Kernel-Independent Fast Multipole Method within the framework of Regularized Stokeslets

Minghao W. Rostami*,a, Sarah D. Olsona

aDepartment of Mathematical Sciences, Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609, U.S.A.

Abstract

The method of regularized Stokeslets (MRS) uses a radially symmetric blob function of infinite support to smooth point forces and allow for evaluation of the resulting flow field. This is a common method to study swimmers at zero Reynolds number where the Stokeslet is the fundamental solution corresponding to the kernel of the single layer potential. Simulating the collective motion of \( N \) micro swimmers using the MRS results in at least \( N^2 \) pair-wise interactions. Efficient simulation of a large number of swimmers in free space is observed with the implementation of the kernel-independent fast multiple method (FMM) for radial basis functions. We illustrate the complexity of the algorithm on a simple test case where we study regularized point forces, showing that the method is \( O(N) \). Additionally, we explore accuracy in time for the MRS where the swimmers are modeled as Kirchhoff rods and the kernel-independent FMM is compared to the direct calculation using the standard MRS. Optimal hydrodynamic efficiency is also explored for different configurations of swimmers.

Key words: kernel-independent, fast multipole method, regularized Stokeslets, fluid-structure interaction, Kirchhoff rod

*Corresponding author

Email addresses: mwu@wpi.edu (Minghao W. Rostami), sdolson@wpi.edu (Sarah D. Olson)

Preprint submitted to Journal of Fluids and Structures

December 21, 2015
1. Introduction

The collective motion of microorganisms will lead to self organization into larger scale groups. At high enough density, experiments have observed the formation of vortices of both bacteria and sperm when confined by a surface [1, 2]. Sperm at high density have also been observed to line up and aggregate to form ‘sperm trains’ [3] that are ten times the length of a single sperm and several sperm wide. Other experiments of bacteria have revealed self organization into veils, vortices, and jets [4–6]. More generally, we can view active matter to include the collective motion of microorganisms, molecular motors, and active colloids [7, 8]. The collective motion of microtubules and kinesin also form asters and vortices [9, 10]. It is of great interest to understand biological implications for collective dynamics as well as to understand the role of hydrodynamic interactions. Comprehensive computational models and analysis has been completed to understand the role of hydrodynamic, steric, and chemical interactions on collective motion of a large number of structures [11–19].

These examples of active matter are at the scale of zero Reynolds number, where viscous effects dominate. The incompressible Stokes equations for a Newtonian fluid are given as

\[-\mu \Delta u + \nabla p = F,\]  
\[\nabla \cdot u = 0,\]

where $\mu$ is the viscosity, $p$ is the pressure at position $x$, $u$ is the velocity at $x$, and $F$ is the force exerted on the fluid at $x$. When there is only one point force $f_o$ which is located at position $y_o$, $F$ can be represented as $f_o \delta(x - y_o)$ where $\delta(\cdot)$ is the Dirac delta distribution. The Stokeslet (Green’s function) is the fundamental solution to Eq. (1a)-(1b) for a point force and is singular at the location of the point force. Note
that for active matter located at $N$ distinct spatial locations in the infinite fluid domain, we can write

$$
\mathbf{F} = \sum_{j=1}^{N} f_j \delta(\mathbf{x} - \mathbf{y}_j)
$$

for $N$ point forces and solve for the fluid flow as a superposition of fundamental solutions.

Several fluid-structure interaction methods have been developed to handle point forces or slender elastic structures in a fluid. The immersed boundary method requires both a Lagrangian and Cartesian grid, which can be computationally expensive [20]. With the boundary integral equation method, one can recast the governing equations for smooth objects immersed in a Stokesian fluid as integral equations of the boundaries of the objects [21]. This reduces the complexity, solving 2D surface integrals instead of computing 3D flows. When close to a boundary, this method will involve discretization and evaluation of integrals with singular kernels. The approach that we use in this study is the method of regularized Stokeslets (MRS) [22, 23], a Lagrangian numerical algorithm that has been developed to regularize point forces and remove the singularity in the resulting fluid flow.

As will be shown in Sections 4 and 5, under the framework of the MRS, computing the velocities at $N$ locations in the fluid caused by $N$ point forces boils down to an $N$-body problem, the computational complexity of which is $O(N^2)$ if done exactly. In the current paper, we focus on the efficient numerical solution of this problem for a large $N$. Fast algorithms for the summation of a large number of pairwise interactions have been the subject of research activity over the last 40 years; among them are the tree codes [24, 25], the Fast Multiple Method (FMM) [26–32] and the panel clustering method [33, 34], whose computational complexities range from $O(N)$ to $O(N \log^{d+2} N)$ with $d$ being the dimension of the computational do-
main. Techniques based on Ewald summation [35-39] have also been developed for a different problem where the interactions between $N$ particles and all their periodic images need to be computed. The FMM, first proposed by Greengard and Rokhlin [28], achieves optimal complexity $O(N)$ and is considered to be one of the top algorithms of the 20th century [40]. The classic FMM requires an analytic factorization of the kernel and has been successfully applied to the Stokes equations (1a)-(1b) where the Stokeslet and Stresslet kernels have been determined based on expansions of the Green’s function for the biharmonic function [30]. However, for many kernels including the ones arising from the MRS, such a factorization can be difficult to find. The kernel-independent FMM [31] was developed with the aim of handling a broader range of problems and only requires a numerical factorization of the kernel. It is applicable to the kernel of any second-order elliptic PDE and has also been extended in [32] to handle kernels that are radial basis functions.

Our main contribution is extending the kernel-independent FMM to solve the resulting fluid flow due to the collective motion of a large number of microorganisms modeled by the MRS. In Section 2, we review the MRS and show how the regularized kernels using the MRS result in radial basis functions. Next, we summarize the numerical algorithm for the kernel-independent FMM in Section 3. To test this algorithm, in Section 4 we report the computational time of the algorithm as well as explore how its error varies with different parameters. As an application to collective motion, we explore the swimming efficiency of different configurations of swimmers using a Kirchhoff rod model formulation in Section 5. We show that the speed, efficiency and tilt of a swimmer in a large group of swimmers depends highly on their initial placement in the group.
2. Method of Regularized Stokeslets

The Method of Regularized Stokeslets (MRS) has been used to successfully model swimmers or structures that can be represented as curves or scattered points in space [22, 23]. The structures are assumed to be neutrally buoyant and immersed in a viscous, incompressible 3D unbounded fluid at zero Reynolds number. The point force \( \mathbf{f} \) (and possibly point torque \( \mathbf{n} \)) that a structure exerts on the fluid will be regularized or smoothed by a blob function \( \varphi(r) \) that is a radially symmetric approximation to a 3D delta distribution satisfying \( 4\pi \int_0^\infty r^2 \varphi(r) dr = 1 \). Here, \( r = ||\mathbf{x} - \mathbf{y}_o||_2 \) is the distance between the location of the point force \( \mathbf{y}_o \) and any point \( \mathbf{x} \) in the fluid. Two example blob functions are [41, 42]:

\[
\psi(r) = \frac{15}{8\pi(1 + r^2)^{7/2}}, \quad \phi(r) = \frac{(5 - 2r^2)}{2\pi^{3/2}} e^{-r^2}.
\]  

It is easy to see that the following two families of blob functions are parametrised by \( \varepsilon > 0 \) and can be obtained by scaling \( \psi(r) \) or \( \phi(r) \) given in Eq. (2) by \( 1/\varepsilon^3 \) and evaluating them at \( r/\varepsilon \):

\[
\psi_\varepsilon(r) = \frac{15\varepsilon^4}{8\pi(\varepsilon^2 + r^2)^{7/2}}, \\
\phi_\varepsilon(r) = \frac{(5\varepsilon^2 - 2r^2)}{2\pi^{3/2}\varepsilon^5} e^{-r^2/\varepsilon^2}.
\]

The regularization parameter \( \varepsilon \) controls the width or spreading of the point force or point torque [41, 42]. The blob in Eq. (3) is shown for several values of \( \varepsilon \) in Figure 1.

We wish to note the similarities between the radially symmetric blob functions and radial basis functions. Radial basis functions that are often used are Laguerre-Gaussians and generalized inverse multi quadrics that can be shifted [43]. The blob in Eq. (4) corresponds to a Laguerre-Gaussian satisfying two continuous moment
conditions [41, 43]. We note that the kernel independent FMM for radial basis functions previously looked at kernels of the form $1/r^2$, $1/\sqrt{r}$, and $1/\sqrt{r^2 + 0.01^2}$ [32]. The blobs under consideration are an example of an RBF that is real analytic.

We now briefly describe the MRS given a point force $f_o$ in an unbounded 3D fluid domain. The singular point force in Eq. (1a) is replaced by a regularized force $F = f_o \psi_\varepsilon(r)$ where $r = \|x - y_o\|$ as before and $\psi_\varepsilon(r)$ is a specified radially symmetric blob function as the ones given in Eqs. (3) and (4). For a given $\psi_\varepsilon(r)$, we define the regularized (and radially symmetric) Green’s function and biharmonic function as $\Delta G_\varepsilon(r) = \psi_\varepsilon(r)$ and $\Delta B_\varepsilon(r) = G_\varepsilon(r)$, respectively. After taking the divergence of Eq. (1a) and using (1b) to simplify, the particular solution for the pressure is given as $p = f_o \cdot \nabla G_\varepsilon$. The pressure can now be used to determine the resulting flow due to the point force given as:

$$u(x) = \frac{1}{\mu} \left[ (f_o \cdot \nabla) \nabla B_\varepsilon(x - y_o) - f_o G_\varepsilon(x - y_o) \right], \quad (5)$$
the regularized Stokeslet. The solution for the pressure and velocity can then be written as follows:

\[
p(x) = [f_o \cdot (x - y_o)] \left( \frac{G'_\varepsilon(r)}{r} \right),
\]

\[
u(x) = \frac{1}{\mu} \left[ f_o \left( \frac{B'_\varepsilon(r)}{r} - G_\varepsilon(r) \right) + \left( [f_o \cdot (x - y_o)](x - y_o) \left( \frac{B''_\varepsilon(r) - B'_\varepsilon(r)}{r^3} \right) \right) \right],
\]

\[
u = \frac{1}{\mu} \left( f_o H_1(r) + [f_o \cdot (x - y_o)](x - y_o)H_2(r) \right).
\]

We emphasize that the above formulas are defined at any \( x \in \mathbb{R}^3 \). The terms \( H_1 \) and \( H_2 \) will be radially symmetric by definition and their exact form will depend on the specific regularization function \( \psi_\varepsilon \). In the case of \( N \) point forces \( \{f_j\}_{j=1}^N \) located at \( \{y_j\}_{j=1}^N \), Eq. (7) can then be written as

\[
u(x) = \frac{1}{\mu} \sum_{j=1}^N \left( f_j H_1(r_j) + [f_j \cdot (x - y_j)](x - y_j)H_2(r_j) \right),
\]

for \( r_j = ||x - y_j|| \).

In some applications, a swimmer may induce point torques in addition to point forces [42, 44], which result in fundamental solutions corresponding to Stokeslets and rotlets for the resulting fluid flow. In the case of a swimmer represented as a Kirchhoff rod [45] with regularized point forces and torques, we can write the solution to the local linear velocity \( \mathbf{u} \) and angular velocity \( \mathbf{w} \) as:

\[
u(x) = \sum_{j=1}^N \frac{1}{\mu} \left( f_j H_1(r_j) + [f_j \cdot (x - y_j)](x_i - y_i)H_2(r_j) + \frac{1}{2}[\mathbf{n}_j \times (x - y_j)]Q(r_j) \right),
\]

\[
w(x) = \sum_{j=1}^N \frac{1}{\mu} \left( \frac{1}{2}[\mathbf{n}_j \times (x - y_j)]Q(r_j) + \frac{1}{4}\mathbf{n}_j D_1(r_j) + \frac{1}{4}[\mathbf{n}_j \cdot (x - y_j)]D_2(r_j) \right),
\]

(10)
where $H_1(r), H_2(r)$ are as defined in Eq. (7), $Q(r) = G'_\varepsilon(r)/r$, $D_1(r) = \psi_\varepsilon(r) - G'_\varepsilon(r)/r$, and $D_2(r) = (G'_\varepsilon(r)/r^3) - (G''_\varepsilon(r)/r^2)$. We note that the angular velocity corresponds to regularized rotlets and dipoles and a detailed derivation can be found in [42].
3. Review of the kernel-independent fast multipole method

Consider the following many-body problem: given \( N_s \) source points \( \{y_j\}_{j=1}^{N_s} \) that exert forces \( \{f_j\}_{j=1}^{N_s} \) on \( N_e \) evaluation points \( \{x_i\}_{i=1}^{N_e} \), find the velocities

\[
u_i = u(x_i) = \sum_{j=1}^{N_s} \Phi(x_i, y_j)f_j, \quad i = 1, 2, \ldots, N_e
\]  

at the evaluation points. In (11), \( \Phi \) is the kernel (or Green’s function) of an underlying partial differential equation (PDE). When (11) arises from the simulation of fluid-structure interactions using the method of regularized Stokes (MRS), \( \{y_j\}_{j=1}^{N_s} \) are points on the structures that exert forces on the surrounding fluid, and \( \{x_i\}_{i=1}^{N_e} \) can include structure points as well as fluid points that move according to these forces. Typically, \( \{x_i\}_{i=1}^{N_e} \) consists of \( \{y_j\}_{j=1}^{N_s} \) and a number of markers in the fluid. We are particularly interested in the 3D case of (11) where \( \{x_i\}_{i=1}^{N_e}, \{f_j\}_{j=1}^{N_s} \) and \( \{y_j\}_{j=1}^{N_s} \) are in \( \mathbb{R}^3 \). In this case, the problem of computing \( \{u_i\}_{i=1}^{N_e} \) can also be viewed as a matrix-vector product

\[
\begin{bmatrix}
\mathbf{u}_1 \\
\vdots \\
\mathbf{u}_i \\
\vdots \\
\mathbf{u}_{N_e} \\
\end{bmatrix} =
\begin{bmatrix}
\Phi(x_1, y_1) \cdots \Phi(x_1, y_j) \cdots \Phi(x_1, y_{N_s}) \\
\vdots \\
\Phi(x_i, y_1) \cdots \Phi(x_i, y_j) \cdots \Phi(x_i, y_{N_s}) \\
\vdots \\
\Phi(x_{N_e}, y_1) \cdots \Phi(x_{N_e}, y_j) \cdots \Phi(x_{N_e}, y_{N_s})
\end{bmatrix}
\begin{bmatrix}
f_1 \\
\vdots \\
f_j \\
\vdots \\
f_{N_s}
\end{bmatrix},
\]  

(12)

where the matrix \( \mathcal{G} \) is \( 3N_e \times 3N_s \). It is obvious that the complexity of computing (11) or (12) is \( O(N_eN_s) \), which quickly becomes prohibitive as \( N_e \) and \( N_s \) grow.

The Fast Multipole Method (FMM) approximates (11) with complexity \( O(N_e + N_s) \). The original FMM [26–30] requires analytic factorization of the kernel:

\[
\Phi(x, y) = \sum_{n=0}^{\infty} R_n(x)S_n(y).
\]  

(13)
When $\Phi(x, y) = \|x - y\|^2$, for instance, a factorization in the form of (13) is as follows:

$$
\Phi(x, y) = (x(1) - y(1))^2 + (x(2) - y(2))^2 + (x(3) - y(3))^2
$$

$$
= (x(1))^2 \cdot 1 + (x(2))^2 \cdot 1 + (x(3))^2 \cdot 1 - 2x(1) \cdot y(1) - 2x(2) \cdot y(2) - 2x(3) \cdot y(3) + 1 \cdot (y(1))^2 + 1 \cdot (y(2))^2 + 1 \cdot (y(3))^2.
$$

For more complex kernels, factorization techniques based on Taylor series, Laurent series and spherical harmonics have been used (see [26] and the references therein). However, they are problem-specific and can be difficult to find for an arbitrary kernel, such as the ones arising from the MRS. The kernel-independent version of FMM [31, 32], on the other hand, factors $\Phi(x, y)$ numerically and can be adapted to handle the kernel of any second-order elliptic PDE as well as a kernel in the form of a radial basis function. Below we give a brief review of this method based on [26, 31, 32].

The hierarchical decomposition of the computational domain. This part is the same for both the classic and kernel-independent FMM and has been described in detail in [26]. Let $D = [x_{\text{min}}, x_{\text{max}}] \times [y_{\text{min}}, y_{\text{max}}] \times [z_{\text{min}}, z_{\text{max}}]$ be the computational domain that contains all the evaluation points $\{x_i\}_{i=1}^{N_e}$ and source points $\{y_j\}_{j=1}^{N_s}$. FMM builds an octant tree of boxes with $D$ being its root node and each level being a partition of $D$. We call $D$ itself the level-0 partition. For $L = 0, 1, 2, \ldots$, if there is a level-$L$ box that contains more than $s$ source points where $s \geq 1$ is a prescribed number, then we create Level $L + 1$ in the tree by subdividing every level-$L$ box uniformly into eight child boxes. In the resulting octant tree, there are $8^L$ boxes on the $L$th level and each leaf box (i.e., a box without any child) contains no more than $s$ source points. In Figure 2, we illustrate the first two iterations of this process applied to a 2D domain. (The main difference when it is applied to a 3D domain is that every box is subdivided uniformly into eight instead of four child boxes.) The
four boxes $B_1, B_2, B_3, B_4$ in Figure 2(b) are the children of $D$ and each of them also has four children shown in Figure 2(c). For example, the four boxes labeled with $CB1$ are the children of $B1$. Alternatively, we can perform the partition adaptively by refining only the level-$L$ boxes that contain more than $s$ source points instead of everyone of them (see [26, 31]). The adaptive approach is much more efficient when the distribution of the points is not uniform. In this paper, we focus on the uniform partition for simplicity.

![Figure 2: The partition of a 2D domain $D$ at three levels.](image)

For any box $B$ in the octant tree, the boxes that are on the same level as $B$ are classified as follows based on their location relative to $B$.

- The neighborhood of $B$ (denoted by $\mathcal{N}(B)$): the set of all the boxes that are on the same level as $B$ and share at least one vertex with $B$. (Therefore, $\mathcal{N}(B)$ includes $B$ itself.) In 3D, a box usually has $3 \times 3 \times 3 = 27$ neighbors. (In 2D, this number becomes $3 \times 3 = 9$.)

- The far field of $B$ (denoted by $\mathcal{F}(B)$): the set of all the boxes that are on the same level as $B$ and do not belong to $\mathcal{N}(B)$. In 3D, there are usually $8^L - 27$ boxes in $\mathcal{F}(B)$ for a level-$L$ box $B$. (In 2D, this number is $4^L - 9$.)

- The interaction list of $B$ (denoted by $\mathcal{I}(B)$): the subset of $\mathcal{F}(B)$ that consists of
the children of the neighbors of $B$’s parent. In 3D, there are usually $27 \times 8 - 27 = 189$ boxes in $I(B)$. (This number is $9 \times 4 - 9 = 27$ in 2D.)

In Figure 3, we show the neighborhood, far field and interaction list of a level-3 box in 2D. For any evaluation point $x_i$, assuming that it belongs to a leaf box $B$, our strategy for evaluating $u_i$ is the following: the part of $u_i$ induced by the source points in $N(B)$ will be computed directly, and the other part of it which is induced by the source points in $F(B)$ will be approximated by FMM.

![Figure 3](image)

Figure 3: $N(B)$ (all the boxes labeled with $NB$ and $B$ itself), $F(B)$ (all the boxes labeled with $FB$ or $IB$) and $I(B)$ (all the boxes labeled with $IB$) for a box $B$ on Level 3.

The equivalent surfaces/coronas. The original kernel-independent FMM [31] assigns each box $B$ in the octant tree two “equivalent surfaces”: the upward equivalent surface which is taken to be the boundary of $B$ (denoted by $\partial B$) and the downward equivalent surface which is taken to be the boundary of the neighborhood of $B$ (denoted by $\partial N(B)$). Fictitious force fields are also imposed on them with respective densities $f^{BU}(y)$ and $f^{BD}(y)$, which are defined as follows: for every point $x \in F(B)$,
the velocity at $x$ induced by the upward equivalent surface equals that induced by the source points in $B$, i.e.,

$$\forall x \in \mathcal{F}(B) : \int_{\partial B} \Phi(x,y)f_{BU}(y)dy = \sum_{y_j \in B} \Phi(x,y_j)f_j; \quad (14)$$

and for every point $x \in B$, the velocity at $x$ induced by the downward equivalent surface equals that induced by the source points in $\mathcal{F}(B)$, or equivalently,

$$\forall x \in B : \int_{\partial N(B)} \Phi(x,y)f_{BD}(y)dy = \sum_{y_j \in F(B)} \Phi(x,y_j)f_j. \quad (15)$$

In other words, the upward and downward equivalent surfaces serve as “proxies” of the source points in $B$ and those in $\mathcal{F}(B)$, respectively. In Figure 4(a), we plot both equivalent surfaces for a box $B$ in 2D.

A modified kernel-independent FMM has been developed in [32] whose main improvement over the original method is the use of equivalent “coronas” instead of surfaces. More precisely, the upward equivalent corona of $B$ is defined to be the space between $\partial B$ and a surface strictly contained in $B$, and the downward equivalent corona is the space between $\partial N(B)$ and a surface strictly contained in $\mathcal{F}(B)$. We can again define the force densities supported on the coronas to satisfy two integral equations very similar to (14) and (15). (The only difference is that both integrals will be over the equivalent coronas instead of the equivalent surfaces $\partial B$ and $\partial N(B)$.) The modified version has been shown in [32] to be more accurate for approximating (11) when $\Phi$ is a radial basis function. Figure 4(b) displays the equivalent coronas of the same box shown in Figure 4(a). In all of our numerical experiments, we use the equivalent coronas for their accuracy; nonetheless, since the change to the original method caused by the use of coronas is only technical, we continue to use the equivalent surfaces in the rest of this review for simplicity.
Figure 4: 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 $6 \times 6$ 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 ■.

**Numerical approximation of the force densities.** The integral equations (14) and (15) will be solved numerically for the force densities $f^{BU}$ and $f^{BD}$ supported on the equivalent surfaces. First, we discretize the integrals on the left-hand sides of both equations using a quadrature rule. Let \( \{q^{BU}_m\}_{m=1}^{N_q} \) be the quadrature points on the upward equivalent surface $\partial B$ and \( \{q^{BD}_m\}_{m=1}^{N_q} \) be the quadrature points on the downward equivalent surface $\partial \mathcal{N}(B)$ (see again Figure 4), where $N_q$ denotes the number of quadrature points. Next, we choose a finite number of samples for $x$ in both (14) and (15). For convenience, the samples of $x$ in (14) are chosen to be \( \{q^{BD}_m\}_{m=1}^{N_q} \) whereas the samples of $x$ in (15) are chosen to be \( \{q^{BU}_m\}_{m=1}^{N_q} \). As a result, we obtain the following two $3N_q \times 3N_q$ linear systems of equations of \( \{f^{BU}(q^{BU}_m)\}_{m=1}^{N_q} \).
and \( \{f^{BD}(q_m^{BD})\}_{m=1}^{N_q} \), respectively:

\[
\sum_{m=1}^{N_q} \omega_m^{BU} \Phi(q_k^{BD}, q_m^{BU}) f^{BU}(q_m^{BU}) = \sum_{y_j \in B} \Phi(q_k^{BD}, y_j) f_j, \quad k = 1, 2, \ldots, N_q
\]  

and

\[
\sum_{m=1}^{N_q} \omega_m^{BD} \Phi(q_k^{BU}, q_m^{BD}) f^{BD}(q_m^{BD}) = \sum_{y_j \in F(B)} \Phi(q_k^{BU}, y_j) f_j, \quad k = 1, 2, \ldots, N_q,
\]

where \( \{\omega_m^{BU}\}_{m=1}^{N_q}, \{\omega_m^{BD}\}_{m=1}^{N_q} \in \mathbb{R}^{N_q} \) are coefficients determined by the quadrature rule used. Once (16) and (17) are solved, the velocity at any point \( x \in F(B) \) induced by the source points in \( B \) can be estimated by

\[
\sum_{m=1}^{N_q} \omega_m^{BU} \Phi(x, q_m^{BU}) f^{BU}(q_m^{BU})
\]

and similarly, the velocity at any point \( x \in B \) induced by the source points in \( F(B) \) is approximately

\[
\sum_{m=1}^{N_q} \omega_m^{BD} \Phi(x, q_m^{BD}) f^{BD}(q_m^{BD}).
\]

In FMM terminology, (18) and (19) are called the multiple expansion and local expansion of box \( B \), respectively. The advantage of computing the two velocities using (18) and (19) is to avoid direct interaction between the source points and the evaluation points, which is the key to reducing the computational complexity of (11). Instead, they will interact indirectly through the quadrature points. It remains to show how to compute the right-hand sides of (16) and (17) efficiently.

**The upward passing step.** In this step, we traverse the octant tree from the leaf level up to Level 2 and solve (16) for every box on these levels. For every leaf box, we simply evaluate the right-hand side of (14) directly. For a non-leaf box \( B \), let \( \{C_{\ell}\}_{\ell=1}^{8} \) be the children of \( B \) and assume that \( \{f^{CU}(q_m^{CU})\}_{m=1}^{N_q} \) are known for all \( \ell \).
Since \( \{q_k^{BD}\}_{k=1}^{N_q} \subset \mathcal{F}(C_\ell) \) for every \( \ell \), according to (18), the right-hand side of (16) can be approximated by

\[
\sum_{\ell=1}^{8} \sum_{m=1}^{N_q} \omega_m^{C_\ell U} \Phi(q_k^{BD}, q_m^{C_\ell U}) f^{C_\ell U}(q_m^{C_\ell U}).
\]  

(20)

In FMM terminology, this process of “merging” the information of the children to gather information about their parent is referred to as the Multipole-to-Multipole translation (or M2M translation). We summarize the upward passing step in Algorithm 1. Assume Level \( L_{\text{max}} \) is the leaf level in the octant tree.

**Algorithm 1:** The upward passing step

For \( L = L_{\text{max}}, L_{\text{max}}-1, \ldots, 2 \)

For every box \( B \) on Level \( L \):

1. If \( L = L_{\text{max}} \):
   
   Compute the right-hand side of (16) directly.

2. Else:
   
   Compute the right-hand side of (16) by (20).

3. Compute \( \{f^{BU}(q_m^{BU})\}_{m=1}^{N_q} \) by solving the linear system (16).

The complexity of the upward passing step can be calculated as follows. Recall that \( s \) is a fixed number that represents the maximum number of source points allowed in each leaf box. Assume that the distribution of the source points is close to being uniform in the following sense: the average number of source points in each box on Level \( L_{\text{max}}-1 \) (the second finest level, which is also the parent level of the leaf level) is strictly greater than \( s \), i.e., \( s \cdot 8^{L_{\text{max}}-1} < N_s \). This implies that

\[
L_{\text{max}} < \log_8 \left( \frac{N_s}{s} \right) + 1 = \log_8 N_s + 1 - \log_8 s \leq \lceil \log_8 N_s \rceil + 1,
\]

where \( \lceil r \rceil \) is the smallest integer greater than or equal to a real number \( r \). For each leaf box, since both \( s \) and \( N_q \) are constants independent of \( N_e \) and \( N_s \), the cost of
computing the right-hand side of (16) directly is also a constant $\gamma_1$. Likewise, for every non-leaf box, the cost of computing the right-hand side of (16) by (20) is a constant $\gamma_2$ as well. In addition, the cost of solving (16) or (17) for any box is a constant $\gamma_3$ determined by $N_q$. To sum up, the total cost of Algorithm 1 is

$$(\gamma_1 + \gamma_3) \cdot 8^{L_{\max}} + \sum_{L=2}^{L_{\max}-1} (\gamma_2 + \gamma_3) \cdot 8^L$$

$$\leq (\max\{\gamma_1, \gamma_2\} + \gamma_3) \cdot \sum_{L=2}^{L_{\max}} 8^L < (\max\{\gamma_1, \gamma_2\} + \gamma_3) \cdot \sum_{L=2}^{[\log_8 N_s]+1} 8^L$$

$$< (\max\{\gamma_1, \gamma_2\} + \gamma_3) \cdot \frac{64N_s - 1}{8 - 1} = O(N_s).$$

The downward passing step. Contrary to the previous step, in this step, we traverse the octant tree from Level 2 down to the leaf level and solve (17) for every box on these levels. Note that for any box $B$ and assume that $P$ is its parent, by the definitions of $F(B)$ and $I(B)$,

$$F(B) = I(B) \cup F(P) \quad \text{and} \quad I(B) \cap F(P) = \emptyset.$$ 

This indicates that the right-hand side of (17) can be rewritten as the following sum:

$$\sum_{y_j \in I(B)} \Phi(q_{BU_k}^B, y_j) f_j + \sum_{y_j \in F(P)} \Phi(q_{BU_k}^B, y_j) f_j.$$ 

Since $\{q_{BU_k}^B\}_{k=1}^{N_q} \subset F(A)$ for any box $A \in I(B)$, by (18), the first sum in (21) can be approximated by

$$\sum_{A \in I(B)} \sum_{m=1}^{N_q} \omega_{AU}^B \Phi(q_{BU_k}^B, q_{AU_m}^A) f_{AU}(q_{AU_m}^A),$$

where $\{f_{AU}(q_{AU_m}^A)\}_{m=1}^{N_q}$ have already been found in the upward passing step. The process of passing information from the interaction list of a box to the box itself is called the Multipole to Local translation (or M2L translation) in FMM. For any box
B on Level 2, $F(P) = \emptyset$ and thus $F(B) = I(B)$ (see Figure 2). Consequently, the right-hand side of (17) can be approximated by (22) alone. For any box $B$ on a level greater than 2, assume $\{f^{PD}(q_{m}^{PD})\}_{m=1}^{N_{q}}$ are known. The first sum in (21) can again be computed by (22); and since $\{q_{k}^{BU}\}_{k=1}^{N_{q}} \subset P$, by (19), the second sum in (21) can be estimated by

$$
\sum_{m=1}^{N_{q}} \omega_{m}^{PD} \Phi(q_{k}^{BU}, q_{m}^{PD}) f^{PD}(q_{m}^{PD}).
$$

Then the right-hand side of (17) can be obtained as the sum of (22) and (23). A box “inheriting” information from its parent like this is referred to as the Local to Local translation (or L2L translation) in the language of FMM. The downward passing step is summarized in Algorithm 2.

**Algorithm 2:** The downward passing step

For $L = 2, 3, \ldots, L_{\text{max}}$

For every box $B$ on Level $L$:

1. If $L = 2$:
   
   Compute the right-hand side of (17) by (22).

2. Else:
   
   Compute (22), (23) and sum them up to get the right-hand side of (17).

3. Compute $\{f^{BD}(q_{m}^{BD})\}_{m=1}^{N_{q}}$ by solving the linear system (17).

We now analyze the complexity of the downward passing step. The cost of computing (22) for any box is bounded above by a constant $\gamma_4$ since the interaction list of a box contains at most 189 boxes$^1$, and computing (23) entails a fixed cost $\gamma_5$ for

---

$^1$Recall that the interaction list of a box $B$ consists of the children of the neighbors’ of $B$’s parent that are not $B$’s neighbors and therefore has size at most $27 \times 8 - 27 = 189$. 

19
any box on Level $L$ where $3 \leq L \leq L_{\text{max}}$. The total cost of Algorithm 2 is therefore no more than

$$
(\gamma_3 + \gamma_4) \cdot 8^2 + \sum_{L=3}^{L_{\text{max}}} (\gamma_3 + \gamma_4 + \gamma_5) \cdot 8^L
$$

$$
< (\gamma_3 + \gamma_4 + \gamma_5) \cdot 8^L < (\gamma_3 + \gamma_4 + \gamma_5) \cdot \sum_{L=2}^{\lceil \log_8 N_s \rceil + 1} 8^L
$$

$$
< (\gamma_3 + \gamma_4 + \gamma_5) \cdot \frac{64N_s - 1}{8 - 1} = \mathcal{O}(N_s),
$$

where $\gamma_3$ is again the cost of solving (16) or (17) for any box.

**Evaluation.** In this step, the velocities $\{u_j\}_{j=1}^{N_e}$ at the evaluation points are finally computed. Note that unlike in the previous two steps where we need to traverse the entire octant tree, evaluation is only performed at the leaf level. For each evaluation point $y_j$ in a leaf box $B$, we first calculate directly the part of $u_j$ induced by the source points in $\mathcal{N}(B)$, i.e.,

$$
\sum_{x_i \in \mathcal{N}(B)} \Phi(x_i, y_j) f_i; \quad (24)
$$

and then we use (19) to compute the part of $u_j$ induced by the source points in $\mathcal{F}(B)$:

$$
\sum_{m=1}^{N_q} \omega_m^{BD} \Phi(q_m^{BD}, y_j) f^{BD}(q_m^{BD}); \quad (25)
$$

where $\{f^{BD}(q_m^{BD})\}_{m=1}^{N_q}$ have been produced during the downward passing step. It is straightforward from their definitions that $\mathcal{F}(B) \cup \mathcal{N}(B) = \mathcal{D}$ and $\mathcal{F}(B) \cap \mathcal{N}(B) = \emptyset$, therefore, $u_j$ equals the sum of (24) and (25).

Since there are at most $27s$ source points in $\mathcal{N}(B)$ by the construction of the octant tree, the cost of computing (24) is bounded above by a constant $\gamma_6$. The cost of evaluating (25) only depends on $N_q$ and is therefore a fixed constant $\gamma_7$. Since there are a total of $N_e$ evaluation points, the cost of the evaluation step is no more
than

\[(\gamma_6 + \gamma_7) \cdot N_e = \mathcal{O}(N_e).\]

To sum up, the total cost of kernel-independent FMM is \(\mathcal{O}(N_e + N_s)\).
4. The “dumbbell” example

The “dumbbell” model was first introduced in [46] as a minimal model for swimming microorganisms. In this model, each microorganism is represented by a pair of beads or particles (hence the name “dumbbell”) which mimic its cell body and flagella, respectively. In the original model, force balance between the flagellum force, drag forces, and a force due to rigidity of the rod connecting the beads was accounted for. The MRS has been applied to study the flow field generated by the collective swimming of microorganisms described by a simplified version of the dumbbell model [47].

We will test the performance of the kernel-independent FMM at approximating (11) arising from the MRS applied to an example considered in [47]. In this example, a large number of microorganisms are uniformly but randomly distributed in a 3D Stokes fluid, and the two particles representing each microorganism exert forces of unit norm and opposite directions pushing them away from each other. Let $N_o$ denote the number of microorganisms. Then there are $N_s = 2N_o$ source points in total. By Eq. (7), the kernel for this example is

$$\Phi(x, y) = \frac{1}{\mu} \left(H_1(r)I + H_2(r)(x - y)(x - y)^T\right),$$

where $r = \|x - y\|$, $\mu$ is the viscosity of the fluid, and $I$ is the $3 \times 3$ identity matrix. The scalar functions $H_1(r)$, $H_2(r)$ are as defined in Eq. (7) and their exact forms depend on the blob function used in the MRS. In our numerical experiments, the blob function given by Eq. (3) is used at every source point $y$, which gives

$$H_1(r) = \frac{2\varepsilon^2 + r^2}{8\pi(r^2 + \varepsilon^2)^{3/2}}, \quad H_2(r) = \frac{1}{8\pi(r^2 + \varepsilon^2)^{3/2}}.$$  

In order to explore the performance of the kernel-independent FMM applied to this problem, we perform a series of numerical experiments where we vary the number
of microorganisms \( (N_o) \), the number of quadrature points \( (N_q) \) used to discretize the equivalent coronas, the regularization parameter \( \varepsilon \) in the MRS as well as the number of CPU cores \( (N_c) \) used. The computational domain \( \mathcal{D} \) is a \( 10 \times 10 \times 10 \) box and stays fixed, and the trapezoidal rule over a uniform grid of quadrature points is used to discretize both integral equations (14) and (15). All the numerical experiments are run in MATLAB (version 2014a) on an Intel Xeon E5-2680 v2 CPU. The run time is calculated using the \texttt{tic} and \texttt{toc} command pair in MATLAB.

The computational complexity. As analyzed in Section 3, direct summation has computational complexity \( \mathcal{O}(N_e N_s) \) whereas the kernel-independent FMM has complexity \( \mathcal{O}(N_s + N_e) \). In the special case where \( N_e = N_s \) (i.e., if we only compute the velocities for the particles and no extra fluid markers), the two complexities become \( \mathcal{O}(N_s^2) \) and \( \mathcal{O}(N_s) \), respectively. The aim of the following set of experiments is to verify both computational complexities by increasing \( N_o \). The parameters used are summarized in Table 1.

As shown in Table 1, starting from the second \( N_o \), each \( N_o \) is 8 times as large as the previous one; and every time \( N_o \) increases by a factor of 8, we also increase \( L_{\text{max}} \) by 1 accordingly so that the number of particles \( s \) in each leaf box stays roughly the same. We compute the velocities of all the particles using the kernel-independent FMM and record its run time \( T_{\text{fmm}} \), which includes the time required for constructing the octant tree, sorting all the particles into the boxes, the upward and downward passing as well as the actual evaluation. Since the run time of direct summation increases drastically as \( N_o \) grows, we only use it to calculate the velocities of 1000 randomly sampled particles, record the run time \( T_{\text{dir}}^{\text{sample}} \) and estimate the run time of the full simulation as

\[
T_{\text{dir}}^{\text{est}} = T_{\text{dir}}^{\text{sample}} \cdot \frac{N_s}{1000}.
\]
Table 1: A summary of the parameters used in the complexity test.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>the computational domain</td>
<td>$[-5, 5] \times [-5, 5] \times [0, 10]$</td>
</tr>
<tr>
<td>$N_o$</td>
<td>number of microorganisms</td>
<td>5000, 40000, 320000, 2560000, or 20480000</td>
</tr>
<tr>
<td>$N_s$</td>
<td>number of particles</td>
<td>10000, 80000, 640000, 5120000, or 40960000</td>
</tr>
<tr>
<td>$L_{max}$</td>
<td>maximum level of refinement of $D$ in the FMM</td>
<td>2, 3, 4, 5, or 6</td>
</tr>
<tr>
<td>$N_q$</td>
<td>number of quadrature points</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>(a $4 \times 4 \times 4$ uniform Cartesian grid is used)</td>
<td></td>
</tr>
<tr>
<td>$\ell$</td>
<td>length of each microorganism</td>
<td>0.02</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>the regularization parameter in the MRS</td>
<td>0.02</td>
</tr>
<tr>
<td>$N_c$</td>
<td>number of CPU cores</td>
<td>1</td>
</tr>
</tbody>
</table>

In Figure 5, we plot both $T_{fmm}$ and $T_{est}$ corresponding to the increasing sequence of $N_o$ listed in Table 1. (Note that this is a log log plot.) Two lines with respective slopes 1 and 2 are also displayed in the same figure for reference. Figure 5 shows that the curve representing $T_{est}$ is almost parallel to the line with slope 2, confirming that the complexity of direct summation is $O(N_s^2)$. The curve corresponding to $T_{fmm}$, on the other hand, is close to being parallel to the line with slope 1, which verifies that the complexity of the kernel-independent FMM is $O(N_s)$. The run time of both methods is also reported in Table 2. Based on the complexity analysis, we expect $T_{fmm}$ to grow by a factor of 8 and $T_{est}$ a factor of 64 as $N_o$ grows by a factor of 8,
which can be seen in Table 2 in the asymptotic regime.

Figure 5: The log log plot of run time vs. number of particles.

The accuracy of the FMM. We also monitor the errors of the kernel-independent FMM in the sequence of experiments described above. To account for the accuracy of this method at approximating the velocities of both the microorganisms and the fluid, we compute the following relative error:

$$E_{db}^{\text{u}} = \sqrt{\frac{\sum_{n=1}^{2000} \| u_{fmm}^{i_n} - u_{i_n} \|^2}{\sum_{n=1}^{2000} \| u_{i_n} \|^2}}$$  \hspace{1cm} (29)

where the subscripts $\{i_n\}_{n=1}^{2000}$ correspond to an ensemble of evaluation points consisting of the 1000 particles sampled in the previous set of experiments and 1000 random points in the fluid. (The reason that we do not include all the particles in (29) is because direct summation is applied to the sampled particles only.) As shown in Table 2, the relative error (29) is between $O(10^{-5})$ and $O(10^{-3})$ in our experiments and grows with the density of the microorganisms.

We conclude that the kernel-independent FMM is able to achieve significant savings in computational cost if the number of microorganisms is large enough and if
Table 2: Comparison of the kernel-independent FMM and direct summation for (11).

<table>
<thead>
<tr>
<th>$N_o$</th>
<th>$N_s$</th>
<th>$L_{\text{max}}$</th>
<th>$T_{fmm}$ (second)</th>
<th>$T_{fmm}^{\text{est}}$ ratio</th>
<th>$T_{dir}^{\text{est}}$ (second)</th>
<th>$T_{dir}$ ratio</th>
<th>$E_{\text{db}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5000</td>
<td>10000</td>
<td>2</td>
<td>5</td>
<td>—</td>
<td>12</td>
<td>—</td>
<td>4.23 $10^{-5}$</td>
</tr>
<tr>
<td>40000</td>
<td>80000</td>
<td>3</td>
<td>53</td>
<td>10.60</td>
<td>494</td>
<td>41.17</td>
<td>2.10 $10^{-4}$</td>
</tr>
<tr>
<td>320000</td>
<td>640000</td>
<td>4</td>
<td>537</td>
<td>10.13</td>
<td>20390</td>
<td>41.28</td>
<td>7.33 $10^{-4}$</td>
</tr>
<tr>
<td>256000</td>
<td>512000</td>
<td>5</td>
<td>3776</td>
<td>7.03</td>
<td>1490300</td>
<td>73.09</td>
<td>2.18 $10^{-3}$</td>
</tr>
<tr>
<td>2048000</td>
<td>4096000</td>
<td>6</td>
<td>28727</td>
<td>7.61</td>
<td>93653000</td>
<td>62.84</td>
<td>4.88 $10^{-3}$</td>
</tr>
</tbody>
</table>

the requirement on accuracy is not very stringent. We note that as the number of microorganisms is increased, the computational domain is fixed; different error may be observed for inhomogeneous source distributions in space. Since the relative error will depend on a number of different parameters, in the following numerical experiments, we fix $N_o = 320000$ (and thus $N_s = 640000$) and explore the impacts of other parameters on the performance of the kernel-independent FMM.

The regularization parameter. We vary the regularization parameter $\varepsilon$ in (26) and always make the length $\ell$ of each microorganism equal $\varepsilon$. The relative error (29) associated with each $\varepsilon$ is reported in Table 3, which illustrates that the kernel independent FMM becomes more accurate as $\varepsilon$ decreases (that is, when the shape of the blob function $\psi_\varepsilon(r)$ gets “taller”).

The number of quadrature points. We also examine the performance of kernel-independent FMM as the Cartesian grids used to discretize the equivalent coronas are refined. More specifically, we consider three uniform meshes: $4 \times 4 \times 4$, $5 \times 5 \times 5$ and $6 \times 6 \times 6$, which consist of 64, 124 and 208 quadrature points, respectively. The
Table 3: The relative error as the regularization parameter $\varepsilon$ varies ($N_o = 320000$).

<table>
<thead>
<tr>
<th>$\varepsilon$ and $\ell$</th>
<th>$E_u^{db}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>$9.12 \times 10^{-5}$</td>
</tr>
<tr>
<td>0.01</td>
<td>$2.62 \times 10^{-4}$</td>
</tr>
<tr>
<td>0.02</td>
<td>$7.33 \times 10^{-4}$</td>
</tr>
<tr>
<td>0.04</td>
<td>$1.76 \times 10^{-3}$</td>
</tr>
<tr>
<td>0.08</td>
<td>$2.84 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

corresponding dimensions of the linear systems (16) and (17) are 192, 372 and 624. As shown in Table 4, the kernel-independent FMM becomes more costly when a finer mesh is used because the dimension of the linear systems that need to be solved becomes larger; in the meantime, it gets more accurate since the two integrals in (14) and (15) are better approximated by the quadrature rule. Note that even when the finest grid is used, the kernel-independent FMM is still significantly more efficient than direct summation for this particular choice of $N_o$.

Table 4: The relative error and run time of the kernel independent FMM associated with different grids for the equivalent coronas ($N_o = 320000$).

<table>
<thead>
<tr>
<th>Cartesian grid</th>
<th>$N_q$</th>
<th>$T_{fmm}$</th>
<th>$E_u^{db}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4 \times 4 \times 4$</td>
<td>64</td>
<td>537</td>
<td>$7.33 \times 10^{-4}$</td>
</tr>
<tr>
<td>$5 \times 5 \times 5$</td>
<td>124</td>
<td>1145</td>
<td>$2.95 \times 10^{-4}$</td>
</tr>
<tr>
<td>$6 \times 6 \times 6$</td>
<td>208</td>
<td>2485</td>
<td>$1.07 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

The number of CPU cores. So far all the numerical results are obtained on a single CPU core. We also parallelize the kernel-independent FMM code using the Parallel Computing Toolbox in MATLAB. In the upward or downward passing step,
we need to traverse the octant tree in a particular order; however, the loop over all the boxes on the same level is parallelizable. In the evaluation step, the loop over all the leaf boxes can also be parallelized. We use the \texttt{parfor} command to parallelize these loops. The numbers of core(s) that we use are 1, 2, 4, and 8, and the run time of each case is shown in Table 5. We also compute the strong scaling efficiency for each case, which is defined to be the following percentage:

\[
\text{strong scaling efficiency} = \frac{\text{run time when 1 core is used}}{N_c \cdot \text{run time when } N_c \text{ core(s) are used}} \cdot 100%.
\]

Table 5 demonstrates that the efficiency decreases by around 10% every time we double the number of cores. The main obstacle to increasing the efficiency of the \texttt{parfor} command lies in the way memory is accessed by the cores: instead of accessing the data stored in a shared memory via pointers, every core has to make a local copy of the data that it will be using. This increases the overhead of parallel computing.

Table 5: Run time of the kernel-independent FMM when multiple cores are used \((N_o = 320000)\).

<table>
<thead>
<tr>
<th>(N_c)</th>
<th>(T_{fmm})</th>
<th>Optimal (T_{fmm})</th>
<th>Strong scaling efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>537</td>
<td>537</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>307</td>
<td>268</td>
<td>87%</td>
</tr>
<tr>
<td>4</td>
<td>176</td>
<td>134</td>
<td>76%</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>67</td>
<td>67%</td>
</tr>
</tbody>
</table>
5. The Kirchhoff rod example

In this example, we simulate the collective swimming of a large number of elastic rods immersed in a 3D Stokes fluid. They can be viewed as a floor of cilia or the flagella of a large number of microorganisms. The elastic rods are modeled using an unconstrained version of the Kirchhoff rod theory [45], and they translate, bend and twist in the fluid. We continue to investigate the effectiveness of the kernel-independent FMM applied to the MRS.

Compared to the previous example, this is a more challenging problem in the following ways. In the “dumbbell” example, translation is the only type of motion present since the flagellum of each microorganism is simply represented by a point. Here, the flagella are modeled as elastic rods that will bend and twist. Thus, besides forces and linear velocities, torques and angular velocities also need to be considered. This results in a more complicated kernel. Furthermore, unlike in the previous example where we only compute the linear velocities at one instant, in this example, we simulate the motion of the rods over a period of time, which in turn entails solving a sequence of $N$-body problems. It not only significantly increases the run time regardless of whether direct summation or the FMM is used but also has more stringent requirement on the accuracy of the latter since the numerical errors will propagate.

In our simulation, each rod is represented by its centerline which is a 3D parametric curve of finite length and is initially straight. The points on the rod are Lagrangian variables parametrized by the arclength from the base point of the rod. To capture the twisting of the rod, we associate each point on the rod with three orthonormal triads: one is tangent to the rod whereas the other two are along the cross section at that point. In the previous example, each source point exerts a random
force to the surrounding fluid. In this example, the forces and torques exerted by the rods are computed as follows. We assume that each rod has an intrinsic curvature that it tries to achieve by constantly adjusting its internal force and torque. In order to compute them, we discretize each rod into a number of segments and treat them as small springs connected by the grid points. At every time step, the elastic energy stored in each segment due to the difference between its current and preferred shapes can be computed, from which we can then back out the internal force and torque of that segment. The force and torque exerted by the fluid to the rods at each grid point can be derived by balancing the linear and angular momentums at that point. Finally, the same force and torque in the opposite directions are applied by the grid point to the fluid. This method has been used in previous work [42, 48, 49] where a detailed description of it can be found.

Let there be \( M + 1 \) uniformly spaced grid points on each rod dividing it into \( M \) equal segments. Let \( L \) be the length of the rods, then \( \Delta s = L/M \) gives the length of each segment. Let \( \{ y^n_j \}_{j=1}^{Ns} \) denote the locations of the grid points at the \( n \)th time step. They are arranged in such a way that \( \{ y^n_j \}_{j=(k-1)(M+1)+1}^{k(M+1)} \) correspond to the grid points on the \( k \)th rod in increasing order of the arclength from the base of the rod. If there are \( Nr \) rods, then \( N_s = Nr(M+1) \). Let \( \{ b^n_k \}_{k=1}^{Nr} \) denote the base points of the rods at the \( n \)th time step, i.e., \( b^n_k = y^n_{(k-1)(M+1)+1} \). Then the initial locations of the grid points \( \{ y^0_j \}_{j=(k-1)(M+1)+1}^{k(M+1)} \) on the \( k \)th rod can be written as

\[
\begin{cases}
\{ b^0_k + q \cdot \begin{bmatrix} 0 \\ 0 \\ \Delta s \end{bmatrix} \}_{q=0}^M 
\end{cases}
\]

Let \( \{ f^n_j \}_{j=1}^{Ns} \) and \( \{ n^n_j \}_{j=1}^{Ns} \) be the forces and torques exerted by the grid points to the fluid, the computation of which has been described above. By Eqs. (9) and (10), at
the \( n \)th time step, the linear velocities \( \{ u^n_i \}_{i=1}^{N_e} \) and angular velocities \( \{ w^n_i \}_{i=1}^{N_e} \) at the evaluation points \( \{ x^n_i \}_{i=1}^{N_e} \) given by the MRS are

\[
\begin{bmatrix}
  u^n_i \\
  w^n_i \\
  \vdots \\
  u^n_{i_e} \\
  w^n_{i_e}
\end{bmatrix} = \sum_{j=1}^{N_s} \Psi \left( x^n_i, y^n_j \right) \begin{bmatrix}
  f^n_j \\
  n^n_j \\
  \vdots \\
  f^n_{N_s} \\
  n^n_{N_s}
\end{bmatrix}, \quad i = 1, 2, \ldots, N_e,
\] (30)

where the kernel

\[
\Psi(x, y) = \frac{1}{\mu} \begin{bmatrix}
  H_1(r)I + H_2(r)(x - y)(x - y)^T & \frac{1}{2}Q(r)[x - y]^T \\
  \frac{1}{2}Q(r)[x - y]^T & \frac{1}{4}D_1(r)I + \frac{1}{4}D_2(r)(x - y)(x - y)^T
\end{bmatrix}.
\] (31)

The scalar functions \( Q(r), D_1(r), D_2(r) \) in (31) are defined in Section 2 (right after Eq. (10)) and depend on the blob function used; and for any vector \( v = [v_1, v_2, v_3]^T \in \mathbb{R}^3 \), the matrix \( [v]_x \) denotes the \( 3 \times 3 \) skew-symmetric matrix

\[
\begin{bmatrix}
  0 & -v_3 & v_2 \\
  v_3 & 0 & -v_1 \\
  -v_2 & v_1 & 0
\end{bmatrix},
\]

that is, the cross product of \( v \) and any vector \( r \in \mathbb{R}^3 \) is \( v \times r = [v]_x \cdot r \) and the cross product of \( r \) and \( v \) is \( r \times v = [v]^T_\times \cdot r \). Computing (30) is equivalent to computing the matrix-vector product

\[
\begin{bmatrix}
  u^n_1 \\
  w^n_1 \\
  \vdots \\
  u^n_{i_e} \\
  w^n_{i_e} \\
  \vdots \\
  u^n_{N_e} \\
  w^n_{N_e}
\end{bmatrix} = \sum_{j=1}^{N_s} \begin{bmatrix}
  \Psi(x^n_1, y^n_j) & \cdots & \Psi(x^n_1, y^n_{N_s}) \\
  \vdots & \cdots & \cdots & \cdots \\
  \Psi(x^n_{i_e}, y^n_j) & \cdots & \Psi(x^n_{i_e}, y^n_{N_s}) \\
  \vdots & \cdots & \cdots & \cdots \\
  \Psi(x^n_{N_e}, y^n_j) & \cdots & \Psi(x^n_{N_e}, y^n_{N_s})
\end{bmatrix} \begin{bmatrix}
  f^n_1 \\
  n^n_1 \\
  \vdots \\
  f^n_{i_e} \\
  n^n_{i_e} \\
  \vdots \\
  f^n_{N_e} \\
  n^n_{N_e}
\end{bmatrix}.
\] (32)
where the matrix $K^n$ is $6N_e \times 6N_s$. In this section, we continue to use the blob function $(3)$, which gives $(27)$ and $(33)$:

$$Q(r) = \frac{5\varepsilon^2 + 2r^2}{8\pi(r^2 + \varepsilon^2)^{5/2}}, \quad D_1(r) = \frac{10\varepsilon^4 - 7\varepsilon^2r^2 - 2r^4}{8\pi(r^2 + \varepsilon^2)^{7/2}}, \quad D_2(r) = \frac{21\varepsilon^2 + 6r^2}{8\pi(r^2 + \varepsilon^2)^{7/2}}. \quad (33)$$

We will focus on the dynamics of the rods and thus choose $\{y^n_j\}_{j=1}^{N_s} = \{x^n_i\}_{i=1}^{N_e}$, though extra evaluation points can easily be included as in the previous example.

At the $n$th time step, let $\{(D_1^n)_i, (D_2^n)_i, (D_3^n)_i\}$ denote the three orthonormal triads associated with $x^n_i$, where $(D_3^n)_i$ is tangent to the rod at $x^n_i$ and $(D_1^n)_i, (D_2^n)_i$ are parallel to the cross section at $x^n_i$. Let $\Delta t$ be the time step. The method proposed in [42] for tracking the movement of Kirchhoff rods with an intrinsic curvature and twist immersed in a 3D Stokes fluid is outlined in Algorithm 3.

The computational complexity of both steps 1 and 3 of Algorithm 3 is $O(N_s)$ and that of step 2 is $O(N_s^2)$ if $(30)$ is computed directly. We want to increase the efficiency of this algorithm by reducing the complexity of step 2 to $O(N_s)$ using the kernel-independent FMM. The changes that need to be made to the FMM algorithm are only technical when it is applied to $(30)$ instead of $(11)$. The main difference is that besides the two fictitious forces fields, we also need to impose two fictitious torque fields on every box. The two integral equations $(14)$ and $(15)$ become

$$\forall x \in F(B) : \int_{\partial B} \Psi(x, y) \begin{bmatrix} f_{n_{BU}}(y) \\ n_{BU}(y) \end{bmatrix} dy = \sum_{y_j \in B} \Psi(x, y_j) \begin{bmatrix} f_j \\ n_j \end{bmatrix}, \quad (35)$$

and

$$\forall x \in B : \int_{\partial N(B)} \Psi(x, y) \begin{bmatrix} f_{n_{BD}}(y) \\ n_{BD}(y) \end{bmatrix} dy = \sum_{y_j \in F(B)} \Psi(x, y_j) \begin{bmatrix} f_j \\ n_j \end{bmatrix}, \quad (36)$$

which lead to linear systems

$$\sum_{m=1}^{N_q} \omega_{m}^{BU} \Psi(q_k^{BU}, q_m^{BU}) \begin{bmatrix} f_{n_{BU}(q_m^{BU})} \\ n_{BU}(q_m^{BU}) \end{bmatrix} = \sum_{y_j \in B} \Psi(q_k^{BU}, y_j) \begin{bmatrix} f_j \\ n_j \end{bmatrix}, \quad k = 1, 2, \ldots, N_q \quad (37)$$
Algorithm 3: Time stepping of the Kirchhoff rod model

For $n = 0, 1, 2, \cdots, N_t$:

1. Compute using central difference the forces $\{f_i^n\}_{i=1}^{N_s}$ and the torques $\{n_i^n\}_{i=1}^{N_s}$ based on the difference between the current shapes of the rods given by $\{x_i^n\}_{i=1}^{N_s}$ and $\{(D^k_i)^n\}_{i=1}^{N_s}$ ($k = 1, 2, 3$) and their desired shape specified by an intrinsic curvature and twist.

2. Compute the linear velocities $\{u_i^n\}_{i=1}^{N_s}$ and the angular velocities $\{w_i^n\}_{i=1}^{N_s}$ from (30), or equivalently, (32).

3. Update the locations of the grid points and the orientation of the triads using forward Euler’s method: for $i = 1, 2, \cdots, N_s$,

$$x_i^{n+1} \leftarrow x_i^n + u_i^n \cdot \Delta t, \quad (D^k_i)^{n+1} \leftarrow R \left( \frac{w_i^n}{\|w_i^n\|}, \|w_i^n\| \cdot \Delta t \right) \cdot (D^k_i)^n,$$

where $R(e, \theta) = (\cos \theta)I + (1 - \cos \theta)e e^T + (\sin \theta)\{e\}_x$ is a $3 \times 3$ orthogonal matrix which, when applied to any vector $v \in \mathbb{R}^3$, rotates it about the unit vector $e \in \mathbb{R}^3$ through an angle of $\theta$. 

and
\[
\sum_{m=1}^{N_q} \omega_m^{BD} \psi(q_k^{BU}, q_m^{BD}) \begin{bmatrix} f^{BD}(q_k^{BD}) \\ n^{BD}(q_m^{BD}) \end{bmatrix} = \sum_{y_j \in \mathcal{F}(B)} \psi(q_k^{BU}, y_j) \begin{bmatrix} f_j \\ n_j \end{bmatrix}, \quad k = 1, 2, \ldots, N_q
\]
(38)
of dimension $6N_q \times 6N_q$ instead of $3N_q \times 3N_q$ as in the “dumbbell” example. The other equations in Section 1 need to be adjusted in a similar fashion to account for the new kernel. Note that all the fictitious force and torque fields will vary with time since the locations of the source points as well as the forces and torques that they apply to the fluid are time-dependent. In Eqs. (35)-(38), we deliberately drop the superscript $n$ that specifies the time step to simplify the notation.

A summary of the values of the parameters used in our simulation can be found in Table 6. The computational domain $\mathcal{D}$ is chosen to be large enough so that none of the grid points on the rods will move outside of $\mathcal{D}$ throughout all the simulations. In the kernel-independent FMM, three levels of partition are used for $\mathcal{D}$ and a $5 \times 5 \times 5$ uniform grid with $N_q = 124$ grid points is used to discretize each equivalent corona, which results in linear systems (37) and (38) of dimension $744 \times 744$. Initially, we place 225 straight rods of length $9 \, \mu m$ in the computational domain. Their projections onto the $xy$-plane form a uniform $15 \times 15$ grid and are shown in Figure 6. In our numerical experiments, we will vary the $z$ coordinates of the base points $\{b_k^{1}\}_{k=1}^{N_r}$ of the rods to investigate how the swimming pattern of the rods will change with their initial geometric configuration.

The number of source (or evaluation) points in this example is 33795, which is on the small end of the range of problems considered in the previous section. This choice is made based on practical considerations. The current simulation will be over a period of time which entails computing (30) or (32) a large number of times. One of our goals is to investigate how the error of the FMM propagates when applied
Table 6: A summary of the parameters used in the Kirchhoff rod example.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{D}$</td>
<td>the computational domain</td>
<td>$[-8, 8] \times [-8, 8] \times [0, 16]$</td>
</tr>
<tr>
<td>$L_{\text{max}}$</td>
<td>maximum level of refinement of $\mathcal{D}$ in the FMM</td>
<td>3</td>
</tr>
<tr>
<td>$N_q$</td>
<td>number of quadrature points</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>(a $5 \times 5 \times 5$ uniform Cartesian grid is used)</td>
<td></td>
</tr>
<tr>
<td>$N_r$</td>
<td>number of rods</td>
<td>225</td>
</tr>
<tr>
<td>$M$</td>
<td>number of segments on each rod</td>
<td>150</td>
</tr>
<tr>
<td>$N_s$</td>
<td>number of grid points on the rods</td>
<td>$225 \times 151 = 33975$</td>
</tr>
<tr>
<td>$L$</td>
<td>length of each rod</td>
<td>9 $\mu$m</td>
</tr>
<tr>
<td>$\Delta s$</td>
<td>length of each segment on the rods</td>
<td>$L/M = 0.06$ $\mu$m</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>size of a time step</td>
<td>$10^{-6}$ s</td>
</tr>
<tr>
<td>$N_t$</td>
<td>number of time steps</td>
<td>20000</td>
</tr>
<tr>
<td>$\varepsilon$</td>
<td>the regularization parameter in the MRS</td>
<td>$5\Delta s$, $6\Delta s$, $7\Delta s$</td>
</tr>
</tbody>
</table>

to a time-dependent problem. Therefore, we need to solve this problem using both the FMM and direct summation and compare their results at each time step. In the previous example, we only apply direct summation to compute the linear velocities for a sample of particles to save time. Here, since we need to update the locations and triads of all the source points before moving on to the next time step, direct summation has to be applied to the full problem. The number of rods is chosen so that the simulation with direct summation for (30) can be performed in a reasonable amount of time.
Recall that what drives the rods to move is their desire to achieve an intrinsic shape. In our simulations, the desired shape is chosen to be a traveling spiral wave determined by a time-dependent strain-twist vector $\Omega$:

$$\Omega(s, t) = \{\Omega_1, \Omega_2, \Omega_3\} = \{\kappa \cos(\tau(s - \sigma t)), \kappa \sin(\tau(s - \sigma t)), 0\}, \quad (39)$$

where $s$ is the arclength between any point on a rod and the base point of the same rod, $t$ is the time, $\kappa, \tau, \sigma$ are the curvature, torsion and frequency of the traveling spiral wave, respectively. This choice for the desired shape of the rods is motivated by the observation that sperm flagella propagate planar or helical waves in experiments [50, 51]. A variety of planar and helical waves including the spiral wave given by (39) have been considered in [52] to study the motion of elastic rods in viscous fluids. The negative sign in (39) indicates that the spiral wave will propagate downward causing the rod to move upward in the fluid when no other rods are present. The curvature $\kappa$ and torsion $\tau$ can be computed as

$$\kappa = \frac{b}{(\frac{\lambda}{2\pi})^2 + b^2} \quad \text{and} \quad \tau = \frac{(\frac{\lambda}{2\pi})}{(\frac{\lambda}{2\pi})^2 + b^2},$$

where $b$ is the amplitude and $\lambda$ is the wavelength of the spiral wave. We choose $\sigma = 1257 \text{ s}^{-1}$, $\lambda = 2.25 \mu\text{m}$ and $b = 0.09 \mu\text{m}$, which implies that the period of the
wave is \(2\pi/\sigma = 0.005\) s (or equivalently, \(5000\Delta t\)) and there are \(L/\lambda = 9/2.25 = 4\) wavelengths along the length of each rod.

Since we are using the unconstrained version of the Kirchhoff rod model, inextensibility of the rods is not guaranteed but can be satisfied approximately if a small enough time step is used. Note that this is a result of the way we model the rods regardless of whether direct summation or FMM is used for computing (30). We choose \(\Delta t = 10^{-6}\) s to ensure that the length of all the 225 rods will not grow or shrink by more than 1% of their initial length (9 \(\mu\)m) at any time step throughout the simulations.

In the first set of experiments, we place 225 straight rods on the plane \(z = 0.2\), i.e., \(b_k^0(3) = 0.2\) for \(k = 1, 2, \cdots, N_r\) (see Figure 7(a)). The \(x\) and \(y\) coordinates of \(\{b_k^0\}_{k=1}^{N_r}\) are shown in Figure 6. We simulate the motion of these rods from 0 s to 0.02 s, i.e., for four beats of the spiral wave specified by (39). Figure 7(b) is a snapshot of the same 225 rods at 0.02 s when \(\varepsilon = 5\Delta s\) is used as the regularization parameter in (3). Since the size of each time step is \(\Delta t = 10^{-6}\) s, 20000 time steps are needed which then lead to 20000 evaluations of (30).

To investigate the accuracy of the FMM and how it progresses with time, we introduce the following notation for the relative error in a quantity \(v\) at the \(n\)th time step:

\[
E^n_v = \sqrt{\frac{\sum_{i=1}^{N_s} \| (v_{fmm}^i)^n - v_i^n \|^2}{\sum_{i=1}^{N_s} \| v_i^n \|^2}}.
\] (40)

In Figure 8, we display the sequences of \(E^n_u\), \(E^n_w\), \(E^n_x\), \(E^n_{D1}\), \(E^n_{D2}\) and \(E^n_{D3}\) corresponding to three different values of the regularization parameter \(\varepsilon\): 5\(\Delta s\), 6\(\Delta s\) and 7\(\Delta s\). As can be seen from this figure, the errors in all the quantities grow fairly slowly for the duration of the simulation, especially when \(\varepsilon = 5\Delta s\) or 7\(\Delta s\); and all the errors are between \(O(10^{-6})\) and \(O(10^{-4})\) at the end of the simulation. In addition, each
time step of our simulation takes approximately 175 s to run on a single core when FMM is used for (30), which is about 50% of the run time when direct summation is used. As shown in the previous example, the advantage of the FMM will be more pronounced when more structures are included. This set of experiments demonstrate the potential of the kernel-independent FMM in simulating the long-term behavior of a large collection of microorganisms modeled by the MRS.

In the following experiments, we examine how the swimming pattern and efficiency of the rods vary with their initial placement. The regularization parameter $\varepsilon$ is fixed to be $5\Delta s$. We consider the following four initial placements of the 225 straight rods: on the plane $z = 0.2$ as in the previous experiments, and on the following paraboloid

$$p(x, y) = -\frac{h}{2 \cdot 7.5^2} (x^2 + y^2) + h$$

(41)

where $h = 0.5, 1$ or $2$. The family of paraboloids (41) satisfy the following properties:
\[ p(x, y) = 0 \] at the points \((-7.5, -7.5), (-7.5, 7.5), (7.5, -7.5) \text{ and } (7.5, 7.5) \) and \( p(0, 0) = h \). The projection of the rods are again given by Figure 6. In Figure 9(a), we plot 225 straight rods placed on the paraboloid (41) with \( h = 2 \), and Figure 9(b) is a snapshot of the same set of rods at \( t = 0.02 \) s.

When there is only one rod propagating a spiral wave (39), it will keep moving upward and mostly maintain its upright position. However, when there are a pack of them interacting with each other through the fluid while trying to achieve the same desired curvature, very different behaviors can be observed depending on their initial placement and locations relative to each other. Since each rod propagates a spiral wave, the more upright it is, the more circular its projection onto the \( xy \)-plane will be. Therefore, we use this projection as an indicator of how upright a rod is swimming. Each subplot of Figure 10 illustrates the projection of all 225 rods onto the \( xy \)-plane at \( t = 0.02 \) s under one of the four initial placements. As can be seen from this figure, in all four cases, the closer a rod is placed towards the center initially, the more likely that it will stay upright during the simulation. In particular, the rod at the center stays upright regardless of the initial placement. Moreover, the more separated the rods are in their initial elevation, the more tilted they will become while swimming in the fluid.

To visualize the vertical movement of the rods as time progresses, we choose the 15 rods in the center row to be representatives and plot in Figure 11 the \( z \) coordinates of their mid points from \( t = 0 \)s to \( t = 0.02 \) s. Each curve in this figure connects the mid points of these 15 rods at the same time step. As shown in Figure 11(a), when the rods are placed on the plane \( z = 0.2 \) initially, they will keep moving upward throughout the simulation with the ones on the two ends moving slightly faster than the ones in the middle. On the other hand, when the rods are initially placed on a paraboloid, they will try to become aligned by swimming upward, staying where
they are or even swimming backward depending on their positions relative to their neighbors, as depicted in Figure 11(b)(c)(d). In the case where \( h = 0.5 \), for example, the mid points of the rods are almost at the same height already when \( t = 0.02 \) s.

We also monitor the hydrodynamic efficiency of vertical swimming. The efficiency of the \( k \)th rod at moving vertically at the \( n \)th time step is defined as

\[
\eta_k^n = \left( \frac{\frac{1}{M+1} \sum_{i=(k-1)\cdot(M+1)+1}^{k\cdot(M+1)} \mathbf{f}_i^n \cdot \mathbf{u}_i^n(3)}{\sum_{i=(k-1)\cdot(M+1)+1}^{k\cdot(M+1)} \mathbf{f}_i^n \cdot \mathbf{u}_i^n} \right)^2, \tag{42}
\]

where \( \mathbf{f}_i^n \cdot \mathbf{u}_i^n \) is the power generated by the \( i \)th point at the \( n \)th time step, \( \mathbf{u}_i^n(3) \) is the \( z \) component of \( \mathbf{u}_i^n \), and the summation is over all the grid points on the \( k \)th rod [53]. The average efficiency of the \( k \)th rod from \( n = 1 \) to \( N_t \) is defined as

\[
\bar{\eta}_k = \frac{1}{N_t} \sum_{n=1}^{N_t} \eta_k^n. \tag{43}
\]

Figure 12 displays the average efficiencies of all 225 rods under the four initial placements. The gray scale of each small square represents the average efficiency of the rod initially placed at its center: the brighter the square is, the larger the average efficiency is. In all four cases, the maximum average efficiency occurs on the four edges and this maximum increases as the paraboloid becomes taller. When \( h = 0.5 \) or \( h = 1 \), as can be seen from Figure 12(b) and (c), the rods in the middle are the least efficient. This is because they move very slowly or even just stay where they are vertically as shown in Figure 11(b) and (c). When \( h = 2 \), however, these rods become more efficient because they move downward quickly to stay aligned with the other rods, which has been illustrated in Figure 11(d). The ring of minimum efficiency in Figure 12(d) is also expected from the stagnation of the rods at those locations observed in Figure 11(d).

40
Figure 8: Relative errors of the kernel-independent FMM applied to the Kirchhoff rod example ($\varepsilon = 5\Delta s, 6\Delta s$ or $7\Delta s$).
Figure 9: Left: 225 straight rods placed on the paraboloid. Right: the same set of rods at 0.02 s ($\varepsilon = 5\Delta s$).
Figure 10: Projection of the rods onto the $xy$-plane $t = 0.02$ s.
Figure 11: The $z$ coordinates of the mid point of the 15 rods on the center row at $t = 0\, \text{s}, 0.005\, \text{s}, 0.010\, \text{s}, 0.015\, \text{s}$ and $0.02\, \text{s}$. 
Figure 12: Mean efficiencies of the rods from $t = 0$ s to 0.02 s (the first four beats).
6. Conclusion

The method of regularized Stokeslets (MRS) gives rise to an $N$-body problem whose computational cost is $O(N^2)$. This becomes a bottleneck of MRS in the simulation of active matter where $N$ is typically very large. We demonstrate the effectiveness of the kernel-independent fast multipole method (FMM) at accelerating this computation by considering both a simple model where the microorganisms are represented by pairs of particles and a much more sophisticated model where they are modeled as elastic rods that will bend and twist. In our numerical experiments, significant savings in run time and satisfactory numerical errors are achieved for $N$ varying from tens of thousands to tens of millions. Swimming speeds and directions, as well as hydrodynamic efficiencies of the microorganisms have been investigated, which indicate a strong dependency on their initial placement. Additionally, the slow progression of the errors in the time-dependent simulations shows the robustness of FMM in studying the self-organization of micro swimmers which usually takes place over a period of time, as has been observed in experiments.

There are many future directions of this work that we would like to explore in both applications and numerical methods. For applications, we plan to increase the length of the simulations presented here until large-scale, recognizable patterns in the microorganisms emerge and compare the simulation results with the experimental results; it is also of our great interest to examine how various factors, such as the viscosity of the fluid and the waveform, frequency, elasticity and density of the micro swimmers, affect the formation of patterns. As for numerical methods, we want to understand how the accuracy of FMM depends on the choice of blob function and regularization parameter in MRS, to explore solution methods for the integral equations arising from the numerical factorization of the MRS kernels as well as ways
to factor them analytically, and to compare the performance of FMM and other fast summation methods such as tree codes.
References


