	# of iters.	# of MVPs ($A_{\mathcal{H}}/P_{\mathcal{D}}^{-1}$)	runtime (sec.)	# of iters.	# of MVPs ($A_{\mathcal{H}}/P_{\mathcal{D}}^{-1}/P_{\mathcal{H}}$)	runtime (sec.)
1	74	74/75	5714	1	1/132/130	3199
3	52	52/53	5679	1	1/94/92	1642
5	74	74/75	6814	1	1/148/146	3964
7	69	69/70	4028	1	1/136/134	3449
9	93	93/94	5633	1	1/95/93	4862
11	70	70/71	3962	1	1/127/125	2803

Next, we study the effect of the cilia spacing h_c on fluid mixing. We track the locations of 3375 particles immersed in the fluid for three beat cycles. They are initially placed on the grid points of a $15 \times 15 \times 15$ uniform grid over the region $[0, 7.2] \times [0, 7.2] \times [0.1, 6.1]$ (when $h_c = 0.3$) or $[0, 14.4] \times [0, 14.4] \times [0.1, 6.1]$ (when $h_c = 0.6$) directly above the bases of the cilia; and they move with the flow passively without exerting any force on the fluid. The initial positions of the particles when $h_c = 0.3$ are plotted in the left panel of Fig. 5. Furthermore, each beat cycle is divided into thirteen equal time intervals such that the beginning of the ith interval coincides with the time at which the ith snapshot in the left panel of Fig. 3 is taken. The length of the time intervals is thus $\Delta t = \frac{1}{13f}$ where f is the cilia beat frequency. We find the forces exerted by the cilia at the beginning of the ith time interval by solving the ith linear system, compute the velocities of the passive particles at the same instant by (3) and calculate their locations at the end of the ith interval using forward Euler's method, assuming that the particles satisfy a no-slip condition and move with the local fluid flow. (Since we choose $\mu = 1$ when solving for the forces earlier, we use the same μ when computing the velocities of the particles.) The 3375 particles after three beat cycles are plotted in the right panel of Fig. 5 again for the $h_c = 0.3$ case.

As a means of visualizing the mixing, in Fig. 6 to Fig. 8, we illustrate the projections of the passive particles onto either the xz-plane or xy-plane at four instants. In each figure, the first row

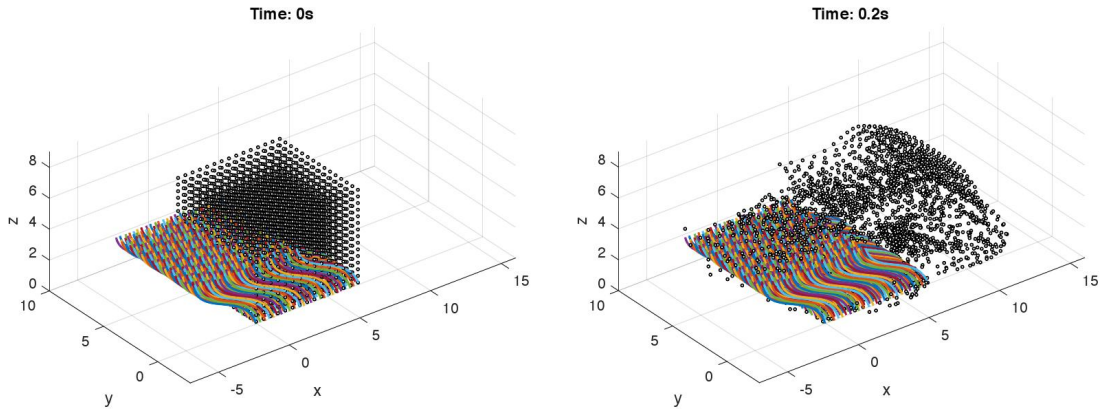


Figure 5: The cilia carpet and particles at the beginnings of the first (left) and fourth (right) beat cycles ($h_c = 0.3$).

of subplots correspond to $h_c = 0.3$ and the second row to $h_c = 0.6$, whereas each column of subplots correspond to a different instant. In the first column of each figure, the projections of the particles at the initial time are shown; they are evenly divided into three horizontal or vertical "stripes" and represented by three different markers accordingly. How the stripe patterns evolve during the simulation provides insights into the fluid mixing along each axis. From Fig. 6 and Fig. 7, both of which display the projections of the particles onto the xz-plane, we observe that the original stripe patterns get distorted and can even become unrecognizable as time proceeds. This indicates strong mixing in both the x and z directions. On the other hand, in Fig. 8 where the projections of the particles onto the xy-plane are shown instead, the original stripe patterns are clearly identifiable throughout the simulation, suggesting that the mixing in the y direction is much weaker. This makes sense since the cilia are specified to move in their initial planes only, which are parallel to the xz plane. Moreover, by comparing the two rows of subplots in each figure, we can see that mixing is stronger along all three axes when the cilia spacing is smaller.

The numerical results also suggest that a smaller cilia spacing promotes the effectiveness of the power stroke: when $h_c = 0.3$, the average signed distance traveled by the 3375 particles along the x-axis is 1.78 and 73% of them move in the positive x direction; when $h_c = 0.6$, these two numbers decrease to 0.82 and 63%, respectively.

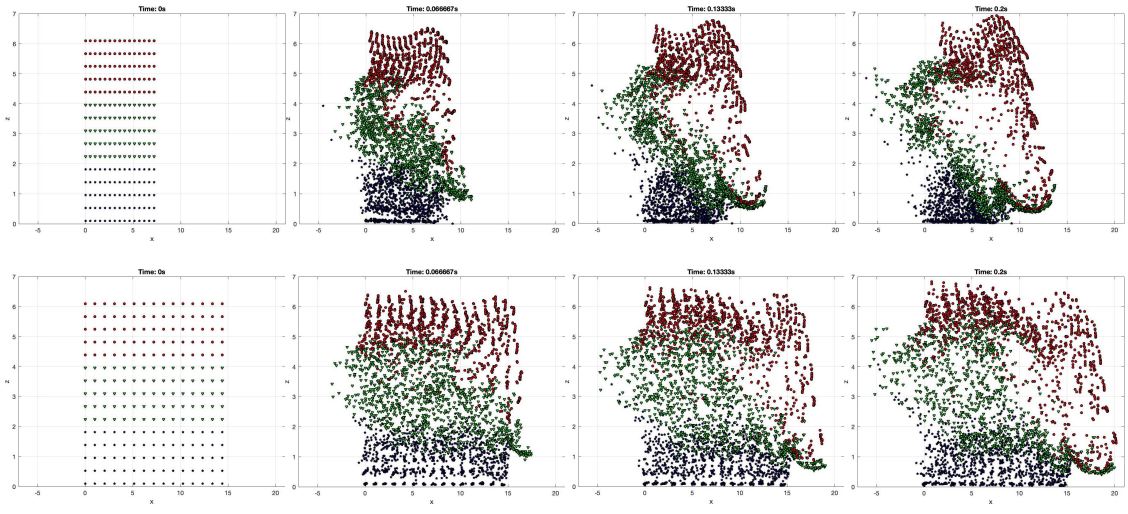


Figure 6: The projections of the particles onto the xz-plane. (They are divided into horizontal stripes initially.) Top row: $h_c = 0.3$. Bottom row: $h_c = 0.6$. First to fourth columns: the beginnings of the first to fourth cycles.

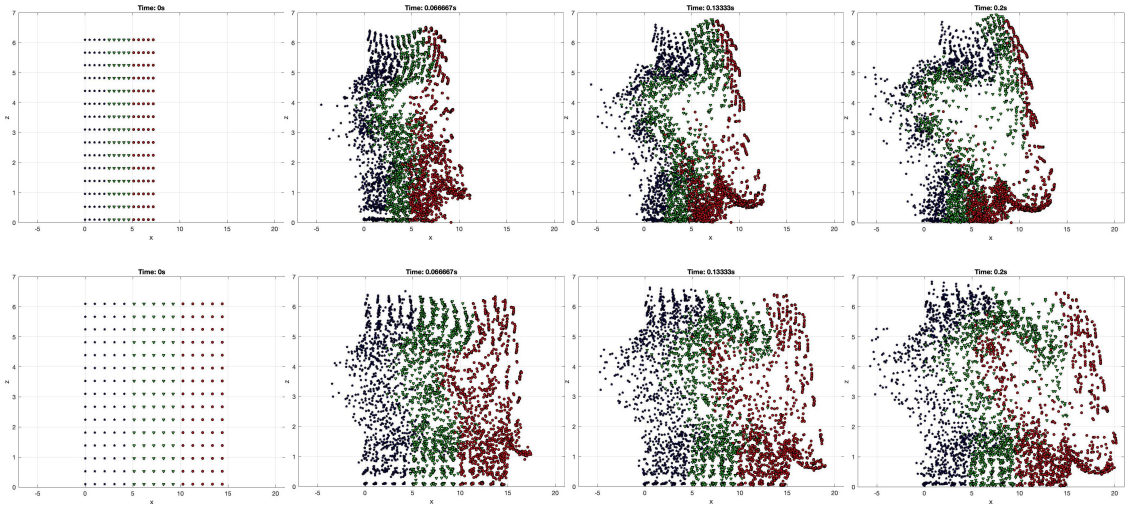


Figure 7: The projections of the particles onto the xz-plane. (They are divided into vertical stripes initially.) Top row: $h_c = 0.3$. Bottom row: $h_c = 0.6$. First to fourth columns: the beginnings of the first to fourth cycles.

4.3. A large group of free swimmers above a planar wall

The setting here is quite similar to that of Section 4.1: there are 80000 random particles uniformly distributed in $[0, 20] \times [0, 20] \times [0.01, 20.01]$ and a stationary, planar wall located at $z = 0$.

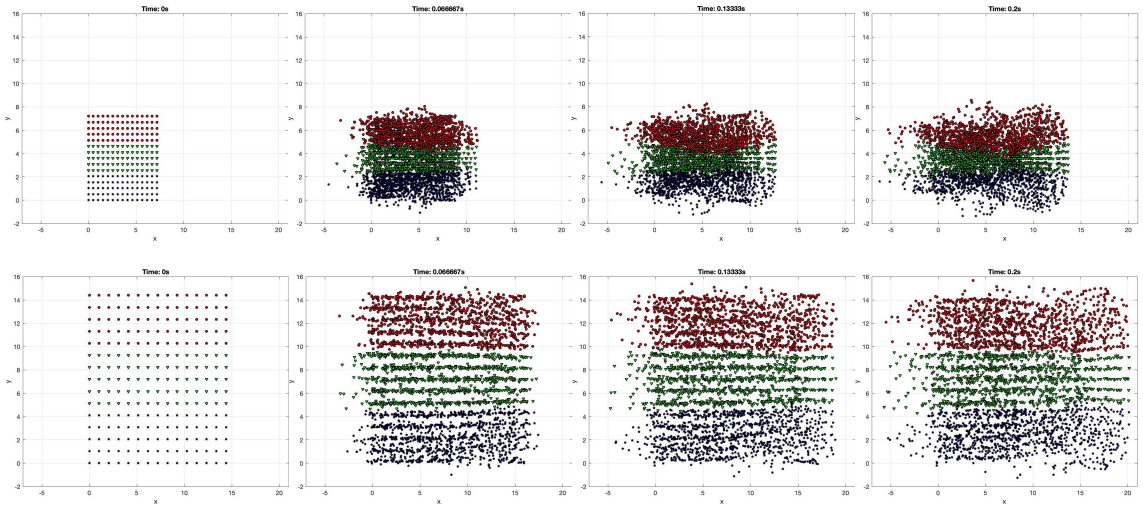
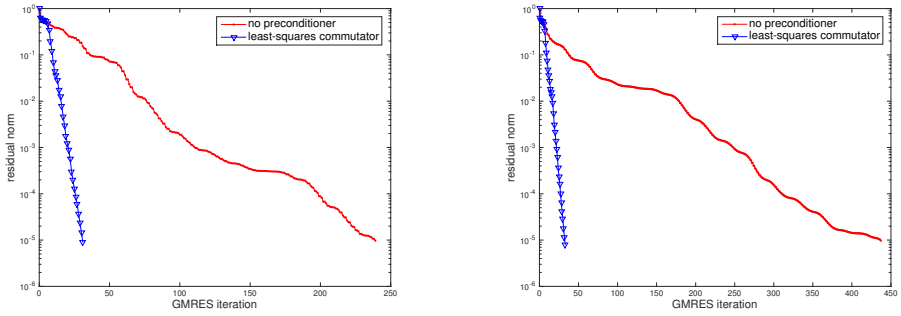


Figure 8: The projections of the particles onto the xy-plane. Top row: $h_c = 0.3$. Bottom row: $h_c = 0.6$. First to fourth columns: the beginnings of the first to fourth cycles.

The difference is that instead of being independent, unconstrained swimmers, the particles are now grouped into n multi-particle, force-free and torque-free swimmers. To be more precise, for $1 \leq i \leq n$, we let the $[(i-1)p+1]$st to the (ip)th particles represent the ith swimmer, where p is the number of particles per swimmer and thus $np = 80000$. The augmented coefficient matrix M is of size $(240000 + 6n) \times (240000 + 6n)$ and its $(2,1)$-block, B, is of size $6n \times 240000$; regardless of what n, p are, the $(1,1)$-block of M, A, is of size 240000×240000 and identical to the matrix W240k considered in Section 4.1. Multiplying M and a vector entails one matrix-vector product involving A, which is again approximated using KIFMM; the actual coefficient matrix in our computations is thus $M_{\mathcal{H}}$ whose $(1,1)$-block is $A_{\mathcal{H}}$ instead of A. In the block-upper triangular preconditioner P_M in (19), we choose $P_A = P_{\mathcal{D}}$ and $P_S = -(BB^T)(BP_{\mathcal{H}}B^T)^{-1}(BB^T)$, where $P_{\mathcal{H}}$ is again the $\mathcal{H}^2$-matrix associated with a version of KIFMM of lower accuracy. Note that $P_{\mathcal{H}}$ is used instead of $A_{\mathcal{H}}$ to further reduce the computational cost of applying P_S^{-1}. The matrix BB^T is $6n \times 6n$, block-diagonal and we pre-compute it before running GMRES; it is cheap to compute and store BB^T as well as to apply its inverse to vectors. We refer to P_M as the least-squares commutator preconditioner. To sum up, each iteration of GMRES preconditioned by P_M mainly requires solving two linear systems whose coefficient matrix is $P_{\mathcal{D}}$, two linear systems whose coefficient matrix is BB^T and computing one matrix-vector product involving $A_{\mathcal{H}}$, one matrix-vector product involving $P_{\mathcal{H}}$. As both $P_{\mathcal{D}}$ and

BB^T are block-diagonal, the four linear solves all boil down to solving a number of much smaller linear systems; and the two matrix-vector products are computed using KIFMM. The $\boldsymbol{\mathcal{U}}$ component in $\boldsymbol{\mathcal{U}}^\star$ (see (17)) is a random vector of unit norm.



(a) 800 free swimmers

(b) 8000 free swimmers

Figure 9: Decay of the residual norms $\left\| M_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}^\star} - \boldsymbol{\mathcal{U}}^\star \right\| / \left\| \boldsymbol{\mathcal{U}}^\star \right\|$ associated with two versions of GMRES (no preconditioner and least-squares commutator preconditioner P_M).

Table 4: Comparison of no preconditioner and the least-squares commutator preconditioner in the free swimmer case. Number of quadrature points: $n_q = 208$ for $A_{\mathcal{H}}$ and $n'_q = 64$ for $P_{\mathcal{H}}$. Number of processors: 16.

	no preconditioner			least-squares commutator		
# of swimmers	# of iter.	# of MVPs ($\boldsymbol{A}_{\mathcal{H}}$)	runtime (sec.)	# of iter.	# of MVPs ($\boldsymbol{A}_{\mathcal{H}}/\boldsymbol{P}_{\mathcal{D}}^{-1}/\boldsymbol{P}_{\mathcal{H}}$)	runtime (sec.)
800	226	226	21944	31	31/64/32	3560
8000	429	429	42773	33	33/68/34	3740

We apply GMRES with no preconditioner and GMRES preconditioned by P_M to two augmented linear systems: one corresponding to 800 swimmers with 100 particles per swimmer and the other to 8000 swimmers with 10 particles per swimmer. The center of each swimmer is simply taken to be the mean of the coordinate vectors of its member particles. We emphasize that both problems correspond to the same 80000 particles; the only difference is how these particles are "lumped" to form the multi-particle swimmers. Such swimmers are fictitious as they are each represented by a collection of random particles uniformly distributed in the cube $[0, 20] \times [0, 20] \times [0.01, 20.01]$ which can be far away from one another. Nonetheless, they serve as "model swimmers" that allow

us to investigate how the number of appended rows and columns and the number of swimmers affect the performance of the solvers while the quantity and distribution of the particles remain *fixed*. We use the stopping criterion $\left\| M_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}^\star} - \boldsymbol{\mathcal{U}}^\star \right\| / \|\boldsymbol{\mathcal{U}}^\star\| < 10^{-5}$ and summarize in Table 4 the computational results of the two versions of GMRES applied to the two augmented systems. In Fig. 9, the history of $\left\| M_{\mathcal{H}} \widehat{\boldsymbol{\mathcal{G}}^\star} - \boldsymbol{\mathcal{U}}^\star \right\| / \|\boldsymbol{\mathcal{U}}^\star\|$ associated with each application of GMRES is also displayed. First, we compare the performance of GMRES at solving $M\boldsymbol{\mathcal{G}}^\star = \boldsymbol{\mathcal{U}}^\star$ and $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$. Since the $(1,1)$-block in M is the same as W240k and is preconditioned by $P_{\mathcal{D}}$ in the least-squares commutator preconditioner P_M, we compare the results of GMRES preconditioned by P_M in Table 4 with those corresponding to W240k and $P_{\mathcal{D}}$ in Table 1. When no preconditioner is applied, the augmented linear system $M\boldsymbol{\mathcal{G}}^\star = \boldsymbol{\mathcal{U}}^\star$ is significantly more challenging: it requires about three ($n = 800$) or six ($n = 8000$) times more GMRES iterations than $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$. The preconditioned GMRES, on the other hand, incurs more or less the same number of iterations when applied to either $M\boldsymbol{\mathcal{G}}^\star = \boldsymbol{\mathcal{U}}^\star$ or $A\boldsymbol{\mathcal{G}} = \boldsymbol{\mathcal{U}}$. Next, we examine how the number of swimmers influence the performance of both versions of GMRES at solving $M\boldsymbol{\mathcal{G}}^\star = \boldsymbol{\mathcal{U}}^\star$. Without a preconditioner, the case with 8000 swimmers entails almost twice as many GMRES iterations as those required by the case with 800 swimmers; in contrast, the two cases require almost the same number of iterations when GMRES preconditioned by P_M is applied. In other words, the least-squares commutator preconditioner is insensitive to the number of extra rows and columns appended to A in M.

5. Conclusion

Large, dense matrices arise from modeling the hydrodynamic interactions among a large group of micro-structures at zero Reynolds number. Matrix-vector multiplications and linear solves involving these matrices are highly challenging, especially in a time-dependent simulation. Under the framework of the Method of Regularized Stokeslets, we extend the Kernel-Independent Fast Multipole Method to calculate the matrix-vector product that gives the flow induced by a large number of point forces above a stationary, planar wall satisfying a no-slip condition. Additionally, we also explore the use of different memory efficient preconditioners to solve the linear system for the unknown forces that the structures are exerting on the fluid when their movement is known. These preconditioners are data-sparse, retaining and representing either a portion or all of the interactions, and they do not need to be constructed explicitly. We demonstrate their effectiveness through a variety of cases including point forces in free space, point forces and torques in free space, and point forces above a wall. The preconditioners, as well as the techniques that we propose to enhance their performance, are applicable to an entire family of matrices, with those obtained from the Rotne-Prager-Yamakawa tensor included. Extending our approach to the case of free swimmers turns out to be rather tricky due to the extra equations for imposing the torque-free, force-free condition. We develop a preconditioner for the augmented linear system by minimizing the entries of a commutator-like matrix in the least squares sense.

We will continue to optimize our KIFMM code. For example, the dual tree traversal [104–106] considered in [77] seems very promising and has not been implemented in the current code. We emphasize that the kernel W_ε (see (7)) is not translation invariant and thus, some optimization techniques are not applicable. The numerical experiments in this work demonstrate the effectiveness of the proposed preconditioning techniques in the biological applications of interest. A careful comparison of our approach with existing methods such as the inverse FMM [68, 69] still needs to be performed in a modern high performance computing environment on problems of larger scales.

There has been a large amount of work in related systems, which could potentially be used to further enhance computational efficiency and/or model additional examples with the regularized Stokeslets framework. For example, we have focused on free-space solutions, but some applications may necessitate a spatially-periodic domain or it may be easier to model periodic copies of cilia or swimmers when dealing with a large number. In the case of regularized forces, regularized greens functions for doubly and triply periodic stokes flow have been developed and fast summations have

been achieved via Ewald methods [83, 107, 108]. We believe that these methods for the regularized KIFMM for periodic flows should be feasible using a near and far field splitting scheme similar to [46], where the case of singular point forces were studied. One future direction will be to extend the KIFMM methodology developed in this work for the regularized image system for periodic Stokes flow [109–111]. Recently, a force-neutral image system was developed to capture a no-slip wall in non-periodic and periodic geometries [47], which is conceivable to also be utilized in the case of regularized point forces.

In this manuscript, we have focused on modeling biological structures that are represented as regularized point forces. The regularization is necessary to ensure finite velocity fields at the locations coinciding with the structures so that a no-slip condition can be utilized to move the structure in time. Previous studies have shown that a smeared or spread force is a good representation for slender or filamentous structures, where a space curve in 2D or 3D is spread mainly to a region governed by the regularization parameter [15, 80]. We note that other applications that are not biological structures could potentially utilize this framework where forces are regularized to ensure integrable kernels. However, one must always ensure that the physics of the system are being maintained, and careful choice of blob functions and desired error accuracy must be considered [112]. We have used a regularization function with infinite support where the majority of the force is spread to a region corresponding to the regularization parameter. In the future, we plan to consider a compactly supported regularization function similar to the one used in [83] and compare the performance of KIFMM when applied to the two families of kernels corresponding to the two types of regularization functions. In addition, we will investigate how the error of KIFMM and the condition number of the KIFMM matrix depend on the regularization parameter ε. We also note that structures represented by a collection of points on a larger temporal and spatial scale are of interest in the current study. In the case that one is modeling point particles and if Brownian dynamics are important, Stokesian dynamics or the RPY tensor is a better choice than MRS to study these micro-structures [12, 113, 114].

Due to the implications that cilia dysfunction has in terms of diseases, there has been great interest in studying the collective dynamics of cilia with regards to fluid mixing. In the section on numerical experiments, we highlight how our method is able to handle a large yet finite number of cilia that are tethered to a planar wall and have a prescribed beat form. For the purposes of this study, we have focused on the case where all of the cilia in the group are beating in phase and there

are no out of plane motions. However, this method can easily extend to scenarios that include cilia with phase differences and nonplanar components of the beat form, and can be used to understand regimes of cilia beating that result in pumping and mixing of the fluid as in [115]. Depending on the questions of interest related to the ciliary beat and fluid flow, previous studies have looked at a single cilium or a small group of cilia, whereas others have studied large patches of cilia or even an infinite array of cilia [100, 116–119]. Our method is a nice alternative in the more realistic case where the cilia carpet is not of infinite extent. We also note that in our numerical experiments, a very basic uniform discretization of the cilia and a relatively large yet realistic regularization parameter are used, which have led to matrices that are difficult to work with. In further applications, other techniques involving optimal discretization of the cilia and methods to decouple the regularization parameter from the physical radius of the cilia could be utilized [119].

Appendix

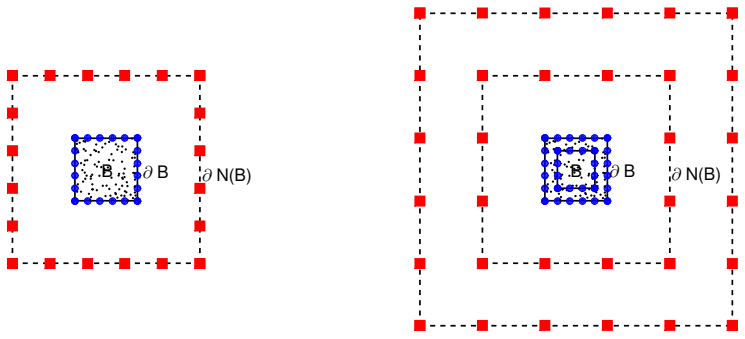
In this section, the fluid viscosity $\mu = 1$ and whenever the kernel function W_ε is considered, it is implied that there is a stationary, planar wall located at $z = 0$.

All of the numerical experiments are performed in MATLAB version R2017a. The large-scale simulations are parallelized using The Parallel Computing Toolbox of MATLAB and run on 16 processors (Intel Xeon CPU E5-2699 v3 @ 2.3 GHz with 32 GB of RAM) on a virtual machine.

The performance of KIFMM applied to the MRS matrices

We present the performance of two versions of KIFMM, namely, the surface version [28] and the corona version [29], at approximating $A\boldsymbol{\mathcal{G}}$, where A represents the pairwise interactions among 80000 random particles uniformly distributed in $[0, 20] \times [0, 20] \times [0.01, 20.01]$ and is generated by the kernel function S_ε, W_ε or K_ε with $\varepsilon = 0.02$.[5] The three matrices are of size 240000×240000 (for S_ε and W_ε) or 480000×480000 (for K_ε) and are denoted by S240k, W240k and K480k, respectively. In both versions of KIFMM, the above cube is uniformly partitioned three times, which creates 512 leaf boxes with about 160 particles in each on average. The difference between the two versions is illustrated in Fig. 10 for a 2D box $\mathfrak{B}$, where the disks represent the quadrature points $\{\mathbf{q}_j\}_{j=1}^{n_q}$ in (10) and the squares represent the quadrature points $\{\widetilde{\mathbf{q}}\}_{j=1}^{n_q}$ in (11). To obtain the fictitious forces $\{\mathbf{g}_j\}_{j=1}^{n_q}$ in (10) (or $\{\widetilde{\mathbf{g}}_j\}_{j=1}^{n_q}$ in (11)) for $\mathfrak{B}$, we need to solve an integral equation in which the domain of integration is the upward (or downward) equivalent surface/corona of $\mathfrak{B}$. We first replace the integral with the trapezoidal rule over n_q quadrature points on the equivalent surface/corona; the integral equation becomes a possibly ill-conditioned linear system, which we then solve using Tikhonov regularization with suitably chosen parameters. The runtimes of direct calculation are 122, 249 and 299 seconds for S240k, W240k and K480k, respectively. For all three problems and both versions of KIFMM, we summarize the computational results corresponding to a few values of n_q in Tables 5 and 6. In both tables, "speedup" refers to the runtime of direct calculation divided by the runtime of KIFMM; therefore, the larger this number is, the more efficient KIFMM is compared to direct calculation. The relative error refers to $\|A_{\mathcal{H}}\boldsymbol{\mathcal{G}} - A\boldsymbol{\mathcal{G}}\|/\|A\boldsymbol{\mathcal{G}}\|$ where $A_{\mathcal{H}}\boldsymbol{\mathcal{G}}$ is the estimate to $A\boldsymbol{\mathcal{G}}$ computed by KIFMM.

[5] The z-coordinates of the particles are bounded away from 0 so that the matrix W240k is nonsingular.



(a) Equivalent surfaces. (b) Equivalent coronas.

Figure 10: The equivalent surfaces and coronas of a box in 2D and the quadrature points on them. Left plot: the upward equivalent surface (—●—) and the downward equivalent surface (– – ■ – –). Right plot: the upward equivalent corona (the region between the two —●— boundaries) and the downward equivalent corona (the region between the two – – ■ – – boundaries). A 6×6 uniform Cartesian grid is used to discretize each equivalent surface or corona. The quadrature points on the equivalent surfaces or coronas are marked by ● or ■, respectively. In the figures, B=the box of interest, ∂B=boundary of B, and ∂N(B)=boundary of neighborhood of B.

Table 5: Performance of the surface version [28] of KIFMM applied to S240k, W240k and K480k. Number of processors: 16.

		S240k		W240k		K480k	
grid	n_q	speed-up	relative error	speed-up	relative error	speed-up	relative error
$3 \times 3 \times 3$	26	24.40	$1.93\ 10^{-3}$	24.80	$5.76\ 10^{-3}$	21.36	$6.26\ 10^{-6}$
$4 \times 4 \times 4$	56	17.43	$9.18\ 10^{-5}$	16.53	$1.91\ 10^{-4}$	14.95	$2.09\ 10^{-7}$
$5 \times 5 \times 5$	98	9.38	$5.47\ 10^{-5}$	8.86	$6.19\ 10^{-5}$	8.31	$1.04\ 10^{-8}$
$6 \times 6 \times 6$	152	4.88	$1.35\ 10^{-6}$	4.28	$5.15\ 10^{-6}$	4.27	$7.11\ 10^{-10}$

Table 6: Performance of the corona version [29] of KIFMM applied to S240k, W240k and K480k. Number of processors: 16.

		S240k		W240k		K480k	
grid	n_q	speed-up	relative error	speed-up	relative error	speed-up	relative error
$4 \times 4 \times 4$	64	15.25	$3.25\ 10^{-6}$	14.59	$8.77\ 10^{-5}$	14.24	$2.18\ 10^{-8}$
$5 \times 5 \times 5$	124	6.78	$1.19\ 10^{-7}$	6.05	$2.03\ 10^{-5}$	5.98	$1.19\ 10^{-9}$
$6 \times 6 \times 6$	208	2.77	$1.65\ 10^{-8}$	2.36	$3.10\ 10^{-7}$	2.43	$6.18\ 10^{-11}$

We expect both the speedup and the error of either version of KIFMM to decrease as the number of quadrature points increases; this is indeed confirmed by the results shown in Tables 5 and 6 for all three matrices. KIFMM is substantially more efficient than direct calculation: even when 208 quadrature points are used, which is the largest number considered, the runtime required by KIFMM is still less than half of that required by direct calculation in all three cases. When the same number of quadrature points is used, KIFMM produces the most accurate estimate for K480k and the least accurate estimate for W240k; in fact, for most values of n_q, the relative error associated with the latter is more than 1000 times greater than that associated with the former. Furthermore, although W240k and S240k are of the same size, whether KIFMM or direct calculation is applied, the runtime required by W240k is always more than twice as long as that required by S240k. The two observations above indicate that W_ε is a rather challenging kernel.

The condition numbers of MRS matrices

We investigate how the condition numbers of MRS matrices change with various factors. For each combination of domain, regularization parameter ε and kernel function, we repeat the following experiment 10000 times: generate 100 random particles uniformly distributed in the domain, construct the MRS matrix representing their pairwise interactions, and compute the condition number of this matrix. The size of each sample matrix is 300×300 (for S_ε and W_ε) or 600×600 (for K_ε). The average condition number of each group of 10000 sample matrices is reported in Table 7.

As expected, when the domain is $[0,1] \times [0,1] \times [0,1]$, since the particles can get arbitrarily close to the wall at $z = 0$, the matrices generated by W_ε are ill-conditioned with an average condition number of 5.62×10^{12}. After we bound the particles away from the wall by changing the domain to

Table 7: Average condition number of 10000 sample MRS matrices generated for each combination of domain, regularization parameter and kernel function

Domain	ε	kernel	avg. cond. #
$[0,1]\times[0,1]\times[0.01,1.01]$	0.02	W_ε	1.13×10^{1}
$[0,1]\times[0,1]\times[0,1]$			5.62×10^{12}
		K_ε	1.22×10^{4}
		S_ε	1.06×10^{1}
	0.04		4.85×10^{1}
	0.08		3.28×10^{2}
$[0,2]\times[0,2]\times[0,2]$	0.02		3.78×10^{0}
$[0,4]\times[0,4]\times[0,4]$			2.07×10^{0}

$[0,1]\times[0,1]\times[0.01,1.01]$, the average condition numbers of the matrices generated by W_ε and those generated by S_ε become almost the same (1.13×10^{1} and 1.06×10^{1}), and they are both considerably smaller than the average condition number of the matrices generated by K_ε (1.22×10^{4}). Moreover, the average condition number grows as the density of the particles or the regularization parameter ε increases and is especially sensitive to the latter.

References

[1] L. Cisneros, J. Kessler, S. Ganguly, R. Goldstein, Dynamics of swimming bacteria: transition to directional order at high concentration, Phys Rev E 83 (2011) 061907.

[2] N. Mendelson, A. Bourque, K. Wilkening, K. Anderson, J. Watkins, Organized cell swimming motions in bacillus subtilis colonies: patterns of short-lived whirls and jets, J Bacteriol 181 (1999) 600–609.

[3] R. Thar, M. Kuhl, Conspicuous veils formed by vibrioid bacteria on sulfidic marine sediment, Appl Environ Microbiol 68 (2002) 6310–6320.

[4] H. Wioland, F. Woodhouse, J. Dunkel, J. Kessler, R. Goldstein, Confinement stabilizes a bacterial suspension into a spiral vortex, Phys Rev Lett 110 (2013) 268102.

[5] H. Moore, K. Dvorakova, N. Jenkins, W. Breed, Exceptional sperm cooperation in the wood mouse, Nature 418 (2002) 174–177.

[6] I. Riedel, K. Kruse, J. Howard, A self-organized vortex array of hydrodynamically entrained sperm cells, Science 309 (2005) 300–303.

[7] T. Sanchez, D. Chen, S. DeCamp, M. Heymann, Z. Dogic, Spontaneous motion in hierarchically assembled active matter, Nature 491 (2012) 431–435.

[8] T. Surrey, F. Nedelec, S. Leibler, E. Karsenti, Physical properties determining self-organization of motors and microtubules, Science 292 (2001) 1167–1171.

[9] C. Peskin, The immersed boundary method, Acta Numerica 11 (2002) 459–517.

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

[11] J. Brady, Stokesian dynamics, Ann Rev Fluid Mech 20 (1988) 111–156.

[12] A. Banchio, J. Brady, Accelerated Stokesian dynamics: Brownian motion, J Chem Phys 118 (2003) 10323–10332.

[13] A. Sierou, J. Brady, Accelerated Stokesian dynamics simulations, J Fluid Mech 448 (2001) 115–146.

[14] R. Cortez, The method of regularized Stokeslets, SIAM J Sci Comput 23 (2001) 1204–1225.

[15] R. Cortez, L. Fauci, A. Medovikov, The method of regularized Stokeslets in three dimensions: Analysis, validation, and application to helical swimming, Phys Fluids 17 (2005) 031504.

[16] J. Rotne, S. Prager, Variational treatment of hydrodynamic interaction in polymers, J Chem Phys 50 (1969) 4831–4837.

[17] H. Yamakawa, Transport properties of polymer chains in dilute solution: hydrodynamic interaction, J Chem Phys 53 (1970) 436–443.

[18] H. Cheng, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm in three-dimensions, J Comput Phys 155 (1999) 468–498.

[19] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J Comput Phys 73 (1987) 325–348.

[20] L. Greengard, V. Rokhlin, A new version of the fast multipole method for the Laplace equation in three dimensions, Acta Numer 6 (1997) 838–853.

[21] V. Rokhlin, Rapid solution of integral equations of classical potential theory, J Comput Phys 60 (1985) 187–207.

[22] A. Tornberg, L. Greengard, A fast multipole method for the three-dimensional stokes equations, J Comput Phys 227 (2008) 1613–1619.

[23] J. Barnes, P. Hut, A hierarchical O(N log N) force-calculation algorithm, Nature 324 (1986) 446–449.

[24] J. Barnes, A modified tree code: Don't laugh; it runs, J Comp Phys 87 (1990) 161–170.

[25] S. Pfalzner, P. Gibbon, Many Body Tree Methods in Physics, Cambridge University Press, Cambridge, UK, 1996.

[26] L. Wang, S. Tlupova, R. Krasny, A treecode algorithm for 3D Stokeslets and Stresslets, 2018. In Review, arXiv:1811.12498.

[27] L. Wang, R. Krasny, S. Tlupova, A kernel-independent treecode based on barycentric Lagrange interpolation, 2019. In Review, arXiv:1902.02250.

[28] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J Comput Phys 196 (2004) 591–626.

[29] L. Ying, A kernel independent fast multipole algorithm for radial basis functions, J Comput Phys 213 (2006) 451–457.

[30] M. Rostami, S. Olson, Kernel-independent fast multipole method within the framework of regularized Stokeslets, J Fluid Struct 67 (2016) 60–84.

[31] E. Lauga, T. Powers, The hydrodynamics of swimming microorganisms, Rep Prog Phys 72 (2009) 096601.

[32] L. Fauci, A. McDonald, Sperm motility in the presene of boundaries, Bull Math Biol 57 (1995) 679–699.

[33] J. Huang, L. Carichino, S. Olson, Hydrodynamic interactions of actuated elastic filaments near a planar wall with applications to sperm motility, J Coupled Syst Multiscale Dyn 6 (2018) 163–175.

[34] D. Woolley, Motility of spermatozoa at surfaces, Reproduction 126 (2003) 259–270.

[35] D. Smith, J. Blake, Surface accumulation of spermatozoa: A fluid dynamic phenomenon, Math Scientist 34 (2009) 74–87.

[36] L. Rothschild, Non-random distribution of bull spermatozoa in a drop of sperm suspension, Nature 198 (1963) 1221–1222.

[37] C. Tung, F. Ardon, A. Fiore, S. Suarez, M. Wu, Cooperative roles of biological flow and surface topography in guiding sperm migration revealed by a microfluidic model, Lab Chip 14 (2014) 1348–1356.

[38] S. Suarez, Mammalian sperm interactions with the female reproductive tract, Cell Tissue Res 363 (2016) 185–194.

[39] J. Hernandez-Ortiz, C. Stoltz, M. Graham, Transport and collective dynamics in suspensions of confined self-propelled particles, Phys Rev Lett 95 (2005) 204501.

[40] A. Costanzo, R. Di Leonardo, G. Ruocco, L. Angelani, Transport of self-propelling bacteria in micro-channel flow, J Phys Condens Matter 24 (2012) 065101.

[41] E. Lushi, H. Willard, R. Goldstein, Fluid flows created by swimming bacteria drive self-organization in confined suspensions, Proc Natl Acad Sci USA 111 (2014) 9733–9738.

[42] A. Tsang, E. Kanso, Flagella-induced trasitions in the collective behavior of confined microswimmers, Phys Rev E 90 (2014) 021001(R).

[43] A. Tsang, E. Kanso, Circularly confined microswimmers exhibit multiple global patterns, Phys Rev E 91 (2015) 043008.

[44] A. Tsang, E. Kanso, Density shock waves in confined microswimmers, Phys Rev Lett 116 (2016) 048101.

[45] J. Blake, A note on the image system for a stokeslet in a no-slip boundary, Mathematical Proceedings of the Cambridge Philosophical Society 70 (1971) 303–310.

[46] W. Yan, M. Shelley, Flexibly imposing periodicity in kernel independent FMM: A multipole-to-local operator approach, J Comp Phys 355 (2018) 214–232.

[47] W. Yan, M. Shelley, Universal image systems for non-periodic and periodic stokes flows above a no-slip wall, J Comp Phys 375 (2018) 263–270.

[48] Z. Gimbutas, L. Greengard, S. Veerapaneni, Simple and efficient representations for the fundamental solutions of stokes flow in a half-space, J Fluid Mech 776 (2015) R1.

[49] M. Bebendorf, Hierarchical matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems, Springer-Verlag, Berlin, Germany, 2008.

[50] W. Hackbusch, Hierarchical matrices: Algorithms and Analysis, Springer-Verlag, Berlin, Germany, 2015.

[51] S. Börm, $\mathcal{H}^2$-matrix arithmetics in linear complexity, Computing 77 (2006) 1–28.

[52] S. Börm, Data-sparse approximation of non-local operators by $\mathcal{H}^2$-matrices, Linear Algebra Appl 422 (2007) 380–403.

[53] S. Börm, K. Reimer, Efficient arithmetic operations for rank-structured matrices based on hierarchical low-rank updates, Comput Vis Sci 16 (2013) 247–258.

[54] W. Hackbusch, S. Börm, Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices, Computing 69 (2002) 1–35.

[55] W. Hackbusch, S. Börm, $\mathcal{H}^2$-matrix approximation of integral operators by interpolation, Appl Numer Math 43 (2002) 129–143.

[56] W. Hackbusch, B. Khoromskij, S. Sauter, On $\mathcal{H}^2$-matrices, in: H. Bungartz, R. Hoppe, Z. Ch (Eds.), Lectures on Applied Mathematics, Springer-Verlag, Berlin, Germany, 2000, pp. 9–29.

[57] S. Börm, L. Grasedyck, W. Hackbusch, Introduction to hierarchical matrices with applications, Eng Anal Bound Elem 27 (2003) 405–422.

[58] W. Hackbusch, A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices, Computing 62 (1999) 89–108.

[59] W. Hackbusch, B. Khoromskij, A sparse $\mathcal{H}$-matrix arithmetic: General complexity estimates, J Comput Appl Math 125 (2000) 479–501.

[60] A. Aminfar, S. Ambikasaran, E. Darve, A fast block low-rank dense solver with applications to finite-element matrices, J Comput Phys 304 (2016) 170–188.

[61] W. Kong, J. Bremer, V. Rokhlin, An adaptive fast direct solver for boundary integral equations in two dimensions, Appl Comput Harmon Anal 31 (2011) 346–369.

[62] M. Chandrasekaran, S. Gu, T. Pals, A fast ULV decomposition solver for hierarchically semiseparable representations, SIAM J Matrix Anal Appl 28 (2006) 603–622.

[63] E. Corona, P. Martinsson, D. Zorin, An $O(N)$ direct solver for integral equations on the plane, Appl Comput Harmon Anal 38 (2015) 284–317.

[64] A. Gillman, P. Martinsson, An $O(N)$ algorithm for constructing the solution operator to 2D elliptic boundary value problems in the absence of body loads, Adv Compt Math 40 (2014) 773–796.

[65] P. Martinsson, V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J Comput Phys 205 (2005) 1–23.

[66] J. Xia, M. Chandrasekaran, M. Gu, X. S. Li, Fast algorithms for hierarchically semiseparable matrices, Numer Linear Algebra Appl 17 (2010) 953–976.

[67] J. Xia, M. Chandrasekaran, M. Gu, X. S. Li, Superfast multifrontal method for large structured linear systems of equations, SIAM J Matrix Anal Appl 31 (2010) 1382–1411.

[68] S. Ambikasaran, E. Darve, The inverse fast multipole method, arXiv:1407.1572v1 [math.NA], 2014.

[69] P. Coulier, H. Pouransari, E. Darve, The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems, SIAM J Sci Comput 39 (2017) A761–A796.

[70] G. Alléon, M. Benzi, L. Giraud, Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics, Numer Algorithms 16 (1997) 1–15.

[71] B. Carpentieri, Algebraic preconditioners for the Fast Multipole Method in electromagnetic scattering analysis from large structures: Trends and problems, Electronic Journal of Boundary Elements 7 (2009) 13–49.

[72] B. Carpentieri, I. Duff, L. Giraud, Sparse pattern selection strategies for robust frobenius-norm minimization preconditioners in electromagnetism, Numer Linear Algebra Appl 7 (2000) 667–685.

[73] B. Carpentieri, I. Duff, L. Giraud, G. Sylvand, Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations, SIAM J Sci Comput 27 (2005) 774–792.

[74] K. Chen, On a class of preconditioning methods for dense linear systems from boundary elements, SIAM J Sci Comput 20 (1998) 684–698.

[75] J. Lee, J. Zhang, C.-C. Lu, Sparse inverse preconditioning of multilevel fast multipole algorithm for hybrid integral equations in electromagnetics, IEEE Trans Antennas Propag 52 (2004) 2277–2287.

[76] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, Philadelphia, USA, 2003.

[77] H. Ibeid, R. Yokota, J. Pestana, D. Keyes, Fast multipole preconditioners for sparse matrices arising from elliptic equations, Comput Vis Sci 18 (2018) 213–229.

[78] R. Yokota, H. Ibeid, D. Keyes, Fast multipole method as a matrix-free hierarchical low-rank approximation, in: Eigenvalue Problems: Algorithms, Software and Applications in Petascale Computing, Springer International, Switzerland, 2017, pp. 267–286.

[79] S. Jung, K. Mareck, L. Fauci, M. Shelley, Rotational dynamics of a superhelix towed in a stokes fluid, Phys Fluids 19 (2007) 103105–1–6.

[80] B. Rodenborn, C. Chen, H. Swinney, B. Liu, H. Zhang, Propulsion of microorganisms by a helical flagellum, Proc Natl Acad Sci USA 110 (2013) 338–347.

[81] E. L. Bouzarth, M. L. Minion, Modeling slender bodies with the method of regularized Stokeslets, Journal of Computational Physics 230 (2011) 3929–3947.

[82] H. Nguyen, R. Ortiz, R. Cortez, L. Fauci, The action of waving cylindrical rings in a viscous fluid, J Fluid Mech 671 (2011) 574–586.

[83] K. Leiderman, E. Bouzarth, R. Cortez, A. Layton, A regularization method for the numerical solution of periodic stokes flow, J Comp Phys 236 (2013) 187–202.

[84] H. Flores, E. Lobaton, S. Mendez-Diez, S. Tlupova, R. Cortez, A study of bacterial flagellar bundling, Bull Math Bio 65 (2005) 137–168.

[85] S. Lim, Dynamics of an open elastic rod with intrinsic curvature and twist in a viscous fluid, Phys Fluids 22 (2010) 024104.

[86] S. Lim, A. Ferent, X. Wang, C. Peskin, Dynamics of a closed rod with twist and bend in fluid, SIAM J Sci Comput 31 (2008) 273–302.

[87] S. Olson, S. Lim, R. Cortez, Modeling the dynamics of an elastic rod with intrinsic curvature and twist using a regularized stokes formulation, J Comput Phys 283 (2013) 169–187.

[88] J. Ainley, S. Durkin, R. Embid, P. Boindala, R. Cortez, The method of images for regularized Stokeslets, J Comput Phys 227 (2008) 4600–4616.

[89] R. Cortez, D. Varela, A general system of images for regularized stokeslets and other elements near a plane wall, J Comput Phys 285 (2015) 41 – 54.

[90] R. Adrian, J. Westerweel, Particle image velocimetry, Cambridge University Press, Cambridge, UK, 2011.

[91] A. Schroeder, C. Willert, Particle image velocimetry : new developments and recent applications, Springer, Berlin, Germany, 2008.

[92] Y. Saad, M. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J Sci Stat Comput 7 (1986) 856–869.

[93] F. Balboa Usabiaga, B. Kallemov, B. Delmotte, A. Bhalla, B. Griffith, A. Donev, Hydrodynamics of suspensions of passive and active rigid particles: a rigid multiblob approach, Comm App Math Comp Sci 11 (2016) 217–296.

[94] E. Nazockdast, A. Rahimian, D. Zorin, M. Shelley, A fast platform for simulating semi-flexible fiber suspensions applied to cell mechanics, J Comp Phys 329 (2017) 173–209.

[95] M. Parks, E. De Sturler, G. Mackey, D. Johnson, S. Maiti, Recycling Krylov subspaces for sequences of linear systems, SIAM J Sci Comput 28 (2006) 1651–1674.

[96] H. Elman, D. Silvester, A. Wathen, Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics, Oxford University Press, Oxford, 2014.

[97] J. Blake, A model for the micro-structure in ciliated organisms, J Fluid Mech 55 (1972) 1–23.

[98] G. Fulford, J. Blake, Muco-ciliary transport in the lung, J Theor Biol 121 (1986) 381–402.

[99] M. Sanderson, M. Sleigh, Ciliary activity of cultured rabbit tracheal epithelium, J Cell Sci 47 (1981) 331–347.

[100] Y. Ding, J. Nawroth, M. McFall-Ngai, E. Kanso, Mixing and transport by ciliary carpets: a numerical study, J Fluid Mech 743 (2014) 124–140.

[101] H. Guo, J. Nawroth, Y. Ding, E. Kanso, Cilia beating patterns are not hydrodynamically optimal, Phys Fluids 26 (2014) 091901.

[102] Y. Ding, E. Kanso, Selective particle capture by asynchronously beating cilia, Phys Fluids 27 (2015) 121902.

[103] J. Nawroth, H. Guo, E. Koch, E. Heath-Heckman, J. Hermanson, E. Ruby, J. Dabiri, E. Kanso, M. McFall-Ngai, Motile cilia create fluid-mechanical microhabitats for the active recruitment of the host microbiome, PNAS 114 (2017) 9510–9516.

[104] J. Carrier, L. Greengard, V. Rokhlin, A fast adaptive multipole algorithm for particle simulations, SIAM J Sci Stat Comput 9 (1988) 669–686.

[105] M. Warren, J. Salmon, A portable parallel particle program, Comput Phys Commum 87 (1995) 266–290.

[106] W. Dehnen, A hierarchical O(N) force calculation algorithm, J Comput Phys 179 (2002) 27–42.

[107] R. Cortez, F. Hoffmann, A fast numerical method for computing doubly-periodic regularized stokes flow in 3d, J Comp Phys 258 (2014) 1–14.

[108] K. Leiderman, E. Bouzarth, H.-N. Nguyen, A regularization method for the numerical solution of doubly-periodic stokes flow, AMS proceeding (2014).

[109] F. Hoffmann, R. Cortez, Numerical computation of doubly-periodic Stokes flow bounded by a plane with applications to nodal cilia, Comm Comp Phys 22 (2017) 620–642.

[110] H. Nguyen, Leiderman, Computation of the singular and regularized image systems for doubly-periodic stokes flow in the presence of a wall, J Comput Phys 297 (2015) 442–461.

[111] F. Mannan, R. Cortez, An explicit formula for two-dimensional singly-periodic regularized stokeslets flow bounded by a plane wall, Comm Comput Phys 23 (2018) 142–167.

[112] P. Martin, On the use of approximate fundamental solutions: connections with the method of fundamental solutions and the method of regularized stokeslets, Eng Anal Bound Elem 99 (2019) 23–28.

[113] Z. Liang, Z. Gimbutas, L. Greengard, J. Huang, S. Jiang, A fast multipole method for the Rotne-Prager-Yamakawa tensor and its applications, J Comput Phys 234 (2013) 133–139.

[114] J. Swan, J. Brady, Simulation of hydrodynamically interacting particles near a no-slip boundary, Phys Fluids 19 (2007) 113306.

[115] A. Shields, B. Fiser, B. Evans, M. Falvo, S. Washburn, R. Superfine, Biomimetic cilia arrays generate simultaneous pumping and mixing regimes, PNAS 107 (2010) 15670 – 15675.

[116] R. Dillon, L. Fauci, An integrative model of internal axoneme mechanics and external fluid dynamics in ciliary beating, J Theor Biol 207 (2000) 415–430.

[117] H. Guo, J. Nawroth, Y. Ding, E. Kanso, Cilia beating patterns are not hydrodynamically optimal, Phys Fluids 26 (2014) 091901.

[118] S. Mitran, Metachronal wave formation in a model of pulmonary cilia, Comput Struct 85 (2007) 763–774.

[119] D. Smith, A boundary element regularized stokeslet method applied to cilia- and flagella-driven flow, Proc R Soc A 465 (2009) 3605–3626.