Journal of Computational Physics 494 (2023) 112506

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A multigrid method for kernel functions acting on interacting structures with applications to biofluids

Weifan Liu [a], Minghao W. Rostami [b,*]

[a] College of Science, Beijing Forestry University, Beijing 100083, China
[b] Department of Mathematics and Statistics, Binghamton University, PO Box 6000, Binghamton, NY 13902-6000, USA

ARTICLE INFO

Article history:
Received 18 January 2023
Received in revised form 2 August 2023
Accepted 11 September 2023
Available online 18 September 2023

Keywords:
Multigrid
Kernel function
Block Gauss-Seidel
Method of regularized Stokeslets
Fluid-structure interaction

ABSTRACT

Simulating the dynamics of discretized interacting structures whose relationship is dictated by a kernel function gives rise to a large dense matrix. We propose a multigrid solver for such a matrix that exploits not only its data-sparsity resulting from the decay of the kernel function but also the regularity of the geometry of the structures and the quantities of interest distributed on them. Like the well-known multigrid method for large sparse matrices arising from boundary-value problems, our method requires a smoother for removing high-frequency terms in solution errors, a strategy for coarsening a grid, and a pair of transfer operators for exchanging information between two grids. We develop new techniques for these processes that are tailored to a kernel function acting on discretized interacting structures. They are matrix-free in the sense that there is no need to construct the large dense matrix. Numerical experiments on a variety of bio-inspired microswimmers immersed in a Stokes flow demonstrate the effectiveness and efficiency of the proposed multigrid solver. In the case of free swimmers that must maintain force and torque balance, additional sparse rows and columns need to be appended to the dense matrix above. We develop a matrix-free fast solver for this bordered matrix as well, in which the multigrid method is a key component.

1. Introduction

Natural and artificial microswimmers are of considerable interest due to their multitude of biological implications and potential biomedical applications. For example, flagellated bacterial carpets can be used as actuators to enhance fluid mixing and pumping [1–3]. In humans, motile cilia exist on the respiratory epithelium and are critical to the mucus clearance in the lung. Cilia in the fallopian tubes of female mammals assist the self propulsion of spermatozoa in the preovulatory phase as well as the fertilization process [4]. It has been hypothesized that controlled microswimmers can perform targeted drug delivery and microsurgery [5,6]. Numerical simulation of the dynamics of these microstructures serves as a powerful and indispensable tool for understanding and eventually harnessing their motility.

In such a computation, the dynamic microswimmers are typically discretized by a Lagrangian grid that evolves with time. The flow field around them can be described accurately by the incompressible Stokes equations. Solvers such as the Method of Regularized Stokeslet (MRS) [7], the Boundary Integral Equation (BIE) formulation [8], and the Boundary Element Method (BEM) [9] give rise to a linear system whose coefficient matrix is generated by an underlying kernel function and characterizes the pair-wise hydrodynamic interactions between the grid points. Knowing the fluid velocities at these points,

* Corresponding author.
E-mail addresses: weifanliu@bjfu.edu.cn (W. Liu), mrostami@binghamton.edu (M.W. Rostami).

https://doi.org/10.1016/j.jcp.2023.112506
0021-9991/

whether prescribed or experimentally observed, we can uncover the hydrodynamic forces that they must have exerted by solving this linear system, which in turn allow us to evaluate the entire flow field. Measuring these forces in a lab is rather challenging. In nature, microswimmers rarely exist in isolation; in fact, they often collaborate with one another to achieve key functionalities. For example, *Volvox*, a freshwater green alga known as a model organism for studying multicellularity, forms hollow, spherical colonies that roll and swim toward light to perform photosynthesis by coordinating the "rowing" of numerous flagella on their surfaces [10]. Simulating these collective behaviors entails the use of a large number of grid points and hence gives rise to a large-scale linear system. In addition to the scale of the problem, another major challenge is that unlike a finite-element or finite-difference matrix corresponding to an Eulerian grid of the fluid domain, the coefficient matrix of this system is dense, that is, it contains very few zero entries.

The "stroke of luck" that allows for the development of fast solution methods for the aforementioned linear system is the decay of the kernel function. More specifically, the greater the distance between a pair of points immersed in a Stokes flow is, the weaker the hydrodynamic interaction between them is, and the smaller the magnitude of the corresponding matrix entries is. Consequently, by dividing the points into a "tree" of clusters based on their proximity, we obtain a hierarchical partition of the coefficient matrix into smaller blocks, among which the ones corresponding to "well-separated" clusters have accurate low-rank approximations and are hence data-sparse. Large dense matrices arising from a wide variety of applications possess similar decay properties, based on which many fast methods have been developed. Direct solvers [11–16] generally seek a hierarchical, data-sparse factorization of the coefficient matrix by exploiting low-rank approximations of its sub-matrices. Alternatively, various iterative methods such as generalized minimal residual method (GMRES), conjugate gradient (CG) and biconjugate gradient stabilized method (BiCGSTAB) have also been applied, typically in conjunction with a preconditioner, to solve linear systems with large dense coefficient matrices. For matrices generated by a kernel function, many preconditioners that exploit their hierarchical, data-spares structures have been developed [17–20]. Other preconditioners such as the structured incomplete factorization preconditioners [21,22], the multilevel Schwarz preconditioners [23], and the sparse approximate inverse preconditioners [24,25] have also been devised for various applications.

In these methods, discretized interacting structures would be viewed simply as a point cloud; the membership of the grid points in these structures prior to discretization would not be utilized, nor would the smoothness in quantities such as velocity and force along each structure. In this paper, we develop a multigrid method for kernel functions acting on discretized interacting structures that exploits the interconnections between the grid points representing the structures as well as the decay of the kernel function. Our method is matrix-free in the sense that there is no need to explicitly construct the large and dense coefficient matrix. Like the well-known multigrid algorithm [26–28], this method iteratively updates the solution to the original linear system based on solutions to smaller linear systems corresponding to coarser discretization of the structures. Its main "ingredients" also include grid coarsening, a smoother for removing the high-frequency components of the error in an approximate solution, and a pair of transfer operators for exchanging information between a coarse grid and a fine grid. Depending on whether a parametrization of the structures is available, we develop two versions of multigrid that are analogous to the geometric multigrid for structured grids and algebraic multigrid [29–34] for unstructured grids.

However, despite the similarity in algorithmic structure between our method and the existing multigrid, there are a number of key differences. Multigrid is traditionally and still predominantly applied to linear systems arising from spatial discretization of Partial Differential Equations (PDEs), where the grids are Eulerian, and the coefficient matrices are sparse; for example, see [35–38] where multigrid methods have been developed for linear systems resulting from spatial discretization of the Navier-Stokes equations and the Stokes equations on structured Eulerian grids. In contrast, in a simulation of interacting structures whose relationship is dictated by a kernel function, the grid is Lagrangian and unstructured, and the coefficient matrix is dense. Because of these differences, commonly used methods, such as the point Gauss-Seidel smoother, are either ineffective or inefficient. In particular, we note that while the high-frequency components in the error correspond to large eigenvalues of a finite-difference or finite-element matrix arising from discretization of an elliptic PDE, this is not necessarily the case for the dense matrix generated by a kernel function. For this reason, specialized smoothers have been developed in [39–41] for large dense matrices arising from the BEM. However, these techniques are equation- and boundary condition-dependent and do not generalize easily to a different scenario, such as the kernel function of interest in this study. The methods in [40,41] also entail eigenvalue computations, which can be challenging for large dense matrices. Taking advantage of the decay of the kernel function, we develop effective and efficient techniques for grid coarsening, smoothing solution errors, and transferring quantities between two grids. We emphasize that these techniques do not require constructing the large dense coefficient matrix associated with the original fine grid.

The rest of the paper is organized as follows. In Section 2, we briefly review the MRS for solving fluid-structure interactions and introduce the kernel function as well as the linear system of interest. Note that our method is applicable to other kernel functions as long as they have similar decay properties. In Section 3, we give an outline of the original multigrid algorithm, which is shared by our method. In Section 4, we propose efficient methods to perform grid coarsening, error smoothing, and inter-grid transferring for kernel functions acting on discretized interacting structures. We consider both the case where the structures are parameterized and the case where only discrete points on them are given, leading to two veins of methods that resemble the geometric multigrid and the algebraic multigrid. These methods are then tested in Section 6 on a variety of microswimmers. For free swimmers that undergo rigid translation and rotation, the linear system considered above must be augmented to incorporate additional constraints. In Section 7, we describe how the proposed multigrid method can be applied to solve these more challenging problems and present additional numerical results. Finally, a summary of the paper and concluding remarks can be found in Section 8.

2. Method of regularized Stokeslets

When viscous forces dominate and inertia is negligible, a fluid flow can be modeled by the incompressible Stokes equations

$$\begin{aligned} \mu\nabla^2\mathbf{u} - \Delta p &= -\mathbf{f}, \\ \nabla\cdot\mathbf{u} &= 0, \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the fluid velocity at $\mathbf{x}$, p is the fluid pressure at $\mathbf{x}$, μ is the fluid viscosity, and $\mathbf{f}$ is the force acting on the fluid. Let $\mathbf{f} = \mathbf{f}_0\delta(\mathbf{x}-\mathbf{x}_0)$ where δ denotes the Dirac delta function and $\mathbf{f}_0$ denotes the point force located at $\mathbf{x}_0$. Then the fluid velocity at $\mathbf{x}$ induced by $\mathbf{f}_0$ can be calculated analytically. It is referred to as the Stokeslet and is singular at $\mathbf{x} = \mathbf{x}_0$. To allow for calculating the fluid velocity at $\mathbf{x}_0$, the Method of Regularized Stokeslets (MRS) was developed in [7], where the Dirac delta function is replaced by a radially symmetric smooth function ϕ_ϵ called the blob function. It has the effect of spreading the point force over a small ball centered at $\mathbf{x}_0$ with radius ϵ and approaches the Dirac delta function as ϵ goes to 0. The resulting analytic solution is no longer singular and is referred to as the regularized Stokeslet.

The MRS was originally developed for unbounded domains [7,42] and has been extended to a domain bounded by a solid planar wall where the flow vanishes [43] using the method of images [44]. We consider both formulations in this paper. Due to the linearity of Eq. (1), the fluid velocity $\mathbf{u}(\mathbf{x})$ at any point $\mathbf{x}$ in either domain induced by N regularized forces $\mathbf{f}_k$ located at $\mathbf{x}_k$ can be written as

$$\mathbf{u}(\mathbf{x}) = \sum_{k=1}^{N} K(\mathbf{x},\mathbf{x}_k)\mathbf{f}_k, \tag{2}$$

where, in three dimensions, $K(\mathbf{x},\mathbf{y})$ is a 3×3 kernel function that determines the hydrodynamic interaction between $\mathbf{x}$, $\mathbf{y}$ and decays as the Euclidean distance $r = \|\mathbf{x}-\mathbf{y}\|_2$ between the two points increases. The precise form of K depends on the choice of the blob function ϕ_ϵ and whether the domain is bounded. In this work, we always use

$$\phi_\epsilon = \frac{15\epsilon^4}{8\pi(r^2+\epsilon^2)^{7/2}}. \tag{3}$$

In addition, the N points $\mathbf{x}_k$ are the Lagrangian grid points used to discretize interacting dynamic structures immersed in the fluid.

If we concatenate all the $\mathbf{u}(\mathbf{x}_k)$ to form a long vector u_h and concatenate all the $\mathbf{f}_k$ to form a long vector f_h, using Eq. (2), we obtain a linear system

$$u_h = A_h f_h \tag{4}$$

where $u_h,\ f_h \in \mathbb{R}^{3N}$, and $A_h \in \mathbb{R}^{3N\times 3N}$ consists of 3×3 blocks $K(\mathbf{x}_i,\mathbf{x}_j)$ and is dense.

In this work, we are interested in the case where N is large, u_h is known, and f_h is wanted. That is, we need to solve a linear system whose coefficient matrix is large and dense. A large N can result from fine discretization and/or a large number of structures. Once f_h is found, we can again use Eq. (2) to evaluate the velocity at any point $\mathbf{x}$ in the fluid domain.

Remark 1. We use the subscript h to indicate that a quantity is associated with the original grid of the structures. In Section 4, we will discuss how to coarsen this grid and transfer quantities from one grid to another. The subscript H will be used for quantities on a coarser grid.

3. Overview of the multigrid method

The multigrid method has been used to solve discretized PDEs arising from numerous applications. Roughly speaking, it is an iterative method that, until convergence, applies a smoother to remove high-frequency oscillations in the error of an approximate solution and then solves the smoothed problem on grids of coarser resolution. Given an initial guess $f^{\langle 0\rangle}$, the kth iteration of a two-grid V-cycle method for solving the linear system $A_h f_h = u_h$ arising from a discretized PDE is outlined below, where $f^{\langle k\rangle}$ denotes the kth iterate of f_h.

Step 1. Apply the smoother to compute an approximate solution, $\tilde{f}^{\langle k-1\rangle}$, to $A_h f_h = u_h$.
Step 2. Compute the residual $r_h = u_h - A_h\tilde{f}^{\langle k-1\rangle}$ and its restriction, r_H, to the coarse grid.
Step 3. Solve $A_H e_H = r_H$ where A_H is the restriction of A_h to the coarse grid.
Step 4. Compute the prolongation, e_h, of e_H to the fine grid and correct $\tilde{f}^{\langle k-1\rangle}$: $f^{\langle k\rangle} = \tilde{f}^{\langle k-1\rangle} + e_h$.

The smoother typically consists of a small fixed number of iterations of a simple iterative solver, such as the Gauss-Seidel or Jacobi method. Information exchange between the fine grid and coarse grid is achieved by a restriction operator R_h^H and a prolongation operator P_H^h, that is, $r_H = R_h^H r_h$ in Step 2, $e_h = P_H^h e_H$ in Step 4, and

$$A_H = R_h^H A_h P_H^h \tag{5}$$

in Step 3. We can recursively coarsen the grid and apply the two-grid method until the linear system restricted to the coarsest grid is sufficiently small. The resulting algorithm is a multigrid solver.

The multigrid method that we propose for the large dense matrix generated by a kernel function acting on discretized interacting structures formally consists of the same steps as above. However, such a matrix poses new challenges: the grid is Lagrangian and unstructured, constructing the matrix can be prohibitively expensive in terms of both runtime and memory usage, and multiplying it to a vector is rather costly as well. Consequently, novel techniques need to be developed for error smoothing, grid coarsening, and inter-grid communication that take advantage of the decay of the kernel function and the resulting data-sparsity of the matrix. In Sections 4 and 5, we describe them in greater detail. Note that the proposed multigrid method does not require constructing the large dense A_h, rendering it matrix-free.

4. The coarsening strategies

In the following three subsections, we describe how to coarsen a grid (Section 4.1), construct the transfer operators between the fine and coarse grids (Section 4.2), and project the large, dense coefficient matrix onto the coarse grid (Section 4.3). Our methods exploit the smoothness in the geometry of biological structures as well as quantities such as velocity and force distributed along them, and the decay of the kernel function as the distance between the source and target points increases.

A Lagrangian grid used to discretize biological structures is unstructured in the xyz space. Nonetheless, when they are modeled by a parametric curve or surface and the Lagrangian grid results from a structured grid in the parameter space, our method is similar to the geometric multigrid. When a parameterization is unavailable and the structures are represented by discrete points, our method is similar to the algebraic multigrid. We consider both cases in each subsection below.

4.1. Coarsening the grid

4.1.1. Case I: a parameterization of the structures is known

Biological structures are often modeled as parametric curves or surfaces in three dimension. For example, a bacterial flagellum can be represented by a helix. Although a curve or surface itself does not have any thickness, placing regularized Stokeslets along it has the effect of adding thickness to it. When the parameterization is known, we can obtain a grid for the structures by discretizing the parameter space, usually uniformly for simplicity.

For example, consider a structure modeled as a parametric curve

$$\boldsymbol{\mathcal{X}}(s):\ \mathbb{R}^1 \longrightarrow \mathbb{R}^3 \text{ with } s \in [0, L]$$

where s is the arclength. Let $0,\ h,\ 2h,\ \cdots,\ (N-1)\cdot h = L$ where $h = L/(N-1)$ constitute a uniform grid on $[0, L]$. The corresponding Lagrangian grid points on the structure are

$$\boldsymbol{\mathcal{X}}(0),\ \boldsymbol{\mathcal{X}}(h),\ \boldsymbol{\mathcal{X}}(2h),\ \cdots,\ \boldsymbol{\mathcal{X}}\big((N-1)\cdot h\big).$$

We can simply coarsen this grid by coarsening the grid of the parameter space. Assume that there are $N = 2^k + 1$ points in the fine grid resulting from the uniform grid with spacing $h = L/2^k$ of the parameter space. A coarser grid consisting of $\mathcal{N} = 2^{k-p} + 1$ points, where p is an integer satisfying $1 \le p < k$, can be obtained by increasing the grid spacing in the parameter space to $H = 2^p \cdot h$. In this case, the two grids are "nested" since the coarse-grid points also belong to the fine grid. In Fig. 1, we show the fine grid (Fig. 1b) and coarse grid (Fig. 1c) of a structure represented by a helical curve. The two corresponding grids of the parameter space are also shown in Fig. 1a, where the black dots are the fine-grid points and the red crosses are the coarse-grid points.

When the structure is modeled as a parametric surface

$$\boldsymbol{\mathcal{X}}(u, v):\ \mathbb{R}^2 \longrightarrow \mathbb{R}^3 \text{ with } u \in [0, L_u] \text{ and } v \in [0, L_v],$$

we can again coarsen a grid of the structure by coarsening the corresponding grid of the parameter space. In Fig. 1, we show the fine grid (Fig. 1e) and coarse grid (Fig. 1f) of a structure represented by a torus. The two corresponding grids of the parameter space are also shown in Fig. 1d, where the black dots are the fine-grid points and the red crosses are the coarse-grid points.

4.1.2. Case II: no parameterization is available

In this case, the structures are represented by a collection of scattered points and not parameterized prior to discretization. Our strategy is based on the Parallel Modified Independent Set (PMIS) algorithm [45,46], a commonly used approach for coarsening an unstructured Eulerian grid in the algebraic multigrid method for discretized PDEs. It iteratively sorts a given set of points (the fine-grid points) into two subsets: the so-called "C-points" and "F-points." The former refers to the coarse-grid points that we seek, and the latter refers to the rest of the fine-grid points. Both sets are initialized to be the empty set. In each iteration of the PMIS algorithm, roughly speaking, the most "influential" unsorted fine-grid points are

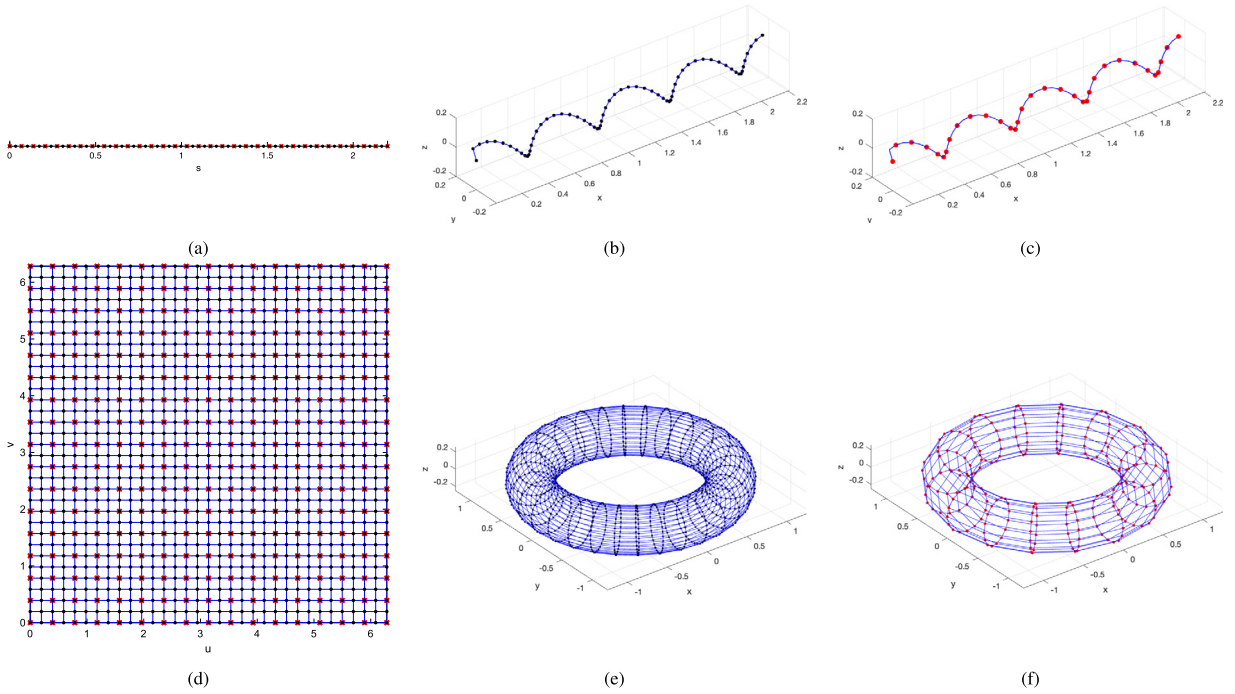

Fig. 1. Grid coarsening for two parameterized structures: a helix (a-c) and a torus (d-f). (a,d): Grid coarsening in the parameter spaces. Black dots: fine-grid points. Red crosses: coarse-grid points. (b,e): The fine grids of the structures. (c,f): The coarse grids of the structures. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

first added to the set of C-points; and the unsorted fine-grid points "influenced" by the new C-points are then added to the set of F-points. The algorithm terminates when all the fine-grid points that we start with have been sorted into one of the two subsets. By construction, the coarse grid is always nested into the fine grid.

In the original PMIS algorithm, the sizes of the entries of the coefficient matrix A_h determine how influential the fine-grid points are. For a large dense A_h, we have to modify this algorithm so that it can be performed without explicitly constructing A_h. Let $\{\mathbf{x}_i\}_{i=1}^N$ be the set of fine-grid points. Since the kernel function and thus the corresponding matrix entries decay as the source and target points move further apart, we utilize the pairwise distance, $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ for $i,\ j = 1,\ 2,\ \cdots,\ N$, to quantify the influence that a point has on another point. The specifics in the case of a single structure are as follows. We define an $N \times N$ auxiliary strength matrix S whose (i, j) entry is

$$S_{ij} = \begin{cases} 1, & \text{if } i \neq j \text{ and } d_{ij} < \frac{1}{\alpha} \cdot \min\limits_k \{d_{ik} :\ i \neq k\} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $0 < \alpha < 1$ is a constant. (In the original PMIS algorithm, the auxiliary strength matrix is calculated based on the absolute values of the entries of A_h instead.) The ith point $\mathbf{x}_i$ is considered to be influenced by the jth point $\mathbf{x}_j$ if and only if $S_{ij} = 1$. Thus, the sum of the ith column of S gives the total number of points influenced by $\mathbf{x}_i$. The weight of $\mathbf{x}_i$ is defined to be this sum plus a random number in $[0, 1]$ and measures how influential $\mathbf{x}_i$ is. The coarseness of the coarse grid found by the PMIS algorithm can be controlled by the value of α: the smaller α is, the larger the threshold on d_{ij} is in Eq. (6), the more points will be influenced by a C-point and added to the set of F-points, and the fewer points there will be in the set of C-points. When there are multiple structures, instead of applying the PMIS algorithm to the fine grid as a whole, we apply it to the fine grid restricted to each structure individually.

As an example, consider an ellipsoidal structure represented by 1016 points, as shown in Fig. 2. When $\alpha = 0.45$ is chosen in Eq. (6), it takes the modified PMIS algorithm four iterations to divide them into F-points and C-points, producing a coarse grid consisting of 127 points. The results of all four iterations are displayed in Fig. 2a to Fig. 2d, where the blue dots are the fine-grid points and the red asterisks are the C-points that have been found after each iteration. The C-points shown in Fig. 2d constitute the coarse grid.

4.2. Constructing the transfer operators

Suppose the fine-grid points are $\mathbf{x}_1,\ \mathbf{x}_2,\ \cdots,\ \mathbf{x}_N \in \mathbb{R}^3$ and the coarse-grid points are $\hat{\mathbf{x}}_1,\ \hat{\mathbf{x}}_2,\ \cdots,\ \hat{\mathbf{x}}_{\mathcal{N}} \in \mathbb{R}^3$ where $\mathcal{N} < N$. Let $\hat{f} \in \mathbb{R}^{3\mathcal{N}}$ be a vector that needs to be prolonged, which is formed by concatenating the following quantities

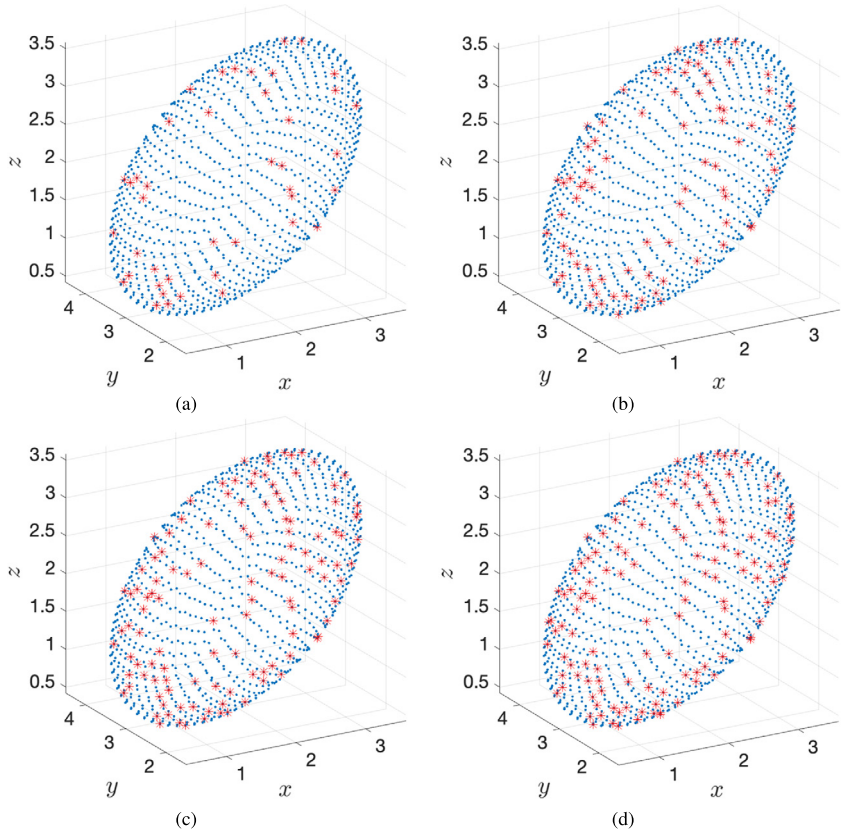

Fig. 2. (Color online) Grid coarsening for an ellipsoidal structure using the modified PMIS algorithm. Blue dots: fine-grid points. Red asterisks: coarse-grid points. (a-d): The results of Iterations 1 through 4.

associated with the coarse-grid points: $\hat{\mathbf{f}}_1,\ \hat{\mathbf{f}}_2,\ \cdots,\ \hat{\mathbf{f}}_{\mathcal{N}} \in \mathbb{R}^3$. We seek a prolongation operator $P \in \mathbb{R}^{3N\times 3\mathcal{N}}$ such that $f = P\hat{f}$ is formed by concatenating $\mathbf{f}_1,\ \mathbf{f}_2,\ \cdots,\ \mathbf{f}_N \in \mathbb{R}^3$, the quantities associated with the fine-grid points. We again consider the case where the structures are parameterized (Section 4.2.1) and the case where they are not (Section 4.2.2). Although the details are quite different, we proceed as follows in both cases. For each fine-grid point $\mathbf{x}_i$, we first identify the coarse-grid points that are adjacent to it. Let $\mathscr{I}_i$ denote the set of indices such that $j \in \mathscr{I}_i$ if and only if the coarse-grid point $\hat{\mathbf{x}}_j$ is adjacent to $\mathbf{x}_i$. We then calculate $\mathbf{f}_i$ as

$$\mathbf{f}_i = \sum_{j\in\mathscr{I}_i} w_{ij}\cdot\hat{\mathbf{f}}_j \tag{7}$$

where w_{ij} is a scalar. The resulting prolongation operator P can be partitioned into 3×3 blocks, and its (i,j) block is

$$P_{ij} = \begin{cases} w_{ij}\cdot I_3, & \text{if } j\in\mathscr{I}_i \\ O_3, & \text{otherwise} \end{cases} \tag{8}$$

where I_3 denotes the 3×3 identity matrix and O_3 denotes the 3×3 zero matrix.

We also need a restriction operator $R \in \mathbb{R}^{3\mathcal{N}\times 3N}$ such that given $f \in \mathbb{R}^{3N}$ defined on the fine grid, we can calculate its counterpart on the coarse grid as $\hat{f} = Rf$. Once P has been constructed, we can simply choose $R = P^T$ to be the restriction operator. In the special case where the two grids are nested, we can also construct R as follows:

$$R_{ij} = \begin{cases} I_3, & \text{if } \hat{\mathbf{x}}_i \text{ and } \mathbf{x}_j \text{ coincide} \\ O_3, & \text{otherwise} \end{cases} \tag{9}$$

where R_{ij} is the (i,j) block in R. Applying this operator to restrict f is equivalent to setting $\hat{\mathbf{f}}_i = \mathbf{f}_j$, which does not entail any additional computational cost.

Therefore, for the rest of this subsection, we focus on the construction of the prolongation operator P, specifically, the choice of $\mathscr{I}_i$ and w_{ij} in Eqs. (7) and (8).

4.2.1. Case I: a parameterization of the structures is known

Assume for now that there is only one structure represented by a parametric curve $\boldsymbol{\mathcal{X}}(s)$ in 3D where $s \in [0, L]$. Let the fine-grid points be

$$\mathbf{x}_1 = \boldsymbol{\mathcal{X}}(0),\ \mathbf{x}_2 = \boldsymbol{\mathcal{X}}(h),\ \cdots,\ \mathbf{x}_N = \boldsymbol{\mathcal{X}}\big((N-1)\cdot h\big) \tag{10}$$

where $(N-1)\cdot h = L$ and the coarse-grid points be

$$\hat{\mathbf{x}}_1 = \boldsymbol{\mathcal{X}}(0),\ \hat{\mathbf{x}}_2 = \boldsymbol{\mathcal{X}}(H),\ \cdots,\ \hat{\mathbf{x}}_{\mathcal{N}} = \boldsymbol{\mathcal{X}}\big((\mathcal{N}-1)\cdot H\big) \tag{11}$$

where $(\mathcal{N}-1)\cdot H = L$. Since the biological structures are smooth, we assume that quantities such as velocity and force are also smooth on them. For any integer $1 \le i \le N$, let $\mathcal{J}(i)$ be the unique integer that satisfies the following: $1 \le \mathcal{J}(i) \le \mathcal{N}-1$,

$$(\mathcal{J}(i)-1)\cdot H \le (i-1)\cdot h < \mathcal{J}(i)\cdot H \tag{12}$$

if $i < N$, and $\mathcal{J}(i) = \mathcal{N}-1$ if $i = N$. Thus, we consider $\hat{\mathbf{x}}_{\mathcal{J}(i)}$ and $\hat{\mathbf{x}}_{\mathcal{J}(i)+1}$ to be the coarse-grid points that are adjacent to $\mathbf{x}_i$ (see Fig. 1a-c), that is,

$$\mathscr{I}_i = \{\mathcal{J}(i), \mathcal{J}(i)+1\}. \tag{13}$$

Using the linear spline on the interval $\big[(\mathcal{J}(i)-1)\cdot H, \mathcal{J}(i)\cdot H\big]$ around $(i-1)\cdot h$ in the parameter space, we can calculate $\mathbf{f}_i$ as

$$\mathbf{f}_i = \sum_{j\in\mathscr{I}_i} w_{ij}\cdot\hat{\mathbf{f}}_j \tag{14}$$

where

$$\begin{aligned} w_{i,\mathcal{J}(i)} &= 1 + \frac{(\mathcal{J}(i)-1)\cdot H - (i-1)\cdot h}{H}, \\ w_{i,\mathcal{J}(i)+1} &= \frac{(i-1)\cdot h - (\mathcal{J}(i)-1)\cdot H}{H}. \end{aligned} \tag{15}$$

Eqs. (7) and (8) follow immediately.

Remark 2. While the interpolation would be more accurate if a higher-order method such as the cubic spline is used, the weights w_{ij} would depend on not only the parameter s but also $\hat{\mathbf{f}}_1,\ \hat{\mathbf{f}}_2,\ \cdots,\ \hat{\mathbf{f}}_{\mathcal{N}}$. As a result, P would have to be constructed case by case for each $\hat{f}$ that needs to be prolonged.

Next, we consider a single structure represented by a parametric surface $\boldsymbol{\mathcal{X}}(u, v)$ in 3D where $u \in [0, L_u]$ and $v \in [0, L_v]$. Let $N = N_u \cdot N_v$ where N_u and N_v are integers greater than 1. Let $h_u = L_u/(N_u - 1)$ be the fine-grid spacing in the u direction and $h_v = L_v/(N_v - 1)$ be the fine-grid spacing in the v direction. Similarly, let $\mathcal{N} = \mathcal{N}_u \cdot \mathcal{N}_v$ where $\mathcal{N}_u < N_u$ and $\mathcal{N}_v < N_v$. Let $H_u = L_u/(\mathcal{N}_u - 1)$ be the coarse-grid spacing in the u direction and $H_v = L_v/(\mathcal{N}_v - 1)$ be the coarse-grid spacing in the v direction. In addition, let the fine-grid points be numbered such that

$$\mathbf{x}_{m+(n-1)\cdot N_u} = \boldsymbol{\mathcal{X}}\big((m-1)\cdot h_u, (n-1)\cdot h_v\big) \tag{16}$$

for $1 \le m \le N_u$ and $1 \le n \le N_v$; and similarly, let the coarse-grid points be numbered such that

$$\hat{\mathbf{x}}_{p+(q-1)\cdot\mathcal{N}_u} = \boldsymbol{\mathcal{X}}\big((p-1)\cdot H_u, (q-1)\cdot H_v\big) \tag{17}$$

for $1 \le p \le \mathcal{N}_u$ and $1 \le q \le \mathcal{N}_v$. For any integer $1 \le m \le N_u$, let $\mathcal{P}(m)$ be the unique integer that satisfies the following: $1 \le \mathcal{P}(m) \le \mathcal{N}_u - 1$,

$$(\mathcal{P}(m)-1)\cdot H_u \le (m-1)\cdot h_u < \mathcal{P}(m)\cdot H_u \tag{18}$$

if $m < N_u$, and $\mathcal{P}(m) = \mathcal{N}_u - 1$ if $m = N_u$. For any integer $1 \le n \le N_v$, let $\mathcal{Q}(n)$ be the unique integer that satisfies the following: $1 \le \mathcal{Q}(n) \le \mathcal{N}_v - 1$,

$$(\mathcal{Q}(n)-1)\cdot H_v \le (n-1)\cdot h_v < \mathcal{Q}(n)\cdot H_v. \tag{19}$$

if $n < N_v$, and $\mathcal{Q}(n) = \mathcal{N}_v - 1$ if $n = N_v$. Thus, for $1 \le m \le N_u$ and $1 \le n \le N_v$, we consider

$$\hat{\mathbf{x}}_{\mathcal{P}(m)+(\mathcal{Q}(n)-1)\cdot\mathcal{N}_u},\ \hat{\mathbf{x}}_{\mathcal{P}(m)+(\mathcal{Q}(n)-1)\cdot\mathcal{N}_u+1},\ \hat{\mathbf{x}}_{\mathcal{P}(m)+\mathcal{Q}(n)\cdot\mathcal{N}_u},\ \text{and}\ \hat{\mathbf{x}}_{\mathcal{P}(m)+\mathcal{Q}(n)\cdot\mathcal{N}_u+1}$$

to be the coarse-grid points adjacent to $\mathbf{x}_i$ where $i = m + (n-1)\cdot N_u$ (see Fig. 1d-f), that is,

$$\mathscr{I}_i = \{\mathcal{P}(m) + (\mathcal{Q}(n)-1)\cdot \mathcal{N}_u,\ \mathcal{P}(m) + (\mathcal{Q}(n)-1)\cdot \mathcal{N}_u + 1,\ \mathcal{P}(m) + \mathcal{Q}(n)\cdot \mathcal{N}_u,\ \mathcal{P}(m) + \mathcal{Q}(n)\cdot \mathcal{N}_u + 1\}. \tag{20}$$

Using the bilinear interpolation on the neighborhood

$$\big[(\mathcal{P}(m)-1)\cdot H_u, \mathcal{P}(m)\cdot H_u\big] \times \big[(\mathcal{Q}(n)-1)\cdot H_v, \mathcal{Q}(n)\cdot H_v\big]$$

around $\big((m-1)\cdot h_u, (n-1)\cdot h_v\big)$ in the parameter space, we can again calculate $\mathbf{f}_i$ as (14) where

$$\begin{aligned}
w_{i,\mathcal{P}(m)+(\mathcal{Q}(n)-1)\cdot\mathcal{N}_u} &= \left(1 + \frac{(\mathcal{P}(m)-1)\cdot H_u - (m-1)\cdot h_u}{H_u}\right)\cdot\left(1 + \frac{(\mathcal{Q}(n)-1)\cdot H_v - (n-1)\cdot h_v}{H_v}\right),\\
w_{i,\mathcal{P}(m)+(\mathcal{Q}(n)-1)\cdot\mathcal{N}_u+1} &= \frac{(m-1)\cdot h_u - (\mathcal{P}(m)-1)\cdot H_u}{H_u}\cdot\left(1 + \frac{(\mathcal{Q}(n)-1)\cdot H_v - (n-1)\cdot h_v}{H_v}\right),\\
w_{i,\mathcal{P}(m)+\mathcal{Q}(n)\cdot\mathcal{N}_u} &= \left(1 + \frac{(\mathcal{P}(m)-1)\cdot H_u - (m-1)\cdot h_u}{H_u}\right)\cdot\frac{(n-1)\cdot h_v - (\mathcal{Q}(n)-1)\cdot H_v}{H_v},\\
w_{i,\mathcal{P}(m)+\mathcal{Q}(n)\cdot\mathcal{N}_u+1} &= \frac{(m-1)\cdot h_u - (\mathcal{P}(m)-1)\cdot H_u}{H_u}\cdot\frac{(n-1)\cdot h_v - (\mathcal{Q}(n)-1)\cdot H_v}{H_v}.
\end{aligned} \tag{21}$$

Eqs. (7) and (8) follow as in the case of a parametric curve.

So far we have considered the case of a single structure modeled as a parametric curve or surface prior to discretization. We now give a brief description of the prolongation operator P when there are $n_s > 1$ structures. Assume that for $1 \le k \le n_s$, the kth structure represented by $\boldsymbol{\mathcal{X}}_k$ is discretized by N_k points in the fine grid and $\mathcal{N}_k$ points in the coarse grid, where N_k and $\mathcal{N}_k$ satisfy the following: $\mathcal{N}_k < N_k$, $\sum_{k=1}^{n_s} N_k = N$, and $\sum_{k=1}^{n_s} \mathcal{N}_k = \mathcal{N}$. Also assume that the points on the kth structure are numbered by $\sum_{i=1}^{k-1} N_i + 1$ to $\sum_{i=1}^{k} N_i$ in the fine grid and $\sum_{i=1}^{k-1} \mathcal{N}_i + 1$ to $\sum_{i=1}^{k} \mathcal{N}_i$ in the coarse grid. Then the prolongation operator P has the following block-diagonal structure:

$$P = \begin{bmatrix} P_1 & & & \\ & P_2 & & \\ & & \ddots & \\ & & & P_{n_s} \end{bmatrix} \tag{22}$$

where $P_k \in \mathbb{R}^{3N_k \times 3\mathcal{N}_k}$ is the prolongation operator in the case where $\boldsymbol{\mathcal{X}}_k$ is the only structure present.

4.2.2. Case II: no parameterization is available

In Section 4.2.1, it is relatively straightforward to determine whether a coarse-grid point $\hat{\mathbf{x}}_j$ is adjacent to a fine-grid point $\mathbf{x}_i$: $j \in \mathscr{I}_i$ if and only if $\hat{\mathbf{x}}_j$ and $\mathbf{x}_i$ belong to the same structure and correspond to adjacent grid points in the parameter space. It is less clear how $\mathscr{I}_i$ should be determined when the structures are not parameterized. Assume that there is a single structure represented by $\{\mathbf{x}_i\}_{i=1}^{N}$ in the fine grid and $\{\hat{\mathbf{x}}_i\}_{i=1}^{\mathcal{N}}$ in the coarse grid. We consider the m coarse-grid points closest to $\mathbf{x}_i$ to be adjacent to it where m is a small fixed integer between 1 and $\mathcal{N}$, that is,

$$\mathscr{I}_i = \left\{j :\ \hat{\mathbf{x}}_j \text{ is one of the } m \text{ coarse-grid points closest to } \mathbf{x}_i\right\}. \tag{23}$$

In Section 4.2.1, we use linear interpolation to calculate $\mathbf{f}_i$ from $\left\{\hat{\mathbf{f}}_j\right\}_{j\in\mathscr{I}_i}$. Here, we interpolate the coarse-grid quantities using Radial Basis Functions (RBFs) instead. For any vector $\mathbf{v} \in \mathbb{R}^3$, let $\mathbf{v}^\ell$ denote the ℓth component of $\mathbf{v}$ where $\ell = 1, 2,$ or 3. Let $\mathscr{I}_i = \{j_1,\ j_2,\ \cdots,\ j_m\}$ where $1 \le j_1 < j_2 < \cdots < j_m \le \mathcal{N}$. (The dependence of $j_1,\ j_2,\ \cdots,\ j_m$ on i is omitted to simplify the notation.) Following the method of [47], for $1 \le i \le N$ and $\ell = 1,\ 2,\ 3$, we introduce the following function from $\mathbb{R}^3$ to $\mathbb{R}$:

$$g_{i\ell}(\mathbf{x}) = \sum_{q=1}^{m} \lambda_{i\ell q}\cdot e^{-\left(\delta\left\|\mathbf{x}-\hat{\mathbf{x}}_{j_q}\right\|\right)^2} + \sum_{k=0}^{3} a_{i\ell k}\cdot p_k(\mathbf{x}) \tag{24}$$

where δ is a shape parameter,

$$p_k(\mathbf{x}) = \begin{cases} 1, & \text{if } k = 0 \\ \mathbf{x}^k, & \text{otherwise} \end{cases}, \tag{25}$$

and $\{\lambda_{i\ell q}\}_{q=1}^{m}$, $\{a_{i\ell k}\}_{k=0}^{3}$ are scalars to be determined. The ℓth component of $\mathbf{f}_i$ is then calculated as $\mathbf{f}_i^\ell = g_{i\ell}(\mathbf{x}_i)$. To find $\{\lambda_{i\ell q}\}_{q=1}^{m}$ and $\{a_{i\ell k}\}_{k=0}^{3}$, we impose

$$g_{i\ell}\left(\hat{\mathbf{x}}_{j_q}\right) = \hat{\mathbf{f}}^{\ell}_{j_q} \tag{26}$$

for $q = 1,\ 2,\ \cdots,\ m$, that is, $g_{i\ell}(\mathbf{x})$ is exact at all the coarse-grid points adjacent to $\mathbf{x}_i$. Since there are $m+4$ unknowns and only m equations, the following four constraints are also added:

$$\sum_{q=1}^{m} \lambda_{i\ell q} \cdot p_k\left(\hat{\mathbf{x}}_{j_q}\right) = 0 \tag{27}$$

for $k = 0,\ 1,\ 2,\ 3$. Let $\lambda_i^{\ell} = [\lambda_{i\ell 1}\ \lambda_{i\ell 2}\ \cdots\ \lambda_{i\ell m}]^T \in \mathbb{R}^m$, $a_i^{\ell} = [a_{i\ell 0}\ a_{i\ell 1}\ a_{i2\ell}\ a_{i\ell 3}]^T \in \mathbb{R}^4$, and $\hat{f}_i^{\ell} = \left[\hat{\mathbf{f}}^{\ell}_{j_1}\ \hat{\mathbf{f}}^{\ell}_{j_2}\ \cdots\ \hat{\mathbf{f}}^{\ell}_{j_m}\right]^T \in \mathbb{R}^m$. We therefore need to solve the linear system

$$\begin{bmatrix} \Phi_i & \Pi_i \\ \Pi_i^T & O_4 \end{bmatrix} \begin{bmatrix} \lambda_i^{\ell} \\ a_i^{\ell} \end{bmatrix} = \begin{bmatrix} \hat{f}_i^{\ell} \\ 0_4 \end{bmatrix} \tag{28}$$

for every $1 \le i \le N$ and $\ell = 1,\ 2,\ 3$, where the (r,q) entry of $\Phi_i \in \mathbb{R}^{m\times m}$ is $e^{-\left(\delta \left\| \hat{\mathbf{x}}_{j_r} - \hat{\mathbf{x}}_{j_q} \right\|\right)^2}$, the (r,k) entry of $\Pi_i \in \mathbb{R}^{m\times 4}$ is $p_{k-1}\left(\hat{\mathbf{x}}_{j_r}\right)$, O_4 is the 4×4 zero matrix, and 0_4 is the 4×1 zero vector.

It remains to be shown that

$$\mathbf{f}_i^{\ell} = \sum_{j\in\mathscr{I}_i} w_{ij} \cdot \hat{\mathbf{f}}_j^{\ell}, \tag{29}$$

where w_{ij} is a scalar that does not depend on ℓ. By Eq. (28),

$$a_i^{\ell} = \Upsilon_i \hat{f}_i^{\ell} \ \text{ and } \ \lambda_i^{\ell} = \Psi_i \hat{f}_i^{\ell} \tag{30}$$

where $\Upsilon_i = \left(\Pi_i^T \Phi_i^{-1} \Pi_i\right)^{-1} \Pi_i^T \Phi_i^{-1} \in \mathbb{R}^{4\times m}$ and $\Psi_i = \Phi_i^{-1} - \Phi_i^{-1}\Pi_i\Upsilon_i \in \mathbb{R}^{m\times m}$. (Note that Φ_i, Π_i and hence Υ_i, Ψ_i do not depend on ℓ.) Since $\mathbf{f}_i^{\ell} = g_{i\ell}(\mathbf{x}_i)$, by Eqs. (24) and (30), Eq. (29) holds with

$$w_{ij_r} = \sum_{q=1}^{m} \psi_{iqr} \cdot e^{-\left(\delta \left\| \mathbf{x}_i - \hat{\mathbf{x}}_{j_q} \right\|\right)^2} + \sum_{k=1}^{4} \upsilon_{ikr} \cdot p_{k-1}\left(\mathbf{x}_i\right), \tag{31}$$

where ψ_{iqr} is the (q,r) entry of Ψ_i and υ_{ikr} is the (k,r) entry of Υ_i. Eqs. (7) and (8) follow immediately.

In the case of multiple structures, the prolongation operator P again has the block-diagonal structure shown in Eq. (22) if the grid points are numbered appropriately (see the description in the paragraph immediately preceding Eq. (22)).

4.3. Approximating the Galerkin projection

Assume that the fine-grid points $\{\mathbf{x}_i\}_{i=1}^N$, coarse-grid points $\left\{\hat{\mathbf{x}}_i\right\}_{i=1}^{\mathcal{N}}$, and a prolongation operator $P_H^h \in \mathbb{R}^{3N\times 3\mathcal{N}}$, a restriction operator $R_h^H \in \mathbb{R}^{3\mathcal{N}\times 3N}$ between them are known. Recall from Section 2 that the coefficient matrix A_h on the fine grid is $3N\times 3N$ and can be partitioned into 3×3 blocks, where the (i,j) block is $K\left(\mathbf{x}_i,\mathbf{x}_j\right)$. We denote $K\left(\mathbf{x}_i,\mathbf{x}_j\right)$ by K_{ij} and $K\left(\hat{\mathbf{x}}_i,\hat{\mathbf{x}}_j\right)$ by $\widehat{K}_{ij}$. (Note that the former is defined between a pair of fine-grid points whereas the latter is defined between a pair of coarse-grid points.) Recall from Section 4.2 that both P_H^h and R_h^H can be partitioned into 3×3 blocks as well. We continue to denote their (i,j) blocks by P_{ij} and R_{ij}, respectively.

We could calculate a $3\mathcal{N}\times 3\mathcal{N}$ coarse-grid representation A_H of A_h as $R_h^H A_h P_H^h$, referred to as the Galerkin projection of A_h onto the coarse grid. Due to the block structures of A_h, R_h^H, and P_H^h, A_H can also be partitioned in to 3×3 blocks, the (i,j) block of which is

$$G_{ij} = \sum_{l=1}^{N} R_{il} \left(\sum_{n=1}^{N} K_{ln} P_{nj} \right). \tag{32}$$

However, calculating A_H naïvely entails matrix-vector products involving A_h, which in turn entails interactions between each pair of fine-grid points. We develop an efficient approach to approximate G_{ij} that takes advantage of the sparsity of R_h^H and P_H^h as well as the data-sparsity of A_h.

Since the strength of K_{ln} decays as the distance between $\mathbf{x}_l$ and $\mathbf{x}_n$ increases, for accurate and computationally efficient approximation of G_{ij}, we propose

$$\widetilde{G}_{ij} = \sum_{l=1}^{N} R_{il} \left(\sum_{n\in\mathscr{N}_l} K_{ln} P_{nj} + \sum_{n\notin\mathscr{N}_l} \widetilde{K}_{ln} P_{nj} \right), \tag{33}$$

where $\mathscr{N}_l$ denotes the set of indices of all the fine-grid points "close" to $\mathbf{x}_l$, $\widetilde{K}_{ln} \approx K_{ln}$, and the calculation of $\widetilde{K}_{ln}$ involves coarse-grid interactions only. We explain the choice of $\widetilde{K}_{ln}$ and $\mathscr{N}_l$ below.

If $\mathbf{x}_n$ is not close to $\mathbf{x}_l$, that is, if $n \notin \mathscr{N}_l$, the contribution of K_{ln} to G_{ij} is less important. It is thus not necessary to calculate K_{ln} exactly. We can use linear or RBF interpolation to approximate K_{ln} as in Section 4.2:

$$\widetilde{K}_{ln} = \sum_{q \in \mathscr{I}_n} w_{nq} \cdot K\left(\mathbf{x}_l, \hat{\mathbf{x}}_q\right) \tag{34}$$

where $\mathscr{I}_n$ is the set of indices of all the coarse-grid points adjacent to $\mathbf{x}_n$ (see Eqs. (13), (20), and (23)), and w_{nq} is a scalar (see Eqs. (15), (21), and (31)). Using the $\widetilde{K}_{ln}$ defined in Eq. (34), we can avoid the interactions among fine-grid points; however, we still need to calculate the interactions between fine-grid points and coarse-grid points. To eliminate the involvement of the fine-grid points in $\widetilde{K}_{ln}$ entirely, we use linear or RBF interpolation again to approximate $K\left(\mathbf{x}_l, \hat{\mathbf{x}}_q\right)$ in Eq. (34), that is,

$$\widetilde{K}_{ln} = \sum_{q \in \mathscr{I}_n} w_{nq} \cdot \left(\sum_{r \in \mathscr{I}_l} w_{lr} \cdot K\left(\hat{\mathbf{x}}_r, \hat{\mathbf{x}}_q\right) \right) = \sum_{q \in \mathscr{I}_n} w_{nq} \cdot \left(\sum_{r \in \mathscr{I}_l} w_{lr} \cdot \widehat{K}_{rq} \right). \tag{35}$$

Combining Eqs. (33) and (35), we have

$$\widetilde{G}_{ij} = \sum_{l=1}^{N} R_{il} \left\{ \sum_{n \in \mathscr{N}_l} K_{ln} P_{nj} + \sum_{n \notin \mathscr{N}_l} \left[\sum_{q \in \mathscr{I}_n} w_{nq} \cdot \left(\sum_{r \in \mathscr{I}_l} w_{lr} \cdot \widehat{K}_{rq} \right) \right] P_{nj} \right\}. \tag{36}$$

We note that thanks to the sparsity of P_H^h and R_h^H (see Eqs. (8) and (9)), the number of terms that need to be summed to calculate $\widetilde{G}_{ij}$ is not as many as Eq. (36) suggests. To see this, for each coarse-grid point $\hat{\mathbf{x}}_j$, we first introduce a new set of indices $\mathscr{E}_j$ defined as follows:

$$\mathscr{E}_j = \{m : \ j \in \mathscr{I}_m\}. \tag{37}$$

Then by Eq. (8) and $R_h^H = \left(P_H^h\right)^T$, we can rewrite Eq. (36) as

$$\widetilde{G}_{ij} = \sum_{l \in \mathscr{E}_i} w_{li} \cdot \left\{ \sum_{n \in \mathscr{E}_j \cap \mathscr{N}_l} w_{nj} \cdot K_{ln} + \sum_{n \in \mathscr{E}_j \setminus \mathscr{N}_l} w_{nj} \cdot \left[\sum_{q \in \mathscr{I}_n} w_{nq} \cdot \left(\sum_{r \in \mathscr{I}_l} w_{lr} \cdot \widehat{K}_{rq} \right) \right] \right\}. \tag{38}$$

In the special case where the coarse grid is nested into the fine grid and R_{ij} is defined by Eq. (9) instead, if $\mathbf{x}_m$ is the fine-grid point that coincides with $\hat{\mathbf{x}}_i$, then Eq. (36) simply becomes

$$\widetilde{G}_{ij} = \sum_{n \in \mathscr{E}_j \cap \mathscr{N}_l} w_{nj} \cdot K_{mn} + \sum_{n \in \mathscr{E}_j \setminus \mathscr{N}_l} w_{nj} \cdot \left[\sum_{q \in \mathscr{I}_n} w_{nq} \cdot \left(\sum_{r \in \mathscr{I}_m} w_{mr} \cdot \widehat{K}_{rq} \right) \right]. \tag{39}$$

In the rest of this paper, we use A_H to denote the approximate Galerkin projection of A_h onto the coarse grid, that is, the (i, j) block of A_H is $\widetilde{G}_{ij}$ defined in Eq. (38) or Eq. (39) instead of the exact G_{ij} defined in Eq. (32).

We still need to specify what the set of indices $\mathscr{N}_l$ is in Eqs. (38) and (39). We wish to identify the fine-grid points close to $\mathbf{x}_l$ without calculating the pair-wise distances between fine-grid points. The main idea is the following: if $\mathbf{x}_l$ is adjacent to $\hat{\mathbf{x}}_i$, $\mathbf{x}_n$ is adjacent to $\hat{\mathbf{x}}_j$, and $\hat{\mathbf{x}}_i$ is close to $\hat{\mathbf{x}}_j$, then $\mathbf{x}_l$ is considered close to $\mathbf{x}_n$.

We first consider the case where the structures are parameterized prior to discretization. We consider $\hat{\mathbf{x}}_i$ to be close to $\hat{\mathbf{x}}_j$ if the parameter values that they correspond to are close to one another. In the case of parametric curves, let s_i denote the grid point in the parameter space that $\hat{\mathbf{x}}_i$ corresponds to. We define $\mathscr{N}_l$ as

$$\mathscr{N}_l = \bigcup_{i \in \mathscr{I}_l} \left\{ n \in \mathscr{E}_j : \ \left| s_j - s_i \right| \le \gamma \text{ and } \hat{\mathbf{x}}_j \text{ is on the same structure as } \hat{\mathbf{x}}_i \right\} \tag{40}$$

where $\gamma > 0$ is a constant. In the case of parametric surfaces, let (u_i, v_i) be the grid point in the parameter space that $\hat{\mathbf{x}}_i$ corresponds to. Similarly, we define $\mathscr{N}_l$ as

$$\mathscr{N}_l = \bigcup_{i \in \mathscr{I}_l} \left\{ n \in \mathscr{E}_j : \ \left| u_j - u_i \right| \le \gamma_1, \ \left| v_j - v_i \right| \le \gamma_2, \text{ and } \hat{\mathbf{x}}_j \text{ is on the same structure as } \hat{\mathbf{x}}_i \right\} \tag{41}$$

where γ_1, γ_2 are constants. Identifying the fine-grid points close to $\mathbf{x}_l$ using Eq. (40) or Eq. (41) only entails calculating the pair-wise distances between the coarse-grid points in the parameter space.

In the case of no parameterization, we simply define $\mathscr{N}_l$ as

$$\mathscr{N}_l = \bigcup_{i \in \mathscr{I}_l} \left\{ n \in \mathscr{E}_j : \ \left\| \hat{\mathbf{x}}_j - \hat{\mathbf{x}}_i \right\|_2 \leq \gamma, \text{ and } \hat{\mathbf{x}}_j \text{ is on the same structure as } \hat{\mathbf{x}}_i \right\}. \tag{42}$$

where γ is a constant. Finding it entails calculating the pair-wise distances between the coarse-grid points on the structures.

Remark 3. Since the coarsening of the grid (Section 4.1), construction of the transfer operators (Section 4.2), and approximation of the Galerkin projection (Section 4.3) can all be performed structure by structure, these processes are easily parallelizable.

5. A block Gauss-Seidel smoother

The classic point Gauss-Seidel method has long been the smoother of choice in the multigrid methods for solving discretized elliptic PDEs on an Eulerian grid. In such a problem, the coefficient matrix is sparse, which allows for efficient implementation of the Gauss-Seidel iteration. Furthermore, since the eigenvectors of the coefficient matrix associated with large eigenvalues are highly oscillatory, a small number of Gauss-Seidel iterations suffice to effectively remove the high-frequency terms in the solution error.

The coefficient matrix of interest here is large, dense, and generated by a kernel function acting on discretized interacting structures. As a result, the point Gauss-Seidel iterations can be computationally expensive. Moreover, it is not necessarily the case that large eigenvalues of this matrix correspond to highly oscillatory eigenvectors. We observe that a block Gauss-Seidel method can serve as an effective smoother for this type of matrices, where the blocks are determined by a proximity-based partition of the Lagrangian grid points. This is fundamentally due to the decay of the kernel function as the distance between two points grows. We describe the block Gauss-Seidel smoother in Section 5.1 and discuss its efficient implementation in Section 5.2.

5.1. Description of the algorithm

The point Gauss-Seidel method is based on the decomposition of an $n \times n$ matrix A as $A = D + L + U$, where D, L, and U are diagonal, lower-triangular, and upper-triangular, respectively. Given an initial guess $f^{\langle 0 \rangle}$, the kth iteration of this method updates the solution to $Af = u$ by

$$f^{\langle k \rangle} = f^{\langle k-1 \rangle} + (L + D)^{-1} \left(u - A f^{\langle k-1 \rangle} \right) \tag{43}$$

where $f^{\langle k \rangle}$ denotes the kth iterate. Since D is diagonal and L is lower-triangular, we can update $f^{\langle k \rangle}$ entry by entry as

$$f_i^{\langle k \rangle} = \frac{1}{a_{ii}} \left(u_i - \sum_{j=1}^{i-1} a_{ij} f_j^{\langle k \rangle} - \sum_{j=i+1}^{n} a_{ij} f_j^{\langle k-1 \rangle} \right) \tag{44}$$

for $i = 1, \ 2, \ \cdots, \ n$, where $f_i^{\langle k \rangle}$ is the ith entry of $f^{\langle k \rangle}$, u_i is the ith entry of u, and a_{ij} is the (i, j) entry of A. For a positive definite A, the convergence of the Gauss-Seidel method is guaranteed by the Ostrowski-Reich theorem [48,49]. More generally, if the spectral radius of the iteration matrix

$$G = I - (L + D)^{-1} A = I - (A - U)^{-1} A, \tag{45}$$

associated with Eq. (43), denoted by $\rho(G)$, is less than 1, then the Gauss-Seidel method converges as well [50,51].

Given a block partition of A, we can extend the point Gauss-Seidel iteration to a block version. Let

$$D = \begin{bmatrix} A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{pp} \end{bmatrix}, \quad L = \begin{bmatrix} O & & & \\ A_{21} & O & & \\ \vdots & \vdots & \ddots & \\ A_{p1} & A_{p2} & \cdots & O \end{bmatrix}, \quad U = \begin{bmatrix} O & A_{12} & \cdots & A_{1p} \\ & O & \cdots & A_{2p} \\ & & \ddots & \vdots \\ & & & O \end{bmatrix}, \tag{46}$$

where A_{ij} denotes the (i, j) block of A and A_{ii} is square. Like in the point Gauss-Seidel method, we can update $f^{\langle k \rangle}$ segment by segment as

$$f_i^{\langle k \rangle} = A_{ii}^{-1} \left(u_i - \sum_{j=1}^{i-1} A_{ij} f_j^{\langle k \rangle} - \sum_{j=i+1}^{p} A_{ij} f_j^{\langle k-1 \rangle} \right) \tag{47}$$

for $i = 1,\ 2,\ \cdots,\ p$, where $f_i^{\langle k\rangle}$ and u_i denote the ith segments of $f^{\langle k\rangle}$ and u, respectively. (The number of entries in $f_i^{\langle k\rangle}$ and u_i equals the number of columns in A_{ij}.) As for the point Gauss-Seidel method, the block Gauss-Seidel method converges if $\rho(G) < 1$, where G is defined in Eq. (45) and D, L, and U are defined in Eq. (46).

In this paper, A is $3N \times 3N$ and generated by a kernel function $K(\mathbf{x}, \mathbf{y}) : \mathbb{R}^3 \times \mathbb{R}^3 \longrightarrow \mathbb{R}^{3\times 3}$, such as the one associated with the MRS (see Section 2), acting on every pair from N Lagrangian grid points $\{\mathbf{x}_i\}_{i=1}^N$. By partitioning the grid points into p groups, where the ith group consists of the N_i points numbered by $\left\{\sum_{j=1}^{i-1} N_j + k\right\}_{k=1}^{N_i}$, we obtain the following partition of A into $p \times p$ blocks: for $i, j = 1,\ 2,\ \cdots,\ p$, the (i, j) block, A_{ij}, is $3N_i \times 3N_j$ and represents the interactions between the ith and jth groups of points; in particular, the ith block on the diagonal, A_{ii}, represents the interactions among the points in the ith group. For the block Gauss-Seidel method to be effective, we wish $\rho(G)$ to be small, implying that D should be dominant in A. Since D represents the self-interactions of the p groups of points and $\|K(\mathbf{x}, \mathbf{y})\|$ increases if $\mathbf{x}$ is closer to $\mathbf{y}$, it is desirable that the points within each group are close to one another. For example, we can order the grid points in such a way that the ones belonging to the ith structure are numbered consecutively from $\sum_{j=1}^{i-1} N_j + 1$ to $\sum_{j=1}^{i} N_j$ for $i = 1,\ 2,\ \cdots,\ p$. If the grid points are not ordered based on proximity, we need to re-order them first and then apply the block Gauss-Seidel method to $P^T AP$ instead, where P is an $n \times n$ permutation matrix. The difference between the resulting block Gauss-Seidel iteration and Eq. (47) is only technical. Therefore, we continue to use Eq. (47) for simplicity.

We observe from numerical experiments that as long as the grid points are grouped appropriately, a small number of the block Gauss-Seidel iterations (47) serve as an effective smoother for the proposed multigrid method.

5.2. Implementation details

For a large dense matrix A, the block Gauss-Seidel iteration given in Eq. (47) introduces the following two challenges: solving a linear system whose coefficient matrix is A_{ii} (a block on the diagonal of A) and matrix-vector multiplication involving A_{ij} (an off-diagonal block in A). Both types of sub-matrices are similar to A in structure. In this subsection, we discuss how to perform these calculations efficiently.

For effective smoothing, we find that partitioning A into large blocks is sometimes necessary, making A_{ii} too large for a direct solver. In this case, we apply the proposed multigrid method to invert A_{ii} as well. Compared to the "outer" multigrid iteration applied to $Af = u$, since A_{ii} is considerably small than A, fewer levels of coarsening are needed in the "inner" multigrid iteration applied to Eq. (47); in addition, we observe that there is no need to carry out the inner multigrid iteration as accurately.

The block-vector multiplication involving A_{ij} in Eq. (47) can also be approximated on a coarser grid as follows. We replace A_{ij} by

$$P_i \widehat{A}_{ij} R_j \tag{48}$$

where $\widehat{A}_{ij}$ represents the interactions between the ith and jth groups of coarse-grid points calculated using the original kernel function K, $R_j = P_j^T$ is the restriction operator for the jth group of points, and P_i is the prolongation operator for the ith group of points (see Eq. (22)). Note that P_i, R_j are sparse and $\widehat{A}_{ij}$ is dense but smaller than A_{ij}. We refer to this technique as the inexact block-vector multiplication. Numerical experiments show that a rather coarse grid can be used in (48) to substantially reduce the computational cost of Eq. (47) without degrading the effectiveness of the smoother.

Remark 4. We propose to use a block Gauss-Seidel smoother instead of a block Jacobi smoother because numerical experiments show that the former requires far fewer iterations when applied to the MRS matrices of interest in this study. However, since the block Gauss-Seidel smoother entails inverting the block lower-triangular matrix $L + D$ whereas only the block-diagonal matrix D needs to be inverted in the block Jacobi method, the former is more difficult to parallelize. Several parallel Gauss-Seidel algorithms have been developed in [52–54] that may be adapted to the block Gauss-Seidel algorithm proposed here. In [52], several parallelization strategies for dense Gauss-Seidel methods based on GPU and multi-threaded CPU implementations were discussed. In [53], a distributed memory parallel Gauss–Seidel algorithm was proposed by using a row-block partition and a torus-wrapping technique. In [54], a parallel Jacobi-embedded Gauss-Seidel method was developed by rewriting a Gauss-Seidel iteration into a highly parallelizable Jacobi-like iteration.

6. Numerical experiments

We demonstrate the effectiveness of the proposed multigrid algorithm on large-scale linear systems with dense coefficient matrices arising from the simulation of interacting structures immersed in a Stokesian fluid. The performance of this method is also compared with the preconditioned Generalized Minimal RESidual method (GMRES). We defer numerical experiments on free swimmers that undergo rigid translation and rotation to Section 7, where additional constraints need to be imposed besides Eq. (4) to maintain force and torque balance.

The preconditioner used to accelerate GMRES is block-diagonal. It is obtained by dividing the grid points into clusters and allowing within-cluster interactions only. The clusters are determined by the proximity of the points or their membership in the structures. More precisely, in the first method, we evenly divide the computational domain into 8^ρ boxes where $\rho \geq 1$

is an integer and consider all the points in a box as a cluster, as in [19]. In the second method, all the points belonging to the same structure are considered to be one cluster. For every example in this section and Section 7, we explore both methods and select the one that leads to the most efficient preconditioned GMRES.

Since the swimmers considered in this section are modeled as parametric curves, in the multigrid method, we use the method described in Section 4.1.1 to coarsen the grid and the method described in Section 4.2.1 to construct the transfer operators. The methods in Sections 4.1.1 and 4.2.1 for swimmers represented by discrete points are considered in Section 7.1.

The smoother employed in the multigrid method is a single block Gauss-Seidel iteration proposed in Section 5, where the block-vector multiplication involving the off-diagonal blocks in Eq. (47) is performed inexactly as described in Section 5.2.

Recall that the multigrid method applied to the fine-grid problem Eq. (4) gives rise to the following two types of subsidiary linear systems of smaller sizes:

- linear systems whose coefficient matrix is A_H, the representation of A_h on a coarser grid
- linear systems whose coefficient matrix is A_{ii}, a block on the diagonal of A_h or A_H after it has been partitioned properly in the block Gauss-Seidel smoother.

Note that A_H and A_{ii} are dense as well. In our numerical experiments, if they are sufficiently small, a direct method will be used; otherwise, an iterative method such as the multigrid method or the preconditioned GMRES will again be applied.

The initial guess, $f^{\langle 0\rangle}$, to f_h in the multigrid method is obtained by prolonging the solution obtained on a coarser grid, that is, $f^{\langle 0\rangle} = P_H^h \left(A_H^{-1}\left(R_h^H u_h\right)\right)$ where A_H is the coarse-grid representation of A_h. (The runtime of the multigrid method reported in this section includes the time spent on computing the initial guess.) It is simply taken to be the zero vector in the preconditioned GMRES.

In all the numerical experiments in this section, we use the relative residual at the kth iteration defined as

$$\eta^{\langle k\rangle} = \frac{\|u_h - A_h f^{\langle k\rangle}\|_2}{\|u_h\|_2}. \tag{49}$$

to measure the accuracy of a solver, where $f^{\langle k\rangle}$ is the kth iterate of f_h computed by the solver.

All the numerical experiments are performed in MATLAB R2018b on a virtual machine equipped with an Intel Xeon CPU 2.30 GHz.

6.1. Bacterial carpets

There have been many studies on the flow around a bacterial carpet, which consists of a large group of flagellated bacteria attached to a surface [55,56,2]. Such a construction can be utilized to perform fluid mixing and transport at a small scale [57,2]. Here, we model the bacterial flagella as identical rotating helices as in [57], where the rotational velocities along the helices are prescribed and the forces along them need to be solved. Once the forces have been calculated, the flow around the carpet can be calculated using Eq. (2).

We consider an uniform array of rotating helices that emanate from an infinite, planar, solid, and stationary wall at $z = 0$. At time $t = 0$, the centerline $\boldsymbol{\mathcal{X}}_0(s) = (x(s), y(s), z(s))^T$ of an upright helix based at the origin and contained in the semi-infinite domain $\left\{(x, y, z)^T : z \geq 0\right\}$ is parameterized as follows:

$$x(s) = \alpha \tanh(\tau s)\cos(2\pi s/\lambda + \phi), \tag{50}$$

$$y(s) = \alpha \tanh(\tau s)\sin(2\pi s/\lambda + \phi), \tag{51}$$

$$z(s) = s, \tag{52}$$

where $0 \leq s \leq L$, λ is the helical pitch, τ is a tapering parameter, ϕ is the phase angle, and α is the radius of the helix. At time t, the position of any point on the helix based at $\mathbf{B}$ is prescribed as

$$\boldsymbol{\mathcal{X}}(s, t) = R(t)\boldsymbol{\mathcal{X}}_0(s) + \mathbf{B}, \tag{53}$$

where $R(t)$ is a time-dependent rotation matrix given by

$$R(t) = \begin{bmatrix} \cos(\omega t) & -\sin(\omega t) & 0 \\ \sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{54}$$

and ω is the rotational speed. The values of L, α, λ, τ, ϕ, and ω are given in Table 1. The value of the regularization parameter ϵ in the MRS is chosen to be 0.01. We position the base points of the carpet of helices to form a uniform grid of spacing 3α in the x and y directions on the plane $z = 1.01\epsilon$, which lies slightly above the wall.

In the fine grid, there are 161 grid points per helix resulting from the uniform grid of s on $[0, L]$ whose spacing is $h = L/160$. The velocities at the grid points can be obtained by differentiating Eq. (53) with respect to time t. The forces

Table 1
Parameter values used in the case of a bacterial carpet.

Parameter	Value
height (L)	2.2
helical radius (α)	0.085
tapering parameter (τ)	1000
helical pitch (λ)	$2\pi\alpha$
angular speed (ω)	-2π

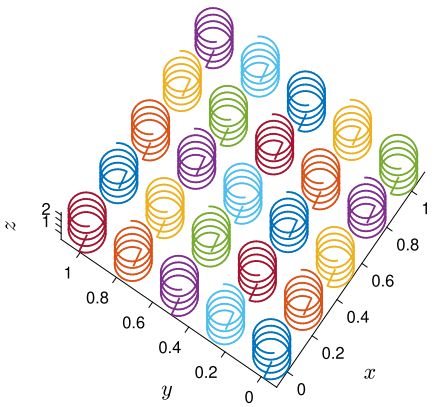

Fig. 3. The 5×5 bacterial carpet at $t = 0$.

that they exert to the surrounding fluid to induce said velocities can be recovered by solving Eq. (4). To incorporate the wall, the method of images for regularized Stokeslets [43] is used to enforce a no-slip ($\mathbf{u} = 0$) boundary condition. In this formulation, an image source consisting of a regularized Stokeslet, doublet, dipole, and rotlet is mirrored across the boundary.

6.1.1. A small carpet

We first consider a 5×5 carpet at $t = 0$ (see Fig. 3) to demonstrate the effectiveness of the approximate Galerkin projection proposed in Section 4.3 and the block Gauss-Seidel smoother proposed in Section 5. We apply a two-grid method to solve Eq. (4), where there are 21 points per helix in the coarse grid resulting from the uniform grid of s with spacing $H = 8h$. The coefficient matrices A_h and A_H corresponding to the two grids are 12075×12075 and 1575×1575, respectively. We calculate the initial guess as $f^{\langle 0 \rangle} = P_H^h \left(A_H^{-1} \left(R_h^H u_h \right) \right)$. To perform the inexact block-vector multiplication within the smoother, we discretize the helices using an even coarser grid that corresponds to the uniform grid of s with spacing $4H$.

In the block Gauss-Seidel smoother, the matrix A_h is partitioned into $25 \times 25 = 625$ blocks of size 483×483, where the (i, j)th block represents the interactions between the ith and jth helices. To demonstrate the damping of high-frequency oscillations in solution errors by the smoother, we compare the pre- and post-smoothing solution errors in the first iterate of the two-grid method. The portions of both errors corresponding to the first helix (that is, the first 483 elements of both error vectors), with their x, y, and z components separated, are displayed in the three subplots of Fig. 4(a). They show that one iteration of the proposed block Gauss-Seidel method is quite effective at damping the error. We also examine the effect of the inexact block-vector multiplication within the smoother. More specifically, we calculate the first iterate in two ways that only differ in whether the block-vector multiplication in the smoother is exact or inexact. The portions of their errors corresponding to the last (25th) helix (that is, the last 483 elements of both error vectors), with their x, y, and z components separated, are displayed in the three subplots of Fig. 4(b). They show that calculating the block-vector products inexactly within the smoother has little effect on its effectiveness.

Next, we examine the effect of the approximate Galerkin projection by calculating $f^{\langle 0 \rangle}$ in two ways that only differ in whether A_H is constructed by the exact or inexact Galerkin projection. The portions of the two initial guesses corresponding to the first helix (that is, the first 483 elements in the two vectors), with their x, y, and z components separated, are displayed in the three subplots of Fig. 4(c). They indicate that the approximate Galerkin projection can produce a sufficiently good coarse-grid matrix.

Finally, in Fig. 4(d), we plot the decay of the relative residual $\eta^{\langle k \rangle}$ defined in Eq. (49) as the iteration count k increases in the semi-log scale. The circle at $k = 1, 2, 3, 4$ represents $\eta^{\langle k \rangle}$ after the kth complete two-grid iteration. (The circle at $k = 0$ represents the relative residual $\eta^{\langle 0 \rangle}$ associated with the initial guess $f^{\langle 0 \rangle}$.) The triangle at $k + \frac{1}{2}$ for $k = 0, 1, 2, 3$ represents the relative residual after smoothing (Step 1 in Section 3) but before coarse-grid correction (Step 4 in Section 3). Inexact block-vector multiplication is used in the smoother, and approximate Galerkin projection is applied to construct the

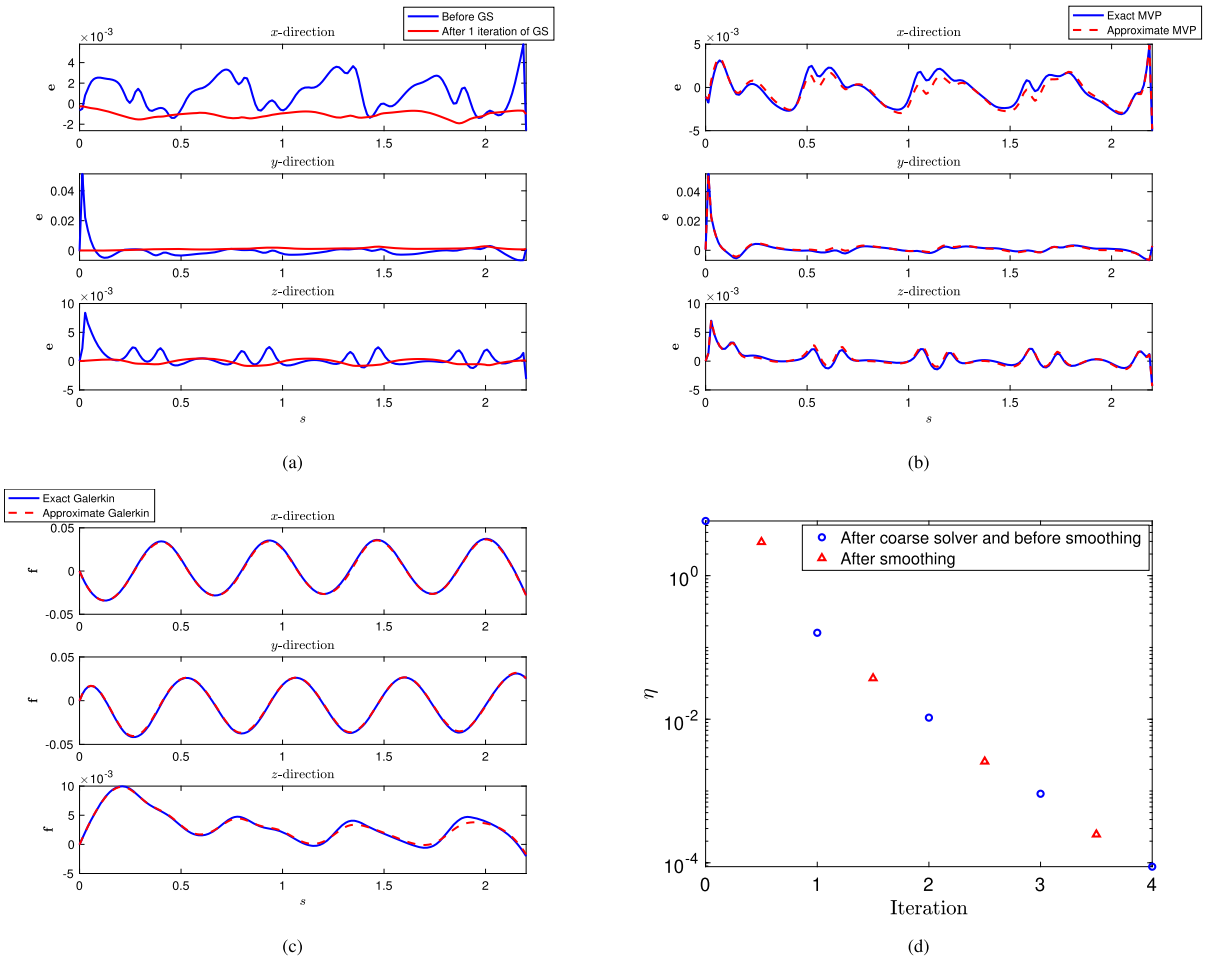

Fig. 4. (Color online) Performance of the multigrid method applied to the 5×5 bacterial carpet. (a) Solution errors in the first helix before and after smoothing (blue vs. red) in the first iteration. (b) Solution errors in the last helix after smoothing via exact and inexact block-vector multiplication (blue vs. red) in the first iteration. (c) Initial guesses calculated using exact and inexact Galerkin projection (blue vs. red). (d) Decay of the relative residual $\eta^{\langle k \rangle}$ (blue circle) as the iteration count k increases in the two-grid method. The relative residual $\eta^{\langle k+\frac{1}{2} \rangle}$ of each intermediate iterate (red triangle) obtained after smoothing and before coarse-grid correction is also shown.

coarse-grid matrix. Fig. 4(d) demonstrates the effectiveness of both steps and how alternating between them leads to the convergence of the two-grid method.

6.1.2. *Large carpets*

We now apply the proposed multigrid method to solve for the hydrodynamic forces generated by 10×10, 20×20, and 25×25 carpets of uniformly placed identical helices at time $t = 0$. Its performance is compared with the performance of the preconditioned GMRES method, where the preconditioner is block-diagonal and has been used to precondition MRS matrices in [19]. The sizes of the fine-grid matrices, A_h, are 48300×48300, 193200×193200, and 301875×301875 in the three cases, respectively. We do not construct these matrices explicitly in either the multigrid method or the GMRES method.

We employ a three-grid method where the two coarse grids result from the uniform grids of s with spacings $H = 8h$ (21 points per helix) and $2H$ (11 points per helix), respectively. When performing the inexact block-vector multiplication within the block Gauss-Seidel smoother, we discretize the helices using an even coarser grid corresponding to the uniform grid of s with spacing $4H$ (6 points per helix). We use $f^{\langle 0 \rangle} = P_{2H}^{h} \left(A_{2H}^{-1} \left(R_{h}^{2H} u_h \right) \right)$ as the initial guess of the three-grid method. Additional implementation details of the multigrid method can be found in Section A.1.

We perform the following two sets of experiments. In the first set, we consider the 10×10 and 20×20 carpets, calculate the matrix-vector products involving A_h exactly, and set the stopping criterion to be $\eta^{\langle k \rangle} < 10^{-5}$. In addition, the matrix-vector products involving A_H, the middle-grid coefficient matrix, are approximated by the Kernel Independent Fast Multipole Method (KIFMM) [58,59]. Table 2 shows the runtime, iteration count, and relative residual of the final iterate for each solver in each case. We observe that in both examples, the runtime of the multigrid method is significantly shorter

Table 2
Comparison of the multigrid method, the preconditioned GMRES method, and the GMRES method without a preconditioner in the case of a 10×10 bacterial carpet and the case of a 20×20 bacterial carpet.

Solver	Size of carpet	Runtime (s)	# of iter.	Final rel. error
Multigrid	10×10	380.28	5	$4.83 \cdot 10^{-6}$
Preconditioned GMRES	10×10	842.51	24	$6.66 \cdot 10^{-6}$
GMRES	10×10	1117.37	34	$7.18 \cdot 10^{-6}$
Multigrid	20×20	3424.54	5	$6.64 \cdot 10^{-6}$
Preconditioned GMRES	20×20	12797.73	32	$5.83 \cdot 10^{-6}$
GMRES	20×20	17951.89	45	$9.49 \cdot 10^{-6}$

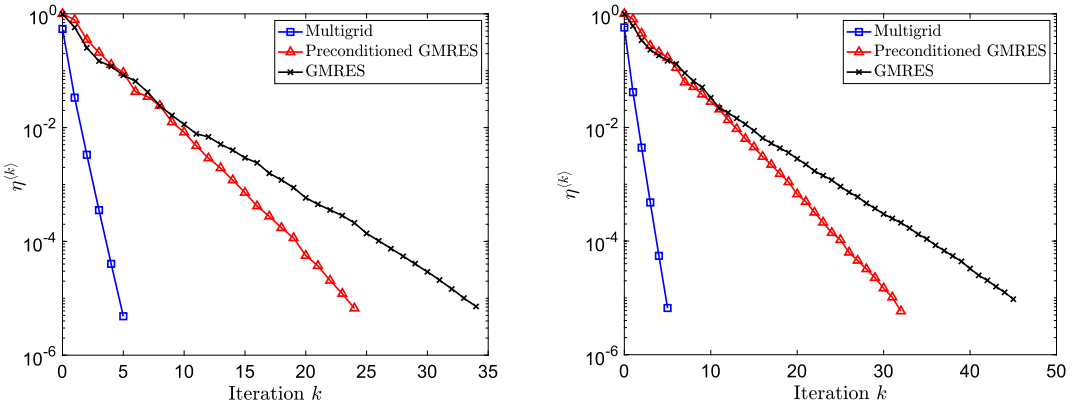

Fig. 5. (Color online) Decay of $\eta^{\langle k \rangle}$ as k increases for the multigrid method (blue squares), the preconditioned GMRES method (red triangles), and the GMRES method without a preconditioner (black crosses) in the case of a 10×10 bacterial carpet (a) and the case of a 20×20 carpet (b).

Table 3
Comparison of the multigrid method and the preconditioned GMRES method in the case of a 20×20 bacterial carpet and the case of a 25×25 bacterial carpet.

Solver	Size of carpet	Runtime (s)	# of iter.	Final rel. error
Multigrid	20×20	1332.26	2	$4.80 \cdot 10^{-4}$
preconditioned GMRES	20×20	3401.42	19	$9.37 \cdot 10^{-4}$
Multigrid	25×25	1997.50	2	$5.26 \cdot 10^{-4}$
preconditioned GMRES	25×25	4722.00	21	$8.16 \cdot 10^{-4}$

than the runtime of the preconditioned GMRES method; furthermore, the former scales much better as the size of the carpet grows. The decay of $\eta^{\langle k \rangle}$ as k increases is also illustrated in Fig. 5 for both solvers. (Results of the GMRES method without a preconditioner are also included in Table 2 and Fig. 5 for reference.)

In the second set of experiments, we consider the 20×20 and 25×25 carpets, approximate the matrix-vector products involving A_h by the KIFMM instead,[1] and relax the stopping criterion to $\eta^{\langle k \rangle} < 10^{-3}$ accordingly. As in the previous set of experiments, the matrix-vector products involving A_H are approximated by the KIFMM. Table 3 shows the runtime, iteration count, and relative residual of the final iterate for each solver in each case. As observed in the previous set of experiments, the multigrid method converges much more rapidly in terms of both runtime and iteration count. Interestingly, applying the KIFMM to A_h makes the multigrid method and the preconditioned GMRES method scale similarly well. This is mainly because the preconditioned GMRES method entails about ten times as many matrix-vector products involving A_h, which makes the accelerating effect of the KIFMM more pronounced.

6.2. A carpet of lung cilia

The motility of cilia is crucial for fluid transport and clearance of foreign particles in many organisms. Many studies have used the MRS to simulate the flow around a carpet of rhythmically beating cilia [60–62,19]. As in [19], we consider a

[1] We observe that the accuracy of the matrix-vector products involving A_h affects the accuracy of the multigrid method significantly more than the matrix-vector products involving A_H. While both are approximated by the KIFMM in this set of experiments, we vary the parameters of the KIFMM such that the former are calculated more accurately.

Table 4
Parameter values in the case of a 25×25 cilia carpet.

Parameter	Value
length (L)	6
angular beat frequency (σ)	30π
regularization parameter (ϵ)	0.05

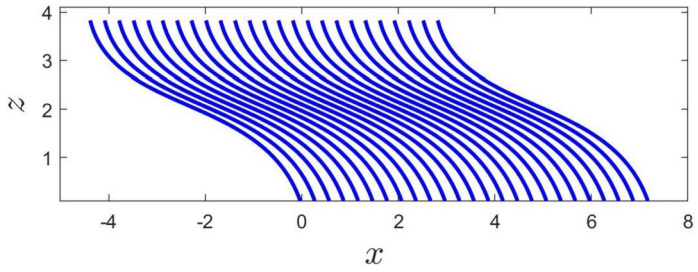

Fig. 6. A row of the 25×25 cilia carpet at $t = 0$.

Table 5
Comparison of the multigrid method and the preconditioned GMRES method in the case of a 25×25 cilia carpet.

Solver	Runtime (s)	# of Iter.	Final rel. error
Multigrid	5518.38	2	$7.7 \cdot 10^{-4}$
preconditioned GMRES	22650.94	49	$9.95 \cdot 10^{-4}$

large carpet of identical cilia tethered to an infinite, planar, solid, and stationary wall at $z = 0$ and contained in the semi-infinite domain $\{(x, y, z)^T : z \geq 0\}$. Each cilium beats in the plane $y = y_b$ where y_b is the y-coordinate of its base point $\mathbf{B} = (x_b, y_b, 0)$ and is modeled by a spatiotemporal curve $\boldsymbol{\mathcal{X}}(s,t) = (x(s,t), y_b, z(s,t))^T$ derived in [63] based on the beat pattern of rabbit tracheal cilia experimentally observed in [64], that is,

$$\begin{bmatrix} x(s,t) \\ z(s,t) \end{bmatrix} = \frac{1}{2}\mathbf{a}_0(s) + \sum_{n=1}^{N_0} \mathbf{a}_n(s)\cos(n\sigma t) + \mathbf{b}_n(s)\sin(n\sigma t) + \begin{bmatrix} x_b \\ 0 \end{bmatrix}, \tag{55}$$

where t is time, $s \in [0, L]$ is the arclength along the cilium, σ is its angular beat frequency, and $\mathbf{a}_n(s)$, $\mathbf{b}_n(s)$ are vectors of polynomials of s. Note that although each cilium is planar, the cilia carpet is 3D as it consists of twenty-five rows and columns of cilia whose base points form a 25×25 uniform grid with spacing 0.3 in the x direction and y direction on the wall. The values of L, σ, and the regularization parameter ϵ used in the MRS can be found in Table 4.

Similar to the case of a bacterial carpet, each cilium is discretized by a number of grid points, and the velocities at these points can be found by differentiating Eq. (55) with respect to time t. By solving Eq. (4), we can find the forces that the grid points must exert to the surrounding fluid to induce said velocities. The wall is again incorporated using the method of images. In the fine grid, we use 130 points per cilium, resulting from the uniform grid of s with spacing $h = L/129$. Accordingly, the size of A_h is 241875×241875.[2]

We consider the cilia carpet at $t = 0$, a row of which is shown in Fig. 6. We employ a three-grid method where the two coarse grids result from the uniform grids of s with spacings $H = 4h$ (33 points per cilium) and $2H$ (17 points per cilium), respectively. When performing the inexact block-vector multiplication within the block Gauss-Seidel smoother, we discretize the helices using an even coarser grid corresponding to the uniform grid of s with spacing $4H$ (5 points per cilium). We use $f^{\langle 0 \rangle} = P_H^h \left(A_H^{-1} \left(R_h^H u_h \right) \right)$ as the initial guess of the three-grid method. Additional implementation details of the multigrid method can be found in Section A.2.

We again compare the performance of the three-grid method and the preconditioned GMRES method at solving for the hydrodynamic forces. As in the second set of experiments on bacterial carpets in Section 6.1.2, the matrix-vector products involving A_h and A_H are approximated using the KIFMM, and the stopping criterion is $\eta^{\langle k \rangle} < 10^{-3}$. Table 5 shows the runtime, iteration count, and relative residual of the final iterate for each solver. Compared to the preconditioned GMRES method, the multigrid method requires 95% fewer iterations and 75% less runtime.

[2] Since the base point of each cilium does not move and exert no force to the fluid, only 128 points per helix are taken into account in Eq. (4). Including it would make A_h singular.

7. Application to free swimmers

In this section, we discuss the application of the proposed multigrid algorithm to simulate free swimmers that undergo rigid translation and rotation. We must impose additional constraints in these simulations, which lead to a larger and more challenging linear system. Solving it iteratively entails solving auxiliary linear systems whose coefficient matrix is generated by a kernel function acting on discretized interacting structures, as the one considered in previous sections.

Assume that there are n free swimmers and the ith one is represented by N_i grid points. Then for the ith swimmer, we need to impose the following two constraints:

$$\sum_{k=1}^{N_i} \mathbf{f}_k^i = \mathbf{0} \qquad \text{and} \qquad \sum_{k=1}^{N_i} \left(\mathbf{x}_k^i - \mathbf{x}_c^i\right) \times \mathbf{f}_k^i = \mathbf{0}, \tag{56}$$

where $\mathbf{f}_k^i$ is the hydrodynamic force exerted by the kth grid point located at $\mathbf{x}_k^i$, and $\mathbf{x}_c^i$ is the center of the swimmer. The constraints in Eq. (56) imply that the swimmer undergoes rigid translation and rotation. Consequently, if the velocities $\{\mathbf{v}_k^i\}_{k=1}^{N_i}$ are prescribed to the grid points $\{\mathbf{x}_k^i\}_{k=1}^{N_i}$, they will induce a translational velocity $\mathbf{U}^i$ and a rotational velocity $\boldsymbol{\Omega}^i$ for the entire swimmer; that is, the total velocity at $\mathbf{x}_k^i$ can be written as $\mathbf{u}_k^i = \mathbf{v}_k^i + \mathbf{U}^i + \boldsymbol{\Omega}^i \times (\mathbf{x}_k^i - \mathbf{x}_c^i)$. Let $N = \sum_{i=1}^{n} N_i$ be the total number of grid points. Taking into account the six constraints in Eq. (56) for every i and formulating $\mathbf{u}_k^i$ using the MRS again, we obtain the following $(3N + 6n) \times (3N + 6n)$ linear system

$$\underbrace{\begin{bmatrix} A_h & B_h^T \\ B_h & O_{6n} \end{bmatrix}}_{C_h} \underbrace{\begin{bmatrix} f_h \\ u_h \end{bmatrix}}_{g_h} = \underbrace{\begin{bmatrix} v_h \\ 0_{6n} \end{bmatrix}}_{w_h}, \tag{57}$$

where $A_h \in \mathbb{R}^{3N\times 3N}$ is the coefficient matrix in Eq. (4), $B_h \in \mathbb{R}^{6n\times 3N}$ is sparse, O_{6n} is the $6n \times 6n$ zero matrix, $f_h \in \mathbb{R}^{3N}$ is formed by concatenating the forces exerted by the N grid points as in Eq. (4), $v_h \in \mathbb{R}^{3N}$ is formed by concatenating the velocities imposed at the grid points, 0_{6n} is the $6n \times 1$ zero vector, and $u_h \in \mathbb{R}^{6n}$ is formed by concatenating all n pairs of $\mathbf{U}^i$ and $\boldsymbol{\Omega}^i$. Solving Eq. (57) gives us the hydrodynamic forces exerted by all the grid points, which can in turn be used in the MRS to calculate the fluid velocity at any point; in addition, it gives us the translational and rotational velocities of each swimmer induced by the prescribed motion. Linear systems whose coefficient matrices are large, sparse and have a similar block structure as C_h arise from mixed finite element discretization of the Stokes and Navier-Stokes equations. Iterative methods for such a problem, also known as the saddle point problem, have been studied extensively [65]. They often entail solving auxiliary linear systems whose coefficient matrix is A_h, the $(1,1)$ block of C_h. We emphasize again that unlike a finite-element or finite-difference matrix, A_h is dense in this study.

For example, a generalized least square formulation of Eq. (57) has been given in [66], where u_h is written as

$$u_h = \min_{\mathfrak{z}\in\mathbb{R}^{6n}} \left\| v_h - B_h^T \mathfrak{z} \right\|_{A_h^{-1}}. \tag{58}$$

Once u_h is found by solving the minimization problem (58), f_h can be calculated as

$$f_h = A_h^{-1}(v_h - B_h^T u_h). \tag{59}$$

Eq. (58) can be solved using LSQR$\left(A_h^{-1}\right)$, a generalized LSQR method [67] developed in [65]. This algorithm entails solving linear systems whose coefficient matrix is A_h, to which the proposed multigrid method can be applied. Alternatively, we can also solve the saddle point problem Eq. (57) using the preconditioned GMRES method. A frequently used preconditioner, which is based on the block LU factorization of the coefficient matrix C_h, is a block upper triangular matrix in the form of

$$P = \begin{bmatrix} \tilde{A}_h & B_h^T \\ O & \tilde{S}_h \end{bmatrix}, \tag{60}$$

where $\tilde{A}_h$ and $\tilde{S}_h$ are preconditioners for A_h and $S_h = -B_h A_h^{-1} B_h^T$ respectively. Since

$$P^{-1} = \begin{bmatrix} \tilde{A}_h^{-1} & -\tilde{A}_h^{-1} B_h^T \tilde{S}_h^{-1} \\ O & \tilde{S}_h^{-1} \end{bmatrix}, \tag{61}$$

applying P^{-1} entails solving auxiliary linear systems whose coefficient matrices are $\tilde{A}_h$ or $\tilde{S}_h$. This type of preconditioner has been adapted to simulate free swimmers in previous work [19], where $\tilde{A}_h$ was chosen to be the block-diagonal preconditioner considered in Section 6, and

$$\tilde{S}_h = -(B_h B_h^T)\left(B_h A_h B_h^T\right)^{-1}(B_h B_h^T), \tag{62}$$

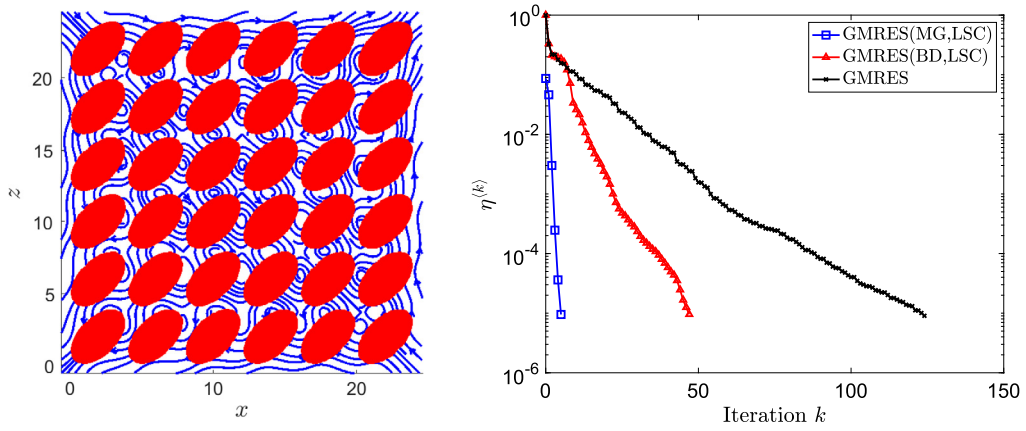

Fig. 7. (Color online) Illustration of the ellipsoidal swimmers and convergence of the solvers applied to Eq. (57) in this case. (a) Projection of the 6×6 array of ellipsoids onto the xz plane and streamlines drawn based on the x- and z-directional fluid velocities around them at $t = 0$. (b) Decay of $\eta^{\langle k\rangle}$ as k increases for GMRES(MG, LSC) (blue squares), GMRES(BD, LSC) (red triangles), and the GMRES method without a preconditioner (black crosses).

the so called Least-Squares Commutator (LSC) preconditioner [68]. Encouraged by the numerical results in Section 6, we examine the performance of P when $\tilde{A}_h$ is chosen such that the action of $\tilde{A}_h^{-1}$ is a few iterations of the proposed multigrid method.

In the two subsections that follow, we apply the GMRES method preconditioned by the block upper triangular P given in Eq. (60) to solve Eq. (57) arising from two types of free swimmers. The LSC preconditioner (62) is again used in the $(2,2)$ block of P. For the $(1,1)$ block of P, we consider both the block-diagonal preconditioner and the one whose inverse is one iteration of the multigrid method. As in Section 6, one block Gauss-Seidel iteration proposed in Section 5 serves as the smoother for the multigrid method. Let the two preconditioned GMRES methods for Eq. (57) be denoted by GMRES(BD, LSC) and GMRES(MG, LSC), respectively, which only differ in the $(1,1)$ block. We calculate the initial guess of GMRES(MG, LSC) by prolonging the approximate solution to a coarse-grid representation of Eq. (60) obtained by one LSQR$\left(A_H^{-1}\right)$ iteration, where A_H is the coarse-grid representation of A_h. (The runtime of GMRES(MG, LSC) reported in this section includes the time spent on computing the initial guess.) The initial guess of GMRES(BD, LSC) is the zero vector. Neither solver requires the construction of A_h.

In all the numerical experiments in this section, we use the relative error at the kth iteration defined as

$$\eta^{\langle k\rangle} = \frac{\|w_h - C_h g^{\langle k\rangle}\|_2}{\|w_h\|_2}. \tag{63}$$

to measure the accuracy of a solver for Eq. (57), where $g^{\langle k\rangle}$ is the kth iterate to g_h produced by the solver. We also set the stopping criterion of all solvers to be $\eta^{\langle k\rangle} < 10^{-5}$.

7.1. Free ellipsoidal swimmers

Vesicles are membranes that play a vital role in the metabolism and transport of cellular products. Models of capsules and vesicles immersed in a viscous flow have been used to study the biomembrane mechanics of red blood cell, artificial capsules in drug delivery, and the dynamics of liquid droplets [69,70]. In this subsection, we model them as free ellipsoids undergoing rigid translation and rotation. A shearing velocity is imposed on their surfaces. The forces that they exert to the surrounding fluid as well as the translational and rotational velocities induced by the prescribed motion can be found by solving the augmented linear system (57).

We consider a 6×6 array of identical ellipsoids whose centers form a uniform grid with spacing 4 on the xz plane at time $t = 0$. Each ellipsoid has semi-axes of length 1, 1.5, and 2 and is represented by 1016 points in the fine grid, resulting from stretching and rotating a unit sphere that has been discretized by a cubed-sphere grid [71]. Therefore, $n = 36$, $N_i = 1016$, and $N = 36576$ in this case. The regularization parameter ϵ used in the MRS is 0.0556, which is roughly 0.05 times the average grid spacing on the ellipsoids. The size of the coefficient matrix C_h in (57) is 109944×109944, whose $(1,1)$ block, A_h, is 109728×109728. In addition, we impose the shearing velocity $(0.2z, 0, 0)^T$ at any grid point $(x, y, z)^T$, causing each ellipsoid to translate and rotate as a whole. In Fig. 7(a), the projection of the array of ellipsoids onto the xz plane as well as the streamlines drawn based on the x- and z-directional fluid velocities at time $t = 0$ are shown.

In GMRES(MG, LSC), the action of $\tilde{A}_h^{-1}$, where $\tilde{A}_h$ is the $(1,1)$ block in the preconditioner P defined in Eq. (60), is one iteration of a two-grid method. To construct the coarse grid, we use the method described in Section 4.1.2 for the case of no parameterization and choose $\alpha = 0.33$ in Eq. (6), which leads to 2376 grid points (66 points per ellipsoid).

Table 6
Comparison of GMRES(MG, LSC), GMRES(BD, LSC), and the GMRES method without a preconditioner in the case of ellipsoidal swimmers.

Solver	Runtime (s)	# of iter.	Final rel. error
GMRES(MG, LSC)	1234.96	5	$9.51 \cdot 10^{-6}$
GMRES(BD, LSC)	2285.61	47	$9.56 \cdot 10^{-6}$
GMRES	3664.18	124	$9.07 \cdot 10^{-6}$

Accordingly, the method described in Section 4.2.1 for the case of no parameterization is used to construct the transfer operators. Within the block Gauss-Seidel smoother, we use the same coarse grid when performing the inexact block-vector multiplication. Additional implementation details of the multigrid method can be found in Section A.3. Moreover, the initial guess of GMRES(MG, LSC) is obtained by prolonging the approximate solution found by applying one LSQR iteration to the coarse-grid representation of Eq. (57). The matrix-vector products involving A_h are approximated by the KIFMM in both GMRES(MG, LSC) and GMRES(BD, LSC).[3]

In Table 6, we report the runtime, iteration count, and relative error of the final iterate of each solver. Compared to GMRES(BD, LSC), GMERS(MG, LSC) requires about 90% fewer iterations and 45% less runtime. We also display the decay of $\eta^{\langle k \rangle}$ as the iteration count k increases for each solver in Fig. 8(b). (Results of the GMRES method without a preconditioner are also included in Table 6 for reference.)

7.2. Free toroidal swimmers

The motility of doughnut-shaped swimmers in a viscous fluid has been considered in various studies [72–74] as it can provide important insights into the self-propulsion of some microorganisms, including the flagellated unicellular microbe Idionectes vortex. We consider the linear system Eq. (57) arising from a large array of free, identical toroidal swimmers.

Following [74], at time $t = 0$, we parameterize the surface of a torus centered at $(x_c, y_c, z_c)^T$ with tube radius r and minimum distance d to the center as

$$\begin{aligned} x(u, v) &= (d + r\cos v)\cos u + x_c \\ y(u, v) &= (d + r\cos v)\sin u + y_c \\ z(u, v) &= r\sin v + z_c, \end{aligned} \tag{64}$$

where $u,\ v \in [0, 2\pi)$. We choose $d = 1$ and $r = 0.25$ in Eq. (64) so that the aspect ratio of each torus is $d/r = 4$. The array consists of 16 tori whose centers form a 4×4 rectangular grid in the xz plane with spacing 3.75 in the x direction and spacing 1.25 in the z direction. In the fine grid, the surface of each torus is discretized by 4096 points resulting from a 128×32 rectangular grid in the parameter space $[0, 2\pi) \times [0, 2\pi)$, whose spacing is $\Delta u = 2\pi/128$ in the u direction and $\Delta v = 2\pi/32$ in the v direction. The regularization parameter used in the MRS is $\epsilon = 0.35 \cdot h$, where $h = d \cdot \Delta u = r \cdot \Delta v \approx 0.05$ is roughly the average grid spacing on the toroidal surfaces.[4] In this case, $n = 16$, $N_i = 4096$, $N = 65536$, C_h is 196704×196704, and A_h is 196608×196608. As in [74], a velocity that is tangent to the surface and of size 100 is imposed at every grid point, causing each torus to translate and rotate as a whole. In Fig. 8, we show the projection of the array of tori onto the xz plane as well as the streamlines drawn based on the x- and z-directional fluid velocities around them at time $t = 0$.

In GMRES(MG, LSC), the action of $\widetilde{A}_h^{-1}$ is one iteration of a three-grid method. The mid-level grid consists of 4096 points (256 points per torus), resulting from a 32×8 grid of the parameter space. In the coarsest grid, there are 1024 points in total (64 points per torus), which correspond to a 16×4 grid of the parameter space. The average grid spacings on the toroidal surfaces are thus about $H = 4h$ and $2H$ for the two grids, respectively. Within the block Gauss-Seidel smoother, we use an even coarser grid with spacing $4H$ when performing the inexact block-vector multiplication. Additional implementation details of the multigrid method can be found in Section A.4. Moreover, the initial guess of GMRES(MG, LSC) is calculated by prolonging the approximate solution obtained by applying one LSQR iteration to the representation of Eq. (57) on the coarsest grid. As in Section 7.1, the matrix-vector products involving A_h are approximated by the KIFMM, whose accuracy varies depending on whether $\widetilde{A}_h^{-1}$ or $\widetilde{S}_h^{-1}$ is being applied, in both GMRES(MG, LSC) and GMRES(BD, LSC).

In Table 7, we report the runtime, iteration count, and relative error of the final iterate of each method. Similar to what has been observed in the case of ellipsoidal swimmers, GMERS(MG, LSC) requires about 88% fewer iterations and 43% less runtime compared to GMRES(BD, LSC).

[3] We observe that the matrix-vector products arising from the application of $\widetilde{S}_h^{-1}$, where $\tilde{S}_h$ is the LSC preconditioner defined in Eq. (62) and the $(2, 2)$ block in P, do not need to be approximated accurately. Therefore, we vary the parameters of the KIFMM to calculate them more crudely but also more efficiently than we calculate the matrix-vector products arising from the application of $\widetilde{A}_h^{-1}$.

[4] Rectangular grid boxes in the parameter space result in curved rectangular grid boxes on the toroidal surfaces (see Fig. 1(d,e,f) for example), whose edge lengths vary between $\min\{(d - r) \cdot \Delta u,\ r \cdot \Delta v\}$ and $\max\{(d + r) \cdot \Delta u,\ r \cdot \Delta v\}$. Since $d \cdot \Delta u = r \cdot \Delta v$ in the fine grid, h is the average of the two.

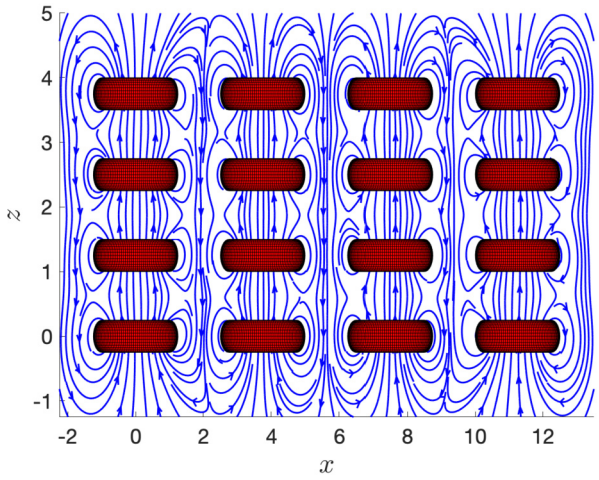

Fig. 8. Projection of the 4×4 array of toroidal swimmers onto the xz plane and streamlines drawn based on the x- and y-directional fluid velocities around them at $t = 0$.

Table 7
Comparison of GMRES(MG, LSC) and GMRES(BD, LSC) in the case of toroidal swimmers.

Solver	Runtime (s)	# of iter.	Final rel. error
GMRES(MG, LSC)	3258.72	6	$6.58 \cdot 10^{-6}$
GMRES(BD, LSC)	5750.71	50	$9.02 \cdot 10^{-6}$

8. Conclusion

The multigrid method has enjoyed a tremendous amount of success at solving discretized boundary-value problems, where the grids are Eulerian and the matrices are large and sparse. It iteratively updates the solution by removing the high-frequency terms in its error and correcting it on a coarser grid. In this study, we consider a vastly different type of problems: the interactions between structures whose relationship is determined by a kernel function that decays with distance. The structures are discretized by a Lagrangian grid. The matrix is generated by evaluating the kernel function between every pair of grid points; therefore, it is large, dense and quite challenging to work with. Nonetheless, the decay of the kernel function results in the data-sparsity of the matrix, which opens up opportunities to develop efficient solvers. One contribution of our paper is extending the multigrid method to this new scenario. More specifically, we develop new techniques for error smoothing, grid coarsening, and inter-grid communication that are more suitable for kernel functions on interacting structures. They fall into two veins depending on whether the structures are parameterized or simply represented by discrete points, which resemble the geometric multigrid and algebraic multigrid. This solver is unique in that it takes advantage of not only the data-sparsity of the matrix but also the smoothness in the geometry of the structures and quantities of interest distributed on them. Moreover, it does not require constructing the large dense matrix and is thus memory-efficient.

Another contribution of this paper is demonstrating the applicability of the proposed multigrid solver to the simulation of a fluid around dynamic microswimmers. The fluid is modeled by the MRS, a Lagrangian method that utilizes the fundamental solution to the incompressible Stokes equations. The kernel function associated with it and hence the strength of the hydrodynamic interaction between two points decays roughly as fast as the reciprocal of the distance between them. The dense matrix that characterizes the interactions between the discretized swimmers maps the forces exerted by the grid points to their velocities. Consequently, for prescribed or experimentally observed velocities, we can apply the multigrid method to uncover the underlying forces, which can in turn be used in the MRS to evaluate the entire flow. We consider large groups of microswimmers in the shapes of a variety of biological structures including bacteria, cilia, vesicles and observe that the multigrid solver outperforms a preconditioned GMRES method previously applied to the MRS matrices. In addition, we consider the special case where these structures need to maintain force and torque balance while swimming. It gives rise to an even larger matrix formed by augmenting the aforementioned dense matrix with additional sparse rows and columns. We show that the performance of its solver can be improved significantly if we apply the multigrid method to key auxiliary problems.

CRediT authorship contribution statement

Weifan Liu: Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Minghao W. Rostami:** Conceptualization, Funding acquisition, Methodology, Resources, Software, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

Liu was supported by the Fundamental Research Funds for the Central Universities (Grant No. BLX202245) and National Natural Science Foundation of China (Grant No. 12301547). Liu and Rostami were supported, in part, by the National Science Foundation (NSF) under award DMS-1818833 (PI: Rostami). Rostami was also supported, in part, by the NSF under award DMS-2146191 (CAREER).

Appendix A. Further implementation details on the multigrid method

In this section, we provide more details of the multigrid method applied to each of the four large-scale problems considered in this study: bacterial carpet (Section 6.1.2), cilia carpet (Section 6.2), free ellipsoidal swimmers (Section 7.1), and free toroidal swimmers (Section 7.2).

In all four problems, only one block Gauss-Seidel iteration is applied as the smoother.

A.1. Bacterial carpet

Recall that in Section 6.1.2, we consider a 10×10 carpet, a 20×20 carpet, and a 25×25 carpet and apply a three-grid method to Eq. (4) arising from each carpet. The three grid spacings, in the parameter domain, are $h \approx 0.014$, $H = 8h = 0.11$, and $2H = 0.22$, respectively. Accordingly, the coefficient matrices on the three grids are denoted by A_h, A_H, and A_{2H}. When calculating A_H and A_{2H} using the approximate Galerkin projection, we choose γ to be H and $2H$ in Eq. (40), respectively.

We employ two smoothers: one for A_h, and the other for A_H. For all three carpets, in the first smoother, we partition A_h into blocks of size 483×483, each of which represents the interactions between two helices. In the second smoother, we partition A_H into blocks of sizes 630×630, 1260×1260, and 315×315 for the 10×10, 20×20, and 25×25 carpets, respectively, each of which represents the interactions between two groups of ten, twenty, and five helices.

A.2. Cilia carpet

Recall that in Section 6.2, we apply a three-grid method to Eq. (4) arising from a 25×25 cilia carpet. The three grid spacings, in the parameter domain, are $h \approx 0.047$, $H = 4h \approx 0.188$, and $2H = 0.375$, respectively. Accordingly, the coefficient matrices on the three grids are again denoted by A_h, A_H, and A_{2H}. When calculating A_H and A_{2H} using the approximate Galerkin projection, we choose γ to be H and $2H$ in Eq. (40), respectively.

We need three smoothers in this case: one for A_h, one for each block A_{ii} on the diagonal of A_h, and one for A_H. In the first smoother, we partition A_h into blocks of size 9600×9600, each of which represents the interactions between two rows of cilia. This large block size makes the smoother quite effective; however, it also makes inverting A_{ii} in Eq. (47) difficult. Therefore, we apply a two-grid method (fine-grid spacing: h, coarse-grid spacing: H) to A_{ii} and another two-grid method (fine-grid spacing: H, coarse-grid spacing: $2H$) to A_H. In the smoother of the former, we partition A_{ii} into blocks of size 384×384, each of which represents the interactions between two cilia; and in the smoother of the latter, we partition A_H into blocks of size 480×480, each of which represents the interactions between two groups of five cilia.

A.3. Ellipsoidal swimmers

Recall that in Section 7.1, we apply a two-grid method to invert the $(1,1)$ block of the preconditioner (60) for Eq. (57) arising from a 6×6 array of ellipsoidal swimmers. Unlike in the other three examples, the coarse grid is obtained by the modified PMIS algorithm described in Section 4.1.2. To calculate the transfer operators, we choose $m = 6$ in Eq. (23), and $\delta = 0.02$ in Eq. (24). The fine-grid and coarse-grid matrices are denoted by A_h and A_H, respectively. When calculating A_H using the approximate Galerkin projection, we choose γ to be about 1.091 in Eq. (42).

We employ two smoothers: one for A_h, and the other for A_{ii}. In the first smoother, we partition A_h into blocks of size 3048×3048, each of which represents the interactions between two ellipsoids. We apply a two-grid method (fine-grid spacing: h, coarse-grid spacing: H) to A_{ii}; and in its smoother, we first partition the ith group of fine-grid points into six clusters using the k-means algorithm, and then partition A_{ii} into blocks representing the interactions between two clusters of points.

A.4. Toroidal swimmers

Recall that in Section 7.2, we apply a three-grid method to invert the $(1,1)$ block of the preconditioner (60) for Eq. (57) arising from a 6×6 array of toroidal swimmers. The average grid spacings over the tori are about h, $H = 4h$, and $2H$ for the three grids, respectively. Accordingly, the coefficient matrices on the three grids are again denoted by A_h, A_H, and A_{2H}. When calculating A_H and A_{2H} using the approximate Galerkin projection, we choose γ_1 to be the coarse-grid spacing in the u direction and γ_2 to be the coarse-grid spacing in the v direction; that is, $\gamma_1 \approx 0.196$, $\gamma_2 \approx 0.785$ in the case of A_H, and $\gamma_1 \approx 0.393$, $\gamma_2 \approx 1.571$ in the case of A_{2H}.

As in the case of cilia carpet, we employ three smoothers: one for A_h, one for A_{ii}, and one for A_H. In the first smoother, we partition A_h into blocks of size 12288×12288, each of which represents the interactions between two tori. We apply a two-grid method (fine-grid spacing: h, coarse-grid spacing: H) to A_{ii} and another two-grid method (fine-grid spacing: H, coarse-grid spacing: $2H$) to A_H. In the smoother of the former, we partition A_{ii} into blocks of size 768×768, each of which represents the interactions between two groups of 256 points; and in the smoother of the latter, we partition A_H into blocks of size 768×768, each of which represents the interactions between two tori.

References

[1] A.L. Buchmann, L.J. Fauci, K. Leiderman, E.M. Strawbridge, L. Zhao, Flow induced by bacterial carpets and transport of microscale loads, in: T. Jackson, A. Radunskaya (Eds.), Applications of Dynamical Systems in Biology and Medicine, in: The IMA Volumes in Mathematics and Its Applications, vol. 158, Springer New York, New York, USA, 2015, pp. 35–53.

[2] M.J. Kim, K.S. Breuer, Use of bacterial carpets to enhance mixing in microfluidic systems, J. Fluids Eng. 129 (3) (2007) 319–324, https://doi.org/10.1115/1.2427083.

[3] M.W. Rostami, W. Liu, A. Buchmann, E. Strawbridge, L. Zhao, Optimal design of bacterial carpets for fluid pumping, Fluids 7 (1) (2022) 25, https://doi.org/10.3390/fluids7010025.

[4] H. Ashraf, A. Siddiqui, M. Rana, Fallopian tube analysis of the peristaltic-ciliary flow of third grade fluid in a finite narrow tube, Chin. J. Phys. 56 (2) (2018) 605–621, https://doi.org/10.1016/j.cjph.2018.02.001.

[5] A.-I. Bunea, R. Taboryski, Recent advances in microswimmers for biomedical applications, Micromachines 11 (12) (2020) 1048, https://doi.org/10.3390/mi11121048.

[6] J. Park, C. Jin, S. Lee, J.-Y. Kim, H. Choi, Magnetically actuated degradable microrobots for actively controlled drug release and hyperthermia therapy, Adv. Healthc. Mater. 8 (16) (2019) 1900213, https://doi.org/10.1002/adhm.201900213.

[7] R. Cortez, The method of regularized Stokeslets, SIAM J. Sci. Comput. 23 (4) (2001) 1204–1225, https://doi.org/10.1137/S106482750038146X.

[8] G.C. Hsiao, W.L. Wendland, Boundary Integral Equations, 1st edition, Springer, Berlin, Heidelberg, 2008.

[9] W.-T. Ang, A Beginner's Course in Boundary Element Methods, Universal Publishers, Boca Raton, USA, 2007.

[10] N. Ueki, S. Matsunaga, I. Inouye, A. Hallmann, How 5000 independent rowers coordinate their strokes in order to row into the sunlight: phototaxis in the multicellular green alga Volvox, BMC Biol. 8 (2010) 103, https://doi.org/10.1186/1741-7007-8-103.

[11] S. Ambikasaran, E. Darve, An $O(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices, J. Sci. Comput. 57 (3) (2013) 477–501, https://doi.org/10.1007/s10915-013-9714-z.

[12] 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, https://doi.org/10.1016/j.jcp.2015.10.012.

[13] S. Chandrasekaran, M. Gu, X.S. Li, J. Xia, Some fast algorithms for hierarchically semiseparable matrices, Tech. Rep. 08–24, Group in Computational and Applied Mathematics, Department of Mathematics, University of California, Los Angeles, USA, 2008.

[14] S. Chandrasekaran, M. Gu, T. Pals, A fast ulv decomposition solver for hierarchically semiseparable representations, SIAM J. Matrix Anal. Appl. 28 (3) (2006) 603–622, https://doi.org/10.1137/S0895479803436652.

[15] P. Martinsson, V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, J. Comput. Phys. 205 (1) (2005) 1–23, https://doi.org/10.1016/j.jcp.2004.10.033.

[16] M. Ma, D. Jiao, Accuracy directly controlled fast direct solution of general $\mathcal{H}^2$-matrices and its application to solving electrodynamic volume integral equations, IEEE Trans. Microw. Theory Tech. 66 (1) (2018) 35–48, https://doi.org/10.1109/TMTT.2017.2734090.

[17] L. Grasedyck, R. Kriemann, S. Le Borne, Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems, Comput. Vis. Sci. 11 (4) (2008) 273–291, https://doi.org/10.1007/s00791-008-0098-9.

[18] B. Quaife, P. Coulier, E. Darve, An efficient preconditioner for the fast simulation of a 2D Stokes flow in porous media, Int. J. Numer. Methods Eng. 113 (4) (2018) 561–580, https://doi.org/10.1002/nme.5626.

[19] M.W. Rostami, S.D. Olson, Fast algorithms for large dense matrices with applications to biofluids, J. Comput. Phys. 394 (2019) 364–384, https://doi.org/10.1016/j.jcp.2019.05.042.

[20] D.A. Sushnikova, I.V. Oseledets, Preconditioners for hierarchical matrices based on their extended sparse form, Russ. J. Numer. Anal. Math. Model. 31 (1) (2016) 29–40, https://doi.org/10.1515/rnam-2016-0003.

[21] J. Xia, Z. Xin, Effective and robust preconditioning of general SPD matrices via structured incomplete factorization, SIAM J. Matrix Anal. Appl. 38 (4) (2017) 1298–1322, https://doi.org/10.1137/17M1124152.

[22] Z. Xin, J. Xia, S. Cauley, V. Balakrishnan, Effectiveness and robustness revisited for a preconditioning technique based on structured incomplete factorization, Numer. Linear Algebra Appl. 27 (3) (2020) e2294, https://doi.org/10.1002/nla.2294.

[23] H. Al Daas, L. Grigori, P. Jolivet, P.-H. Tournier, A multilevel Schwarz preconditioner based on a hierarchy of robust coarse spaces, SIAM J. Sci. Comput. 43 (3) (2021) A1907–A1928, https://doi.org/10.1137/19M1266964.

[24] K. Chen, An analysis of sparse approximate inverse preconditioners for boundary integral equations, SIAM J. Matrix Anal. Appl. 22 (4) (2001) 1058–1078, https://doi.org/10.1137/S0895479898348040.

[25] N.I.M. Gould, J.A. Scott, On approximate-inverse preconditioners, Tech. Rep. RAL 95-026, Computing and Information Systems Department, Atlas Centre, Rutherford Appleton Laboratory, Oxfordshire, England, 1995.
[26] R. Fedorenko, The speed of convergence of one iterative process, USSR Comput. Math. Math. Phys. 4 (3) (1964) 227–235, https://doi.org/10.1016/0041-5553(64)90253-8.
[27] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Math. Comput. 31 (138) (1977) 333–390, https://doi.org/10.2307/2006422.
[28] C.C. Douglas, Multi-grid algorithms with applications to elliptic boundary value problems, SIAM J. Numer. Anal. 21 (2) (1984) 236–254, https://doi.org/10.1137/0721017.
[29] A. Brandt, S. McCormick, J. Ruge, Algebraic multigrid (AMG) for sparse matrix equations, in: D.J. Evans (Ed.), Sparsity and Its Applications, Cambridge University Press, Cambridge, UK, 1985, pp. 257–284.
[30] A. Brandt, Algebraic multigrid theory: the symmetric case, Appl. Math. Comput. 19 (1) (1986) 23–56, https://doi.org/10.1016/0096-3003(86)90095-0.
[31] T.F. Chan, J. Xu, L. Zikatanov, An agglomeration multigrid method for unstructured grids, in: J. Mandel, C. Farhat, X.-C. Cai (Eds.), Domain Decomposition Methods 10, in: Contemporary Mathematics, vol. 218, American Mathematical Society, Providence, USA, 1998, pp. 67–81.
[32] D.J. Mavriplis, V. Venkatakrishnan, A 3D agglomeration multigrid solver for the Reynolds–averaged Navier–Stokes equations on unstructured meshes, Int. J. Numer. Methods Fluids 23 (6) (1996) 527–544, https://doi.org/10.1002/(SICI)1097-0363(19960930)23:6<527::AID-FLD429>3.0.CO;2-Z.
[33] P. Vaněk, J. Mandel, M. Brezina, Algebraic multigrid on unstructured meshes, Tech. Rep. 34, Center for Computational Mathematics, University of Colorado at Denver, Denver, USA, 1994.
[34] J.M. Weiss, J.P. Maruszewski, W.A. Smith, Implicit solution of preconditioned Navier-Stokes equations using algebraic multigrid, AIAA J. 37 (1) (1999) 29–36, https://doi.org/10.2514/2.689.
[35] R.D. Guy, B. Philip, B.E. Griffith, Geometric multigrid for an implicit-time immersed boundary method, Adv. Comput. Math. 41 (3) (2015) 635–662, https://doi.org/10.1007/s10444-014-9380-1.
[36] V. John, L. Tobiska, Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier-Stokes equations, Int. J. Numer. Methods Fluids 33 (4) (2000) 453–473, https://doi.org/10.1002/1097-0363(20000630)33:4<453::AID-FLD15>3.0.CO;2-0.
[37] E. Tohidi, K. Muzhinji, S. Shateyi, S.S. Motsa, The mixed finite element multigrid method for Stokes equations, Sci. World J. 2015 (2015) 460421, https://doi.org/10.1155/2015/460421.
[38] M. Wang, L. Chen, Multigrid methods for the Stokes equations using distributive Gauss-Seidel relaxations based on the least squares commutator, J. Sci. Comput. 56 (2) (2013) 409–431, https://doi.org/10.1007/s10915-013-9684-1.
[39] J.H. Bramble, Z. Leyk, J.E. Pasciak, The analysis of multigrid algorithms for pseudodifferential operators of order minus one, Math. Comput. 63 (208) (1994) 461–478, https://doi.org/10.2307/2153279.
[40] U. Langer, D. Pusch, S. Reitzinger, Efficient preconditioners for boundary element matrices based on grey-box algebraic multigrid methods, Int. J. Numer. Methods Eng. 58 (13) (2003) 1937–1953, https://doi.org/10.1002/nme.839.
[41] G. Of, An efficient algebraic multigrid preconditioner for a fast multipole boundary element method, Computing 82 (2) (2008) 139–155, https://doi.org/10.1007/s00607-008-0002-y.
[42] 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, https://doi.org/10.1063/1.1830486.
[43] J. Ainley, S. Durkin, R. Embid, P. Boindala, R. Cortez, The method of images for regularized Stokeslets, J. Comput. Phys. 227 (9) (2008) 4600–4616, https://doi.org/10.1016/j.jcp.2008.01.032.
[44] J.R. Blake, A note on the image system for a stokeslet in a no-slip boundary, Math. Proc. Camb. Philos. Soc. 70 (2) (1971) 303–310, https://doi.org/10.1017/S0305004100049902.
[45] J.S. Butler, Improving coarsening and interpolation for algebraic multigrid, Master's thesis, University of Waterloo, Waterloo, Canada, 2006.
[46] H. De Sterck, U.M. Yang, J.J. Heys, Reducing complexity in parallel algebraic multigrid preconditioners, SIAM J. Matrix Anal. Appl. 27 (4) (2006) 1019–1039, https://doi.org/10.1137/040615729.
[47] R. Schaback, H. Wendland, Characterization and construction of radial basis functions, in: N. Dyn, D. Leviatan, D. Levin, A. Pinkus (Eds.), Multivariate Approximation and Applications, Cambridge University Press, Cambridge, UK, 2001, pp. 1–24.
[48] E. Reich, On the convergence of the classical iterative method of solving linear simultaneous equations, Ann. Math. Stat. 20 (3) (1949) 448–451.
[49] A.M. Ostrowski, On the linear iteration procedures for symmetric matrices, Rend. Mat. Appl. 14 (1954) 140–163.
[50] G. Golub, C. Van Loan, Matrix Computations, 4th edition, Johns Hopkins University Press, Baltimore, USA, 2013.
[51] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd edition, SIAM, Philadelphia, USA, 2003.
[52] H. Courtecuisse, J. Allard, Parallel dense Gauss-Seidel algorithm on many-core processors, in: 2009 11th IEEE International Conference on High Performance Computing and Communications, 2009, pp. 139–147.
[53] A distributed memory parallel Gauss–Seidel algorithm for linear algebraic systems, Comput. Math. Appl. 57 (8) (2009) 1369–1376.
[54] A. Ahmadi, F. Manganiello, A. Khademi, M.C. Smith, A parallel Jacobi-embedded Gauss-Seidel method, IEEE Trans. Parallel Distrib. Syst. 32 (6) (2021) 1452–1464.
[55] N. Darnton, L. Turner, K. Breuer, H.C. Berg, Moving fluid with bacterial carpets, Biophys. J. 86 (3) (2004) 1863–1870, https://doi.org/10.1016/S0006-3495(04)74253-8.
[56] Y.-T. Hsiao, J.-H. Wang, K.-T. Wu, J. Tsai, C.-H. Chang, W.-Y. Woon, Collective flow dynamics across a bacterial carpet: understanding the forces generated, Appl. Phys. Lett. 105 (20) (2014) 203702, https://doi.org/10.1063/1.4902111.
[57] A. Buchmann, L.J. Fauci, K. Leiderman, E. Strawbridge, L. Zhao, Mixing and pumping by pairs of helices in a viscous fluid, Phys. Rev. E 97 (2018) 023101, https://doi.org/10.1103/PhysRevE.97.023101.
[58] L. Ying, G. Biros, D. Zorin, A kernel-independent adaptive fast multipole algorithm in two and three dimensions, J. Comput. Phys. 196 (2) (2004) 591–626, https://doi.org/10.1016/j.jcp.2003.11.021.
[59] L. Ying, A kernel independent fast multipole algorithm for radial basis functions, J. Comput. Phys. 213 (2) (2006) 451–457, https://doi.org/10.1016/j.jcp.2005.09.010.
[60] Y. Ding, J.C. Nawroth, M.J. McFall-Ngai, E. Kanso, Mixing and transport by ciliary carpets: a numerical study, J. Fluid Mech. 743 (2014) 124–140, https://doi.org/10.1017/jfm.2014.36.
[61] Y. Ding, E. Kanso, Selective particle capture by asynchronously beating cilia, Phys. Fluids 27 (12) (2015) 121902, https://doi.org/10.1063/1.4938558.
[62] D.J. Smith, A boundary element regularized Stokeslet method applied to cilia- and flagella-driven flow, Proc. R. Soc. A, Math. Phys. Eng. Sci. 465 (2112) (2009) 3605–3626, https://doi.org/10.1098/rspa.2009.0295.
[63] G. Fulford, J. Blake, Muco-ciliary transport in the lung, J. Theor. Biol. 121 (4) (1986) 381–402, https://doi.org/10.1016/S0022-5193(86)80098-4.
[64] J. Blake, A model for the micro-structure in ciliated organisms, J. Fluid Mech. 55 (1) (1972) 1–23, https://doi.org/10.1017/S0022112072001612.
[65] M. Benzi, G.H. Golub, J. Liesen, Numerical solution of saddle point problems, Acta Numer. 14 (2005) 1–137, https://doi.org/10.1017/S0962492904000212.
[66] S.J. Benbow, Solving generalized least-squares problems with LSQR, SIAM J. Matrix Anal. Appl. 21 (1) (1999) 166–177, https://doi.org/10.1137/S0895479897321830.
[67] C.C. Paige, M.A. Saunders, LSQR: an algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Softw. 8 (1) (1982) 43–71, https://doi.org/10.1145/355984.355989.

[68] H. Elman, D. Silvester, A. Wathen, Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics, 2nd edition, Oxford University Press, 2014.
[69] S.K. Veerapaneni, A. Rahimian, G. Biros, D. Zorin, A fast algorithm for simulating vesicle flows in three dimensions, J. Comput. Phys. 230 (14) (2011) 5610–5634, https://doi.org/10.1016/j.jcp.2011.03.045.
[70] T. Omori, T. Ishikawa, Y. Imai, T. Yamaguchi, Membrane tension of red blood cells pairwisely interacting in simple shear flow, J. Biomech. 46 (3) (2013) 548–553, https://doi.org/10.1016/j.jbiomech.2012.09.017.
[71] R.D. Nair, S.J. Thomas, R.D. Loft, A discontinuous Galerkin global shallow water model, Mon. Weather Rev. 133 (4) (2005) 876–888, https://doi.org/10.1175/MWR2903.1.
[72] G.I. Taylor, The action of waving cylindrical tails in propelling microscopic organisms, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 211 (1105) (1952) 225–239, https://doi.org/10.1098/rspa.1952.0035.
[73] S. Hess, L. Eme, A.J. Roger, A.G. Simpson, A natural toroidal microswimmer with a rotary eukaryotic flagellum, Nat. Microbiol. 4 (10) (2019) 1620–1626, https://doi.org/10.1038/s41564-019-0478-6.
[74] J. Huang, L. Fauci, Interaction of toroidal swimmers in Stokes flow, Phys. Rev. E 95 (4) (2017) 043102, https://doi.org/10.1103/PhysRevE.95.043102.