A fast direct solver for integral equations on locally refined boundary discretizations and its application to multiphase flow simulations

Yabin Zhang[1] · Adrianna Gillman[2] · Shravan Veerapaneni[1]

Abstract
In transient simulations of particulate Stokes flow, to accurately capture the inter-action between the constituent particles and the confining wall, the discretization of the wall often needs to be locally refined in the region approached by the parti-cles. Consequently, standard fast direct solvers lose their efficiency since the linear system changes at each time step. This manuscript presents a new computational approach that avoids this issue by pre-constructing a fast direct solver for the wall ahead of time, computing a low-rank factorization to capture the changes due to the refinement, and solving the problem on the refined discretization via a Woodbury formula. Numerical results illustrate the efficiency of the solver in accelerating par-ticulate Stokes simulations.

Communicated by: Michael O'Neil

This article belongs to the Topical Collection: *Advances in Computational Integral Equations*

Guest Editors: Stephanie Chaillat, Adrianna Gillman, Per-Gunnar Martinsson, Michael O'Neil, Mary-Catherine Kropinski, Timo Betcke, Alex Barnett

✉ Yabin Zhang
   yabinz@umich.edu

[1] Department of Mathematics, University of Michigan, Ann Arbor, USA

[2] Department of Applied Mathematics, University of Colorado, Boulder, USA

1 Introduction

A common computational task that arises in simulations of particulate Stokes flow is evaluating the hydrodynamic interaction of small moving geometries, such as drops, bacteria, or biological cells, with large static structures, such as microfluidic chips, vascular walls, or channel walls. Boundary integral equation (BIE) methods, solved via iterative solvers accelerated by fast summation methods, are often used in practice for such systems as they avoid meshing the volume including the cumbersome task of volume re-meshing in transient simulations. In [1], a fast direct solver was proposed, which further reduces the cost of simulations by precomputing the compressed inverse of the BIE operator corresponding to the large static structures, which can be applied in linear time. This can be extremely useful in practice since most applications require a large number of time steps to observe the physics of interest, e.g., alignment of vesicles in a periodic channel [2], pattern formation in suspensions of active particles [3, 4] and cell sorting [5].

However, when the suspended particles evolve in close proximity to the confining walls, the discretization of the walls must be locally refined to resolve the hydrodynamic interaction [6]. In this situation direct solvers are less attractive since the new inverse operator needs to be evaluated continuously. We present a fast algorithm that avoids building a new inverse operator from scratch for each time step by precomputing an inverse operator corresponding to a reference mesh and rapidly updating it whenever the boundary discretization is locally refined (or coarsened). This work is an extension of Zhang-Gillman [7, 8], where Laplace BIEs on locally perturbed geometries were considered. The central idea is that the discretized BIE on the walls can be written as an extended version of the linear system for the original geometry, and a fast direct solver on the original geometry can be reused to reduce the computational burden of solving the problems on the refined discretization.

Related work At a high-level, fast direct solvers exploit the fact that the off-diagonal blocks of the discretized system are low rank. In the context of integral equations, some of them include the *Hierarchically Block Separable (HBS)* [9, 10], the *Hierarchically Semi-Separable (HSS)* [11, 12], the *Hierarchical Interpolative Factorization (HIF)* [13] and the $\mathscr{H}$ or $\mathscr{H}^2$- matrix methods [14]. The techniques developed in [7, 8] for the extended linear system (ELS), designed for problems with locally perturbed geometries, can be coupled with any of the above direct solver approaches. In this work, we employ a particular fast direct solver based on HBS matrix representation and inverse presented in [10]. For the rest of the manuscript, when an HBS representation or inverse is built for a discretized boundary integral equation, it refers to the particular compression and inverse approximation given in [10]. Other fast direct solvers for BIEs can be used in place of the HBS solver, and the results will be comparable.

An alternative to using the ELS is to update the hierarchical representation of the discretized integral operator directly. Existing techniques in [15, 16] update the HIF of the system with a cost that is bounded above by the cost of building a HIF of the

perturbed or refined problem from scratch. For problems that do not require a large number of discretization points, directly updating HIF is expected to be cheaper than building a new one from scratch. This idea is first investigated in [15], and a parallel implementation for Stokes BIEs on multiply-connected domains is presented in [16]. Being direct solvers, these techniques are advantageous when a large number of solves are required for each new geometry. Generalizing the idea to other standard fast direct solvers, such as those based on HBS or HSS matrix, requires knowledge of the particular compression techniques used in the chosen fast direct solver and is non-trivial.

Several previous works employ fast direct solvers as preconditioners for the linear systems that result from the discretization of integral equations and differential equations [17–21]. Most of them build a low-accuracy direct solver for the linear system and apply the forward operator via a fast matrix multiplication technique. The improvement of the convergence of the iterative solver is directly related to the accuracy at which the preconditioner was built. Section 4 explores the left preconditioner option and how the accuracy of the direct solver impacts the quality of the preconditioner. In general, whether a high accuracy preconditioner is worth developing depends on the cost of the construction and how many solves are desired.

Contributions Motivated by the fluids applications mentioned above, we apply the solution technique given in [7, 8] to Stokes flow problems defined on complex geometries, some of which are adapted from real application geometry data. The linear system associated with the discretization of an integral equation for Stokes flow has a physical nullspace corresponding to the pressure being unique up to a constant. Fast direct solvers like HBS are sensitive to the existence of such non-trivial nullspace due to the fact that matrices of smaller sizes are inverted in the hierarchical structure and the singularity will immediately cause trouble. The nullspace can be removed via an analytic technique, but the resulting linear system can have a high condition number due to the physics and/or complexity of the geometry. We observe from numerical tests that the linear system that needs to be solved for Stokes problems has a condition number that is *at least* the square of the linear system for a Laplace problem on the same geometry. The high condition number of the system results in the small matrices inverted within the hierarchical structure of a fast direct solver being ill-conditioned, resulting in loss of accuracy that is not often seen in Laplace problems. This is even more cumbersome when local refinement is added to the original discretization. The solution technique given in [7, 8] requires inverting a matrix whose conditioning may be worse than the original discretized BIE. Since the condition number of the linear system for Stokes problems is often already relatively high compared to Laplace problems, the compression required in these solution methods should be done carefully. This manuscript reviews the technique, particularly the low-rank approximation construction for the update matrix in [7], and focuses on some important details for applying it to more complicated problems efficiently and stably. Since the direct solution approach may not be able to achieve the desired accuracy due to the conditioning for certain problems, we also consider using the local refinement fast direct solver as a preconditioner for the ELS. When

coupled with a fast matrix multiplication technique for applying the ELS, the resulting solution technique converges in a constant number of iterations *independent* of the number of discretization points (as long as the geometric features are sufficiently resolved).

Limitations This manuscript only considers two-dimensional problems even though the ideas introduced here generalize to higher dimensions. Additional work is needed in integrating other computational machinery, e.g., 3D surface quadrature methods. In dense suspension flows, the particle-wall near interactions happen over long length- and time-scales. Clearly, the solver developed here is not applicable to this setting since the wall geometry needs to be globally refined, in which case the approach prescribed in [1] is better suited. Lastly, when the particles approach arbitrarily close to the walls, close evaluation schemes (e.g., [6, 22]) are required to improve the accuracy of interaction force computation. Incorporating these methods and testing the solver is left to future work.

Outline The manuscript begins by reviewing boundary integral formulations for Stokes problems and a technique for discretizing the resulting integral equations in Section 2. Next, the ELS for locally refined discretization and the corresponding direct solver are presented in Section 3. The proposed preconditioner for the ELS is presented in Section 4. Next Section 5 illustrates the performance of the presented solution techniques. Finally, Section 6 closes the manuscript with a summary and concluding remarks.

2 Boundary integral formulation

This manuscript considers integral equation techniques for solving both interior and exterior Stokes flow problems. The indirect integral equation formulation is employed, wherein, the solution can be cast as a convolution over the boundary Γ of a kernel with an unknown boundary charge density. For example, the velocity $\mathbf{u}$ can be represented by

$$\mathbf{u}(\mathbf{x}) = \int_{\Gamma} \mathcal{K}(\mathbf{x}, \mathbf{y})\boldsymbol{\tau}(\mathbf{y})ds_{\mathbf{y}} = (\mathcal{K}_{\Gamma}\boldsymbol{\tau})(\mathbf{x}),$$

where $\mathcal{K}$ denotes a kernel related to the fundamental solution of the Stokes equations and $\boldsymbol{\tau}$ denotes the unknown charge density. The kernel is chosen based on the problem under consideration. One option is to represent the solution via the single layer integral operator denoted by $\mathbf{u}(\mathbf{x}) = (\mathcal{S}_{\Gamma}\tau)(\mathbf{x})$, where $\mathcal{S}$ denotes the Stokes single layer kernel (Stokeslet) defined in its tensor components by

$$S_{ij}(\mathbf{x}, \mathbf{y}) = \frac{1}{4\pi\mu}\left(\delta_{ij}\log\left(\frac{1}{r}\right) + \frac{r_i r_j}{r^2}\right), \quad i, j = 1, 2, \tag{1}$$

where $\mathbf{r} = \mathbf{x} - \mathbf{y}$, $r = \|\mathbf{r}\|$ and δ_{ij} is the Kronecker delta.

Another option is to use a double layer integral operator $\mathbf{u}(\mathbf{x}) = (\mathcal{D}_\Gamma \tau)$ to represent the velocity. The tensor components of the double layer kernel $\mathcal{D}$ are

$$D_{ij}(\mathbf{x}, \mathbf{y}) = \frac{1}{\pi} \frac{r_i r_j}{r^2} \frac{\mathbf{r} \cdot \mathbf{n_y}}{r^2}, \quad i, j = 1, 2$$

where $\mathbf{n_y}$ is the surface normal vector at the point $\mathbf{y} \in \Gamma$.

Likewise, the pressure can be represented via an integral operator. It should be chosen to match the representation of the velocity. For example, if the velocity is represented with the single layer integral operator, then the pressure is given by

$$p(\mathbf{x}) = \int_\Gamma \mathcal{Q}(\mathbf{x}, \mathbf{y}) \tau(\mathbf{y}) ds_\mathbf{y}$$

where

$$Q_j(\mathbf{x}, \mathbf{y}) = \frac{1}{2\pi} \frac{r_j}{r^2}, \quad j = 1, 2$$

and $\boldsymbol{\tau}$ is the same boundary charge density as in the definition of the velocity. If the velocity is represented via the double layer integral operator, then the pressure is given by

$$p(\mathbf{x}) = \int_\Gamma \mathcal{P}(\mathbf{x}, \mathbf{y}) \tau(\mathbf{y}) ds_\mathbf{y}$$

where

$$P_j(\mathbf{x}, \mathbf{y}) = \frac{\mu}{\pi} \left(-\frac{\mathbf{n}_{j,\mathbf{y}}}{r^2} + 2\frac{r_j}{r^4} \mathbf{r} \cdot \mathbf{n_y} \right), \quad j = 1, 2,$$

$\mathbf{n}_{j,\mathbf{y}}$ denotes the jth component of the surface normal vector $\mathbf{n_y}$, and $\boldsymbol{\tau}$ is the boundary charge density as defined in the velocity.

2.1 Interior Stokes problem

Consider the incompressible Stokes equation inside a geometry Ω_{in} given by

$$\begin{aligned} -\mu \Delta \mathbf{u}(\mathbf{x}) + \nabla p(\mathbf{x}) &= \mathbf{0}, & \text{for } \mathbf{x} \in \Omega_{\text{in}} \\ \nabla \cdot \mathbf{u}(\mathbf{x}) &= 0, & \text{for } \mathbf{x} \in \Omega_{\text{in}} \\ \mathbf{u}(\mathbf{x}) &= \mathbf{g}(\mathbf{x}), & \text{for } \mathbf{x} \in \Gamma = \partial \Omega_{\text{in}}, \end{aligned} \tag{2}$$

where μ denotes the viscosity, $\mathbf{u}$ denotes the velocity, $\mathbf{g}(\mathbf{x})$ is a vector valued function denoting the boundary data, and $p(\mathbf{x})$ is a scalar valued function denoting the pressure. Figure 1(a) illustrates a sample geometry. The Dirichlet boundary data needs to satisfy the following consistency condition:

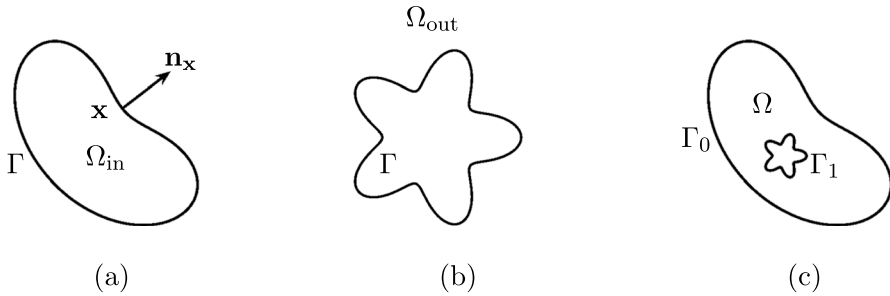$$\int_\Gamma \mathbf{g}(\mathbf{x}) \cdot \mathbf{n_x} ds_\mathbf{x} = 0 \tag{3}$$

Fig. 1 (**a**) A sample geometry for a purely interior BVP where the domain Ω_{in} is the interior of the boundary $\Gamma = \partial\Omega_{in}$, (**b**) a sample geometry for a purely exterior BVP where the domain Ω_{out} is the exterior of the boundary $\Gamma = \partial\Omega_{out}$, and (**c**) a sample geometry for an interior-exterior BVP where the domain Ω is the interior of the outer boundary Γ_0 but exterior of the inner boundary Γ_1

where $\mathbf{n_x}$ denotes the outward pointing normal vector at $\mathbf{x} \in \Gamma$.

Representing the velocity via the double layer kernel

$$\mathbf{u}(\mathbf{x}) = (\mathcal{D}_\Gamma \tau)(\mathbf{x})$$

results in having to solve the following boundary integral equation

$$-\frac{1}{2}\tau(\mathbf{x}) + (\mathcal{D}\tau)(\mathbf{x}) = \mathbf{g}(\mathbf{x}) \tag{4}$$

for the unknown density τ [23]. Discretization of the BIE (4) via the Nyström method results in having to solve a dense linear system of the form

$$\left(-\frac{1}{2}\mathbf{I} + \mathbf{D}\right)\tau = \mathbf{g} \tag{5}$$

where $\mathbf{D}$ denotes the matrix that results from the discretization of the double layer integral operator, $\mathbf{g}$ denotes a vector with entries given by the evaluation of $\mathbf{g}(\mathbf{x})$ at the quadrature nodes, and the vector τ denotes the vector of the unknown density values at the discretization points.

Remark 2.1 The solution to (2) is unique up to a constant which results in the linear system (5) having a rank-1 nullspace. This nullspace can be removed by adding the following integral operator $\mathcal{N}$

$$(\mathcal{N}\tau)(\mathbf{x}) = \mathbf{n_x} \int_\Gamma \tau(\mathbf{y}) \cdot \mathbf{n_y} ds_{\mathbf{y}} \tag{6}$$

to the left-hand side of the integral equation (5). [24] presents a Nyström technique for discretizing the integral operator (6). The linear system that results from the discretization using the double layer representation with the null space correction is

$$-\frac{1}{2}\tau + (\mathbf{D} + \mathbf{N})\tau = \mathbf{g} \tag{7}$$

where $\mathbf{N}$ is the matrix that results from the discretization of (6).

2.2 Exterior Stokes problem

Exterior incompressible Stokes problems are also considered in this paper. By an exterior problem, we mean that the velocity is sought in the domain Ω_{out} defined as the plane minus the interior of a curve Γ as shown in Fig. 1(b). By using a combined field representation for the velocity

$$\mathbf{u}(\mathbf{x}) = (\mathcal{D}_\Gamma \boldsymbol{\tau})(\mathbf{x}) + (\mathcal{S}_\Gamma \boldsymbol{\tau})(\mathbf{x}) = [\Leftarrow\mathcal{D}\Downarrow\mathcal{S}\Rightarrow_\Gamma \boldsymbol{\tau}](\mathbf{x}),$$

one is left with solving a second kind integral equation

$$\frac{1}{2}\boldsymbol{\tau} + [\Leftarrow\mathcal{D}\Downarrow\mathcal{S}\Rightarrow_\Gamma \boldsymbol{\tau}] = \mathbf{g}. \tag{8}$$

The linear system that results from discretizing this integral equation is full rank.

Remark 2.2 We also consider interior-exterior problems as shown in Fig. 1(c), where the boundary Γ is composed of an enclosing boundary curve Γ_0 and one or more holes with boundary Γ_1 inside the enclosed region. The domain Ω is defined as the set that is interior to Γ_0 but exterior to Γ_1; i.e., $\Gamma = \Gamma_0 \cup \Gamma_1$.

3 An extended linear system and direct solver for boundary value problems with locally refined discretization

The efficient solver presented in this paper utilizes techniques previously developed in [7, 8], which are originally designed to handle BIEs defined on locally perturbed geometries. A geometry is said to be *locally perturbed* if small parts of the boundary are modified from a previous BIE solve while the remainder of the boundary remains the same. We exploit the fact these techniques can be applied to handle local refinement of a discretization. For Stokes problems, the original fast solver techniques needed to be modified in order to handle the higher condition number associated with these problems. This section reviews the techniques from [7, 8] and presents the new version needed for Stokes problems. Section 3.1 defines a problem with locally refined discretization and introduces notation. Section 3.2 then presents the ELS and the efficient technique of solving that linear system using a solver built for the original discretization. Section 3.3 introduces compression ideas for the update matrix that capture changes in discretization.

The fast direct solver presented in this section scales linearly with respect to the number of points on the original discretization. The solver can also scale linearly with respect to the number of points that are added in the region of refinement when a linear scaling inversion scheme is used to invert the discretized boundary integral operator on the refined part of the boundary. If the number of points added is not large (i.e., less than one thousand), dense linear algebra is recommended for

handling the refined region since fast inversion algorithms such as HBS inversion [9, 10] tend to be slower than dense linear algebra for small matrices.

3.1 Model problem with locally refined discretization

Consider the interior BVP defined by equation (2) on the geometry Ω_{in} illustrated in Fig. 1(a). As an example, let the boundary originally be discretized with ten 16-point Gaussian panels. Then one panel Γ_r is chosen to be refined into four panels as illustrated in Fig. 2. Figure 2(a) and (b) illustrate the pre- and post-refinement respectively. Let $\mathcal{I}_k$, $\mathcal{I}_c$, and $\mathcal{I}_p$ ("p" for "plus") denote the discretization points that are kept, deleted, and added for the refinement. Thus $\mathcal{I}_o = \mathcal{I}_k \cup \mathcal{I}_c$ denotes the collection of points in the original discretization and $\mathcal{I}_n = \mathcal{I}_k \cup \mathcal{I}_p$ denotes the collection of discretization points on the boundary after refinement.

The linear system that results from the discretization of (7) for the original and new panel distributions can be ordered utilizing discretization point subscript notation. Let $\mathbf{A}$ denote the discretized integral equation (7) on the boundary; i.e., $\mathbf{A} = -\frac{1}{2}\mathbf{I} + \mathbf{D} + \mathbf{N}$. The discretized integral equation using the original panel distribution (corresponding to I_o) can be written as follows:

$$\mathbf{A}_{o,o}\boldsymbol{\sigma}_o = \begin{bmatrix} \mathbf{A}_{k,k} & \mathbf{A}_{k,c} \\ \mathbf{A}_{c,k} & \mathbf{A}_{c,c} \end{bmatrix} \begin{pmatrix} \boldsymbol{\sigma}_k \\ \boldsymbol{\sigma}_c \end{pmatrix} = \begin{pmatrix} \mathbf{g}_k \\ \mathbf{g}_c \end{pmatrix} = \mathbf{g}_o. \tag{9}$$

Likewise the linear system resulting from the refined discretization of the boundary integral equation can be ordered as follows
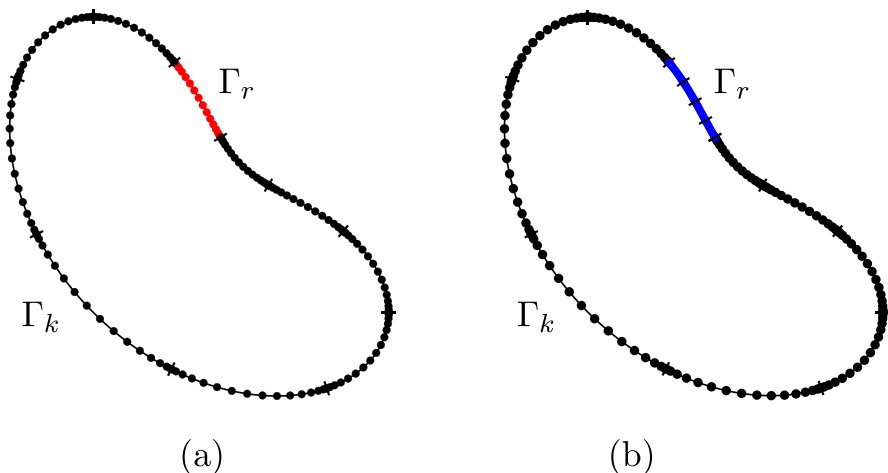

(a)   (b)

Fig. 2 (**a**) The original Gaussian panel discretization of the geometry in Fig. 1(a). The discretization contains ten 16-point Gaussian panels uniformly distributed in parameterization space, and the panel in red is chosen to be refined. (**b**) A locally refined discretization which replaced the single red panel in panel (**a**) with four blue panels. The part of the boundary curve that is refined is denoted by Γ_r and the rest is denoted by Γ_k

$$\mathbf{A}_{n,n}\boldsymbol{\tau}_n = \begin{bmatrix} \mathbf{A}_{k,k} & \mathbf{A}_{k,p} \\ \mathbf{A}_{p,k} & \mathbf{A}_{p,p} \end{bmatrix} \begin{pmatrix} \boldsymbol{\tau}_k \\ \boldsymbol{\tau}_p \end{pmatrix} = \begin{pmatrix} \mathbf{g}_k \\ \mathbf{g}_p \end{pmatrix} = \mathbf{g}_n \tag{10}$$

In equations (9) and (10), the subscript notation refers to the submatrices of $\mathbf{A}$ on the respective geometries corresponding to different boundary interactions. For example, $\mathbf{A}_{k,k}$ denotes the submatrix of $\mathbf{A}$ corresponding to the interaction of the points in $\mathcal{I}_k$ with themselves ($A(\mathcal{I}_k, \mathcal{I}_k)$ in Matlab notation) and $\mathbf{A}_{k,p}$ denotes the submatrix of $\mathbf{A}$ corresponding to the interaction of the points in $\mathcal{I}_k$ with the points in $\mathcal{I}_p$.

Remark 3.1 While the techniques in this section were presented for the interior problem presented in equation (2), the techniques apply directly to exterior problems as well.

3.2 The extended linear system and direct solver

As an alternative to casting the problem solely on the "new" discretization, an ELS that is equivalent to equation (10) can be considered. In this paper, we use the ELS from [8]. The ELS takes the form

$$\mathbf{A}_{\mathrm{ext}}\boldsymbol{\tau}_{\mathrm{ext}} = \begin{bmatrix} \mathbf{A}_{k,k} & \mathbf{0} & \mathbf{A}_{k,p} \\ \mathbf{A}_{c,k} & \mathbf{A}_{c,c} & \mathbf{0} \\ \mathbf{A}_{p,k} & \mathbf{0} & \mathbf{A}_{p,p} \end{bmatrix} \begin{pmatrix} \boldsymbol{\tau}_k \\ \boldsymbol{\tau}_c^{\mathrm{dum}} \\ \boldsymbol{\tau}_p \end{pmatrix} = \begin{pmatrix} \mathbf{g}_k \\ \mathbf{0} \\ \mathbf{g}_p \end{pmatrix} = \mathbf{g}_{\mathrm{ext}} \tag{11}$$

where $\boldsymbol{\tau}_k$ and $\boldsymbol{\tau}_p$ are the unknown boundary densities evaluated at points in $\mathcal{I}_n$ and $\boldsymbol{\tau}_c^{\mathrm{dum}}$ is a dummy boundary density at the points in $\mathcal{I}_c$ that is not used to evaluate the solution in the domain. This linear system can be written as $\mathbf{A}_{\mathrm{ext}} = \tilde{\mathbf{A}} + \mathbf{Q}$ where

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{o,o} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{p,p} \end{bmatrix} \text{ and } \mathbf{Q} = \begin{bmatrix} \mathbf{0} & -\mathbf{A}_{k,c} & \mathbf{A}_{k,p} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{p,k} & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

The matrix $\tilde{\mathbf{A}}$ is full rank and block-diagonal with the first block $\mathbf{A}_{o,o}$ equal to the integral equation operator discretized with the original mesh. Thus if the inverse of $\mathbf{A}_{o,o}$ has been precomputed (directly or via a fast direct solver), the cost of inverting $\tilde{\mathbf{A}}$ is the cost of the inverting $\mathbf{A}_{p,p}$ which is small in the problems under consideration. The update matrix $\mathbf{Q}$ is a block sparse matrix consisting of only three non-zero sub-blocks. Since these non-zero blocks of $\mathbf{Q}$ correspond to non-self interactions, they are low rank; i.e., $\mathbf{Q}$ is low rank. Let $\mathbf{Q} = \mathbf{LR}$ denote the low-rank factorization of $\mathbf{Q}$.

The advantage of writing the linear system in the extended form (11) and writing it as the sum of a block diagonal matrix with a low-rank matrix is that the inverse can be approximated via a Woodbury formula

$$\boldsymbol{\tau}_{\mathrm{ext}} = \left(\tilde{\mathbf{A}} + \mathbf{Q}\right)^{-1}\mathbf{g}_{\mathrm{ext}} \approx \left(\tilde{\mathbf{A}} + \mathbf{LR}\right)^{-1}\mathbf{g}_{\mathrm{ext}} \approx \tilde{\mathbf{A}}^{-1}\mathbf{g}_{\mathrm{ext}} - \tilde{\mathbf{A}}^{-1}\mathbf{L}\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right)^{-1}\mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{g}_{\mathrm{ext}}. \tag{12}$$

This inverse can be applied rapidly to vectors by exploiting the block structure of the matrices. The only matrix that needs to be inverted in the application of (12) is $\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right)$. This matrix is of size $k = k_{kc} + k_{kp} + k_{pk}$ where k_{kc}, k_{kp}, and k_{pk} denote the ϵ −ranks (as defined below) of the low-rank approximations of $\mathbf{A}_{k,c}$, $\mathbf{A}_{k,p}$, and $\mathbf{A}_{p,k}$ respectively.

Definition 3.1 Given a tolerance ϵ ans a $m \times n$ matrix $\mathbf{W}$ (assuming $m < n$). Let $\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{W}$ be the singular value decomposition (SVD) of $\mathbf{W}$. The ϵ − rank k_ϵ is defined to be the smallest positive integer such that the corresponding truncated SVD approximation $\mathbf{U}_\epsilon\Sigma_\epsilon\mathbf{V}_\epsilon^T$ where Σ_ϵ contains only the first k_ϵ diagonal entries of Σ satisfies

$$\|\mathbf{U}_\epsilon\Sigma_\epsilon\mathbf{V}_\epsilon^T - \mathbf{W}\| \leq \epsilon\|\mathbf{W}\|.$$

Typically, k is small and thus the matrix can be inverted via dense linear algebra for little computational cost. Algorithm 1 summarizes the technique for rapidly applying the inverse of $\mathbf{A}_{\text{ext}}$ provided a fast direct solver for $\mathbf{A}_{o,o}$ has already been computed. The algorithm is designed so that it can be used with any fast direct solver including the HBS [10], HSS [11, 12], HIF [13], and $\mathscr{H}$ or $\mathscr{H}^2$- matrix methods [14]. Section 3.3 presents fast techniques for creating the low-rank factorizations of $\mathbf{Q}$ and Appendix A reviews some known theoretical results on the stability for using the Woodbury formula for applying the inverse in (12).

Given a fast direct solver for the original discretization $\mathbf{A}_{o,o}^{-1}$, and the right-hand-side vector defined for the refined discretization $\mathbf{g}_n = \begin{pmatrix} \mathbf{g}_k \\ \mathbf{g}_p \end{pmatrix}$, this algorithm determines the solution to the refined problem (10) by obtaining the solution to the equivalent ELS via a Woodbury formula (12).

Pre-computation:
    Step 1: Factorize the update matrix $\mathbf{Q} \approx \mathbf{LR}$ via the method in Section 3.3 and Section A.
    Step 2: (invert $\mathbf{A}_{p,p}$)
    if N_p is small,
        Evaluate and invert $\mathbf{A}_{p,p}^{-1}$ via dense linear algebra.
    else,
        Build an approximate inverse of $\mathbf{A}_{p,p}$ via a fast direct solver such as HBS.
    end if
    Step 3: Apply the applying scheme for $\mathbf{A}_{o,o}^{-1}$ and $\mathbf{A}_{p,p}^{-1}$ to evaluate $\mathbf{X} = \tilde{\mathbf{A}}^{-1}\mathbf{L}$.
    Step 4: Evaluate and invert the Woodbury operator $(\mathbf{I} + \mathbf{RX})$ via dense linear algebra.

Solve:
    Step 1: Evaluate $\tilde{\mathbf{A}}^{-1}\mathbf{g}_{\text{ext}} = \begin{pmatrix} \mathbf{A}_{o,o}^{-1}\begin{pmatrix} \mathbf{g}_k \\ \mathbf{0} \end{pmatrix} \\ \mathbf{A}_{p,p}^{-1}\mathbf{g}_p \end{pmatrix}$ utilizing the fast matrix vector applies provided
        by the direct solver(s).
    Step 2: Evaluate τ_{ext} via the Woodbury formula (12).

Algorithm 1 Applying the fast direct solver for the locally refined problem

Remark 3.2 The factorization technique for $\mathbf{Q}$ (step 1 in Algorithm 1) to be discussed in Section 3.3 scales linearly with respect to N_k, N_c and N_p, the number of discretization points in $\mathcal{I}_k$, $\mathcal{I}_c$, and $\mathcal{I}_p$ respectively. Thus, if a fast direct solver is constructed for $\mathbf{A}_{p,p}^{-1}$, then all steps in pre-computation and solve of Algorithm 1 scale linearly with respect to N_k, N_c and N_p. Otherwise, Algorithm 1 scales linearly with respect to N_k and N_c but cubically with respect to N_p due to the dense linear algebra calculations for $\mathbf{A}_{p,p}^{-1}$. Table 1 lists the cost scaling of Algorithm 1 and the fast direct solver in [7]. More details on the step-by-step cost analysis are given in [7]. The scaling for the fast direct solver given in [8] is the same as Algorithm 1.

3.3 Efficient construction of the low-rank approximation for Q ≈LR

The low-rank approximation of $\mathbf{Q}$ can be constructed by concatenating the low-rank factorizations for the non-zero blocks in it;

$$
\underset{2N_k \times 2N_c}{\mathbf{A}_{k,c}} \approx \underset{2N_k \times k_{kc}}{\mathbf{L}_{k,c}} \underset{k_{kc} \times 2N_c}{\mathbf{R}_{k,c}}, \qquad \underset{2N_k \times 2N_p}{\mathbf{A}_{k,p}} \approx \underset{2N_k \times k_{kp}}{\mathbf{L}_{k,p}} \underset{k_{kp} \times 2N_p}{\mathbf{R}_{k,p}}, \qquad \text{and}
$$

$$
\underset{2N_p \times 2N_k}{\mathbf{A}_{p,k}} \approx \underset{2N_p \times k_{pk}}{\mathbf{L}_{p,k}} \underset{k_{pk} \times 2N_k}{\mathbf{R}_{p,k}}. \tag{13}
$$

Thus the low-rank factorization of $\mathbf{Q}$ can be expressed as

$$
\underset{2N_{\text{ext}} \times 2N_{\text{ext}}}{\mathbf{Q}} \approx \underset{2N_{\text{ext}} \times k}{\mathbf{L}} \underset{k \times 2N_{\text{ext}}}{\mathbf{R}} \tag{14}
$$

where

$$
\mathbf{L} = \begin{bmatrix} \mathbf{0} & -\mathbf{L}_{k,c} & \mathbf{L}_{k,p} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{p,k} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{R} = \begin{bmatrix} \mathbf{R}_{p,k} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{k,c} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_{k,p} \end{bmatrix},
$$

$k = k_{pk} + k_{kc} + k_{kp}$ and $N_{\text{ext}} = N_k + N_c + N_p$.

The construction of the low-rank factorization of $\mathbf{A}_{k,p}$ starts with defining a circle P^{div} for Γ_r which divides Γ_k into two parts: the far-field and near-field with respect

Table 1 Cost scaling for the fast direct solver in [7] and the proposed solver in Algorithm 1. The fast direct solver in [8] has the same scaling as Algorithm 1. Here N_k, N_c and N_p are the number of discretization points in $\mathcal{I}_k$, $\mathcal{I}_c$, and $\mathcal{I}_p$ respectively

Method	Pre-computation	Solve
[7] with dense linear algebra for $\mathbf{A}_{p,p}$	$O\left(N_k + N_c^3 + N_p^3\right)$	$O\left(N_k + N_c^2 + N_p^2\right)$
[7] with fast direct solver for $\mathbf{A}_{p,p}$	$O\left(N_k + N_c^3 + N_p\right)$	$O\left(N_k + N_c^2 + N_p\right)$
Algorithm 1 with dense linear algebra for $\mathbf{A}_{p,p}$	$O\left(N_k + N_c + N_p^3\right)$	$O\left(N_k + N_c + N_p^2\right)$
Algorithm 1 with fast direct solver for $\mathbf{A}_{p,p}$	$O\left(N_k + N_c + N_p\right)$	$O\left(N_k + N_c + N_p\right)$

to Γ_r. Figure 3(a) illustrates this separation. Let the superscript notation denote "far" and "near" parts of Γ_k. The separation corresponds to classifying the rows $\mathbf{A}_{k,p}$ into two groups; the near- and far-field interactions. We first construct low-rank approximations to the far-field and near-field interaction separately and then merge them together for a final low-rank approximation $\mathbf{L}_{k,p}\mathbf{R}_{k,p} \approx \mathbf{A}_{k,p}$.

For far-field interaction, the potential due to points in $\mathcal{I}_p$ evaluated at points on Γ_k^{far} can be approximated by a linear combination of basis functions defined on any proxy surface that shields Γ_r away from Γ_k^{far}. Let P^{bas} denote the shielding proxy circle for Γ_r. Here P^{bas} is chosen to have a smaller radius as P^{div} but the same center. Figure 3(b) illustrates an example of these circles. A low-rank approximation for $\mathbf{A}_{k,p}^{\text{far}}$ can be constructed via an interpolative decomposition (ID) approximation (as defined below) for the matrix $\mathbf{A}_{k,\text{bas}}^{\text{far}}$ which captures the interaction between points on Γ_k^{far} and P^{bas}. This is similar to the far-field compression idea in [1, 10, 25]. The collection of skeleton row indices J_k^{far} from the ID for $\mathbf{A}_{k,\text{bas}}^{\text{far}}$ correspond to discretization points (or degrees of freedom) on Γ_k. Let $\mathbf{P}_k^{\text{far}}$ denote the interpolation matrix, then a low-rank approximation to $\mathbf{A}_{k,p}^{\text{far}}$ can be defined as $\mathbf{A}_{k,p}^{\text{far}} \approx \mathbf{L}_{k,p}^{\text{far}}\mathbf{R}_k^{\text{far}}$ where $\mathbf{L}_{k,p}^{\text{far}} = \mathbf{P}_k^{\text{far}}$ and $\mathbf{R}_k^{\text{far}} = \mathbf{A}_{k,p}^{\text{far}}(J_k^{\text{far}}, :)$. Here $\mathbf{A}_{k,p}^{\text{far}}(J_k^{\text{far}}, :)$ denotes the submatrix of $\mathbf{A}_{k,p}^{\text{far}}$ with rows specified by J_k^{far}.

Definition 3.2 Given a tolerance ϵ and a $m \times n$ matrix $\mathbf{W}$ (assuming $m < n$), if there exists a positive integer $k_\epsilon \leq m$ and $m \times k_\epsilon$ matrix $\mathbf{P}$ and vector J such that

$$\|\mathbf{W} - \mathbf{PW}(J(1 : k_\epsilon), :)\| \leq \epsilon\|\mathbf{W}\|,$$

we call $\mathbf{PW}(J(1 : k_\epsilon),:)$ an interpolative decomposition (ID) approximation for $\mathbf{W}$ with respect to the tolerance ϵ. Here J is a vector of integers j_i such that $1 \leq j_i \leq m$ gives an ordering of the rows in $\mathbf{W}$, and $\mathbf{W}(J(1 : k_\epsilon),:)$ is a submatrix of $\mathbf{W}$ with rows specified by the first k_ϵ entries of J. $\mathbf{P}$ is a $m \times k_\epsilon$ matrix that contains a k_ϵ
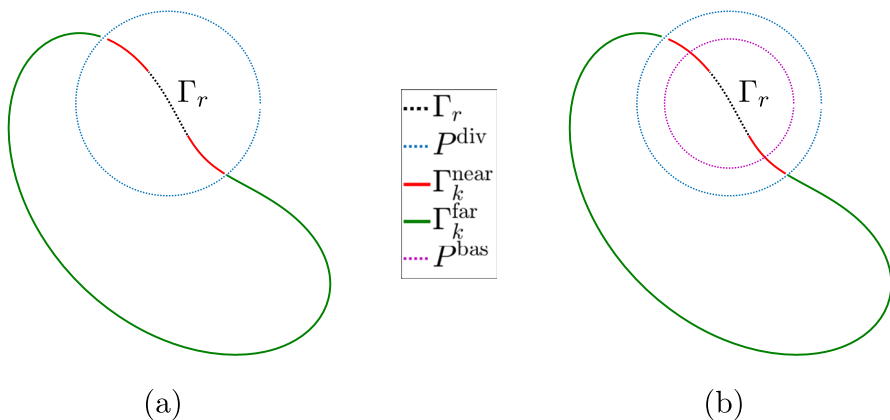

(a)   (b)

Fig. 3 (**a**) The proxy circle for Γ_r shown in dash blue line divides Γ_k into far (in green) and near (in red) with respect to Γ_r (**b**) The interaction between the far-field part of Γ_k and points on Γ_r can be captured by the interaction between points on Γ_k^{far} and a smaller proxy circle for Γ_r shown in dash purple

$\times \ k_\epsilon$ identity matrix. Namely, $\mathbf{P}(J(1 : k_\epsilon), :) = \mathbf{I}_{k_\epsilon}$. The rows specified by $J(1 : k_\epsilon)$ is called the *skeleton row index*, and the matrix $\mathbf{P}$ is referred to as the *interpolation matrix*. We are mostly interested in the case where k_ϵ is strictly smaller than m and for such case the ID approximation is often referred to as a "compression" for the original matrix $\mathbf{W}$. k_ϵ is called the rank number for the compression. An ID approximation with rank number k_ϵ is considered to be an optimal compression in rank number if any ID approximations of rank number $k < k_\epsilon$ leads to $\|\mathbf{W} - \mathbf{PW}(J(1 : k_\epsilon),:)\| > \epsilon \|\mathbf{W}\|$.

Due to the large number of discretization points on Γ_k^{far}, it is often too expensive to build the ID for $\mathbf{A}_{k,\text{bas}}^{\text{far}}$ directly. In practice, we organize the discretization points on Γ_k^{far} into special structure such as the dyadic partition (see Fig. 4 for an example) or binary tree (see Fig. 5 for an example). The goal of using the special structure is to keep the cost of building the low-rank approximation linear with respect to the number of points on Γ_k^{far}.

The choice of structure for creating the low-rank factorization depends on how localized the refinement is and the position of the portion of the boundary to be refined Γ_r relative to Γ_k. For example, the channel example given in Section 5.1 considers two kinds of local changes to the channel geometry in Fig. 7(a): a very localized refinement of the discretization illustrated in Fig. 7(b); and a geometric perturbation consisting of the addition of three interior circular holes as illustrated in Fig. 7(c). For the problem in Fig. 7(b), the far-field and near-field separation is straightforward and a dyadic partition of the far-field points on Γ_k based on distance to Γ_r is convenient and efficient. However, for the problem in Fig. 7(c), since the three holes do not cluster, a circle enclosing all holes would contain a large section of the channel boundary if not all of it. This means that many points on Γ_k will be "near-field" points although they are quite far away from any of the holes. An efficient way to handle this problem is to introduce three circles each enclosing an individual hole and define P^{bas} to be the union of the three circles. And a binary tree, which does not have to depend on distance to Γ_r, is a more appropriate choice. Figure 4 plots an example dyadic partition for the refined channel problem in Fig. 7(b), and Fig. 5 plots the first three levels of an example binary tree structure for the addition of holes problem in Fig. 7(c). More details on how to construct the hierarchical ID given the appropriate special structure can be found in Section 3 of [26] and Section 4 and 6 of [10]).

If there are not a large number of points that are near, which is often the case, the near-field interaction matrix $\mathbf{A}_{k,p}^{\text{near}}$ can be compressed directly. Otherwise, a dyadic partition of discretization points on Γ_k^{near} based on their distance to Γ_r can be adopted. The ID for $\mathbf{A}_{kp}^{\text{near}}$ can then be constructed in a hierarchical way utilizing the idea of tree-node wise proxy circles (See Section 3 of [26]).

Once both far and near part of $\mathbf{A}_{k,p}$ are compressed, the low-rank factors can be concatenated to form a low-rank approximation of $\mathbf{A}_{k,p}$. One may want to apply ID again to the concatenated factors to further reduce the rank numbers.

Since the removed points $\mathcal{I}_c$ and added points $\mathcal{I}_p$ discretize the same boundary curve segment Γ_r, the far-field part of the low-rank approximation for $\mathbf{A}_{k,p}$ and $\mathbf{A}_{k,c}$
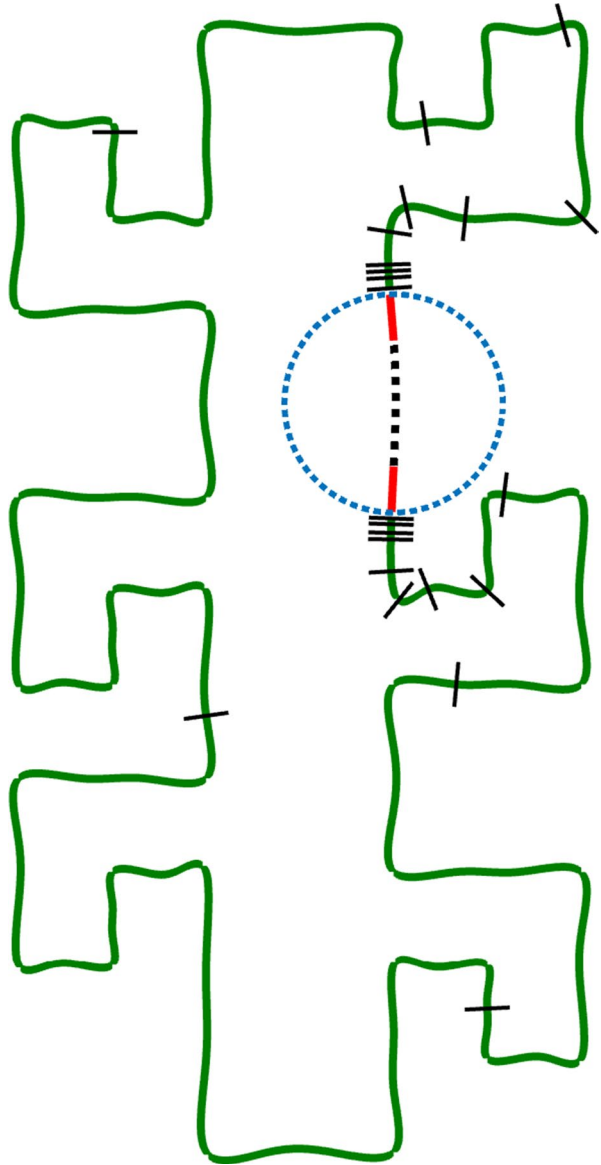
can be built from the same ID approximation as $\mathbf{A}_{k,\text{bas}}^{\text{far}}$. This is possible since the construction of the approximations does not require explicit evaluation of the matrices $\mathbf{A}_{k,p}$ and $\mathbf{A}_{k,c}$. Only the submatrices corresponding to the skeleton rows need to be evaluated for making the $\mathbf{R}$ matrices.

The near-field factorization of $\mathbf{A}_{k,c}$ can be constructed in similar fashion as the near-field factorization of $\mathbf{A}_{k,p}$. For $\mathbf{A}_{p,k}$, we consider again a far-field and near-field separation of the points on Γ_k based on the distance to Γ_r, which corresponds to

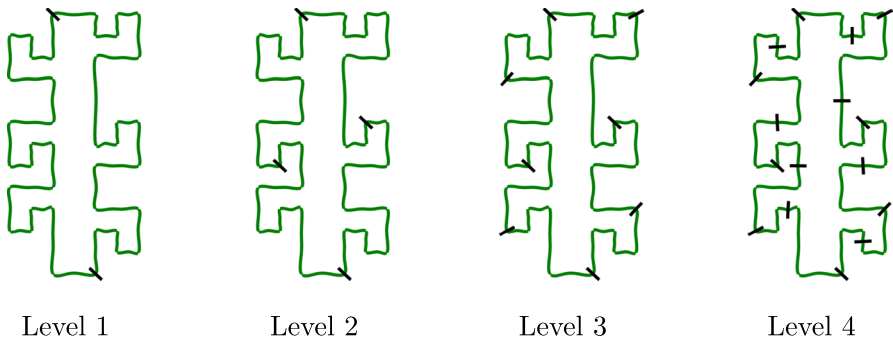| Level 1 | Level 2 | Level 3 | Level 4 |

Fig. 5 Top four levels of a binary tree partition for the channel geometry. The subintervals (or boxes) corresponding to each level are plotted

classifying the columns of the matrix into two groups. The far-field interaction $\mathbf{A}^{\text{far}}_{p,k}$ can be obtained by an ID the matrix that captures the interaction between the added points discretizing Γ_r and sample points on the separation circle P^{div} named $\mathbf{A}_{p,\text{div}}$. If the number of points added is large, we can relieve the computational burden by using a dyadic partition or binary tree in the same manner as was done for building the ID for $\mathbf{A}^{\text{far}}_{k,\text{bas}}$. The construction for $\mathbf{A}^{\text{near}}_{p,k}$ is similar to the near-field part of the approximation for the near-field of $\mathbf{A}_{k,p}$ and $\mathbf{A}_{k,c}$.

Remark 3.3 When approximating the three blocks in $\mathbf{Q}$, we always use ID to compress the rows of the matrices. We also uniformly define the $\mathbf{L}$ factor of the low-rank approximation to be the interpolation matrix (or product of multiple interpolation matrices if special tree structure is used) and the $\mathbf{R}$ factor to be the submatrix of the discretized BIE specified by the skeleton row indices given by the IDs. This uniform format for all three blocks is intentional as it improves the conditioning of applying the Woodbury formula. More details on this are presented in Appendix A. Note the block-wise compression technique given in [7] manages to compress all far-field part of the three blocks $\mathbf{A}_{k,p}$, $\mathbf{A}_{k,c}$, and $\mathbf{A}_{p,k}$ using one binary tree by doing row-wise ID for $\mathbf{A}_{k,p}$ and $\mathbf{A}_{k,c}$ but column-wise ID for $\mathbf{A}_{p,k}$. Namely, the far-field for all three blocks is approximated by the same set of skeleton points on Γ_k. For Laplace problems, the technique in [7] is expected to be more efficient than the one presented here especially for the case where Γ^{far}_k contains lots of points. But for Stokes problems, the mixed usage of row- and column-wise ID leads to conditioning issues and should be avoided.

With the special structure and partitioning, the cost of constructing the low-rank factorization for $\mathbf{A}_{k,p}$ is $O\big((N_k + N_p)k_{kp}\big)$. Similarly, the cost for factorizing $\mathbf{A}_{k,c}$ is $O\big((N_k + N_c)k_{kc}\big)$. And the cost of factoring $\mathbf{A}_{p,k}$ is $O\big((N_k + N_p)k_{pk}\big)$. With the low-rank factors for each block constructed, the final low-rank factorization for $\mathbf{Q}$ can be formed by simply concatenating the corresponding factors as in (13). If a close-to-optimal low-rank factorization (in terms of rank numbers) for

Q is desired, one can "re-compress" the factorization by building an ID for **L** and define the re-compressed final factorization accordingly.

4 A preconditioner for BVPs on locally refined discretization

The ELS presented in Section 3.2 is very useful for problems where there is local refinement of the discretization. While the fast direct solver for the ELS is efficient, it can suffer from a loss in accuracy when the problem has a high condition number. This is frequent occurrence for Stokes problems especially in complex geometries. An alternative to fast direct solvers is to use an iterative solver coupled with a fast matrix-vector multiplier such as the FMM in these instances. The large condition number often means that a large number of iterations are required for the iterative solver to converge. This section presents an alternative solution technique which is essentially the union of a fast direct solver with an iterative solver. Roughly speaking, the technique is to use the direct solver presented in Algorithm 1 as preconditioner for the ELS that is solved via an iterative solver coupled with a fast matrix-vector multiplier.

Section 4.1 details how the accuracy in which the direct solver is constructed impacts its ability to be a preconditioner. Then Section 4.2 details the preconditioner developed for the ELS (11).

4.1 HBS inverse approximation as preconditioner

It is becoming more common to use low-accuracy fast direct solvers as preconditioners for linear systems that arise from discretizations of integral equations and differential equations [17–21]. This section explores effectiveness of fast direct solvers as preconditioners for the discretized integral equation associated with an interior Stokes problem.

Consider the linear system $\mathbf{A}\boldsymbol{\sigma} = \mathbf{g}$ which results from the discretization of equation (4). Let ϵ denote the tolerance for which the fast direct solver was constructed and $\mathbf{A}_{\epsilon}^{inv}$ denote the corresponding approximate inverse of $\mathbf{A}$. Then the left-preconditioned problem is defined as

$$\left(\mathbf{A}_{\epsilon}^{inv}\mathbf{A}\right)\boldsymbol{\sigma} = \left(\mathbf{A}_{\epsilon}^{inv}\mathbf{g}\right). \tag{15}$$

To investigate the performance of the fast direct solver as a preconditioner with different tolerances ϵ, we consider the fish geometry in Fig. 6 with no local refinements. In particular, we place two hundred 16-point Gaussian panels uniform in parameterization space on the boundary. The linear system (15) is solved via GMRES [27]. The application of **A** and $\mathbf{A}_{\epsilon}^{inv}$ is done via the HBS technique from [10]. The performance of the solver will be the same for any fast direct solver. The tolerance for the compression of the matrix-vector operator, i.e., applying **A**, is fixed at 10^{-10}. The time for constructing the HBS representation of the matrix is 6.81 seconds on a single core 1.6GHz 8GB RAM desktop. Table 2

Fig. 6 Illustration of a fish
geometry where the red portion
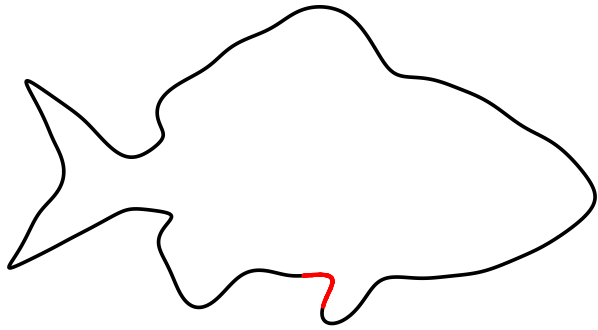of the boundary is refined

Table 2 Number of iteration n_{iter}, time in seconds to build the preconditioner T_{pre}, time in seconds for
GMRES to converge T_{sol} and the minimum number of solves MinSol needed to justify the use of the pre-
conditioner when using an HBS inverse approximation with accuracy ϵ as a preconditioner for the inte-
rior BIE on the fish geometry in Fig. 6. The boundary geometry is discretized with two hundred 16-point
Gaussian panels uniformly distributed in parameterization space. With this discretization, the average
relative solution error at sample locations on the interior is roughly 7×10^{-10}

ϵ	n_{iter}	T_{pre}	T_{sol}	MinSol
No preconditioner	55	NA	5.2e-1	NA
1e-10	2	7.66	6.6e-2	17
1e-8	2	4.92	1.0e-1	12
1e-6	4	2.89	1.3e-1	8
1e-5	6	2.23	1.5e-1	7
1e-4	11	1.74	2.2e-1	6
1e-3	36	1.12	5.1e-1	–
1e-2	52	1.11	7.9e-1	–
1e-1	53	0.97	8.1e-1	–

reports the performance of the preconditioned solution technique. For all experi-
ments, the tolerance of the iterative solver is set to 10^{-11} and the average rela-
tive error in the solution compared against the exact solution at sampled interior
locations is roughly 7×10^{-10}. The condition number for the Woodbury system
for this problem is reported in Table 14 of Appendix A. It shows that the linear
system $\mathbf{A}\boldsymbol{\sigma} = \mathbf{g}$ is well-conditioned. Thus even without a preconditioner, only 55
iterations are needed to achieve the desired tolerance. The results indicate that
low-accuracy approximations ($\epsilon > 10^{-3}$) do not improve the performance of the
iterative solver enough to justify constructing the preconditioner. For $\epsilon < 10^{-3}$,
the minimum number of repeated solves needed to justify the use of the precon-
ditioner grows as ϵ decreases. This experiment illustrates that the use of a low-
accuracy fast direct solver as preconditioner is not fruitful in improving the con-
vergence rate of iterative solvers. For problems where the condition number of
the discretized linear system is large, a preconditioner may be required for the

iterative solver to converge within reasonable number of iterations with the available computing resources.

4.2 Preconditioned iterative solver for the locally refined problem

Just like the discretized BIE for a Stokes boundary value problem on a given geometry, the ELS (11) can also suffer from conditioning issues. This section presents a preconditioner based on the solver from Section 3 and a fast matrix-vector multiplier that can be utilized to accelerate an iterative solver. It is expected that the number of iterations needed to converge will be less than if there was no preconditioner at all. Additionally, there is no loss of digits associated with inverting poorly conditioned matrices.

The idea behind the preconditioner is simple. Let $\mathbf{A}_{o,o}^{inv}$ and $\mathbf{A}_{p,p}^{inv}$ denote the approximate (or exact if the matrices are small enough) inverses of $\mathbf{A}_{o,o}$ and $\mathbf{A}_{p,p}$, respectively. Then

$$\tilde{\mathbf{A}}^{inv} = \begin{bmatrix} \mathbf{A}_{o,o}^{inv} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{p,p}^{inv} \end{bmatrix} \approx \tilde{\mathbf{A}}^{-1}$$

and

$$\mathbf{A}_{\text{ext}}^{inv} = \tilde{\mathbf{A}}^{inv} - \tilde{\mathbf{A}}^{inv}\mathbf{L}\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{inv}\mathbf{L}\right)^{-1}\mathbf{R}\tilde{\mathbf{A}}^{inv} \approx \mathbf{A}_{\text{ext}}^{-1}.$$

The Woodbury formula can be applied efficiently to any vector via the technique presented in Algorithm 1.

Instead of solving the true ELS, we propose solving the approximation of the linear system (11) where $\mathbf{A}_{\text{ext}}$ is approximated by a block diagonal plus low-rank form; i.e., as $\mathbf{A}_{\text{ext}} \approx \left(\tilde{\mathbf{A}} + \mathbf{LR}\right)$. The matrix $\tilde{\mathbf{A}}$ can be applied to a vector $\mathbf{b}$ block-wise

$$\tilde{\mathbf{A}}\mathbf{u} = \begin{bmatrix} \mathbf{A}_{o,o} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{p,p} \end{bmatrix}\begin{bmatrix} \mathbf{b}_o \\ \mathbf{b}_p \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{o,o}\mathbf{b}_o \\ \mathbf{A}_{p,p}\mathbf{b}_p \end{bmatrix}.$$

The evaluation of $\mathbf{A}_{o,o}\mathbf{b}_o$ can be accelerated via fast matrix-vector multiplication algorithms, such as the FMM or the approximate forward operator created in the process of building a fast direct solver, and is constructed for the original discretization. Similar to the fast direct solver for the ELS presented in Algorithm 1, if N_p is small, the matrix $\mathbf{A}_{p,p}$ can be constructed and applied via dense linear algebra. Otherwise, a separate fast matrix-vector multiplication can be constructed for $\mathbf{A}_{p,p}$. Since $\mathbf{L}$ and $\mathbf{R}$ are block sparse and low rank, they can be applied to any vector densely with little cost.

In this paper, we assume a forward HBS representation, the HBS inverse, and matrix-vector multiplication for applying $\mathbf{A}_{o,o}$ and $\mathbf{A}_{o,o}^{inv}$ are available. Then the ELS for the problem defined on the refined geometry

$$\left(\tilde{\mathbf{A}} + \mathbf{Q}\right)\boldsymbol{\tau}_{\text{ext}} \approx \left(\tilde{\mathbf{A}} + \mathbf{LR}\right)\boldsymbol{\tau}_{\text{ext}} = \mathbf{g}_{\text{ext}} \tag{16}$$

Table 3 Number of iterations n_{iter}, time in seconds for computing the preconditioner T_{pre} and time in seconds for the iterative solver to converge T_{sol} when applying the ELS preconditioner to the boundary value problem on the refined fish geometry in Fig. 6. The red portion of the boundary is refined. Originally there were $N_c = 128$ points on the red portion. In the new problem there are $N_p = 1024$ points on the red portion of the boundary. The number of points unchanged is $N_k = 3072$. We assume an HBS representation and the inverse for the original problem are available

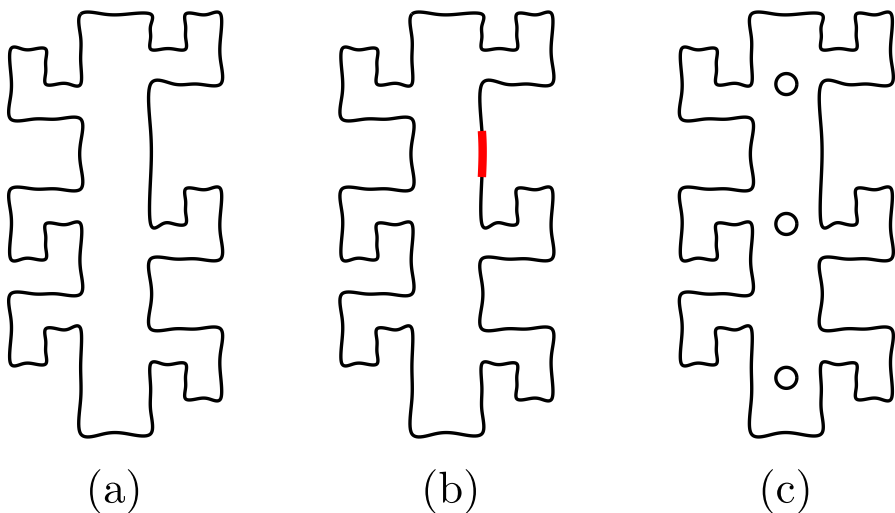| Method	n_{iter}	T_{pre}	T_{sol}
GMRES with fast mat-vec	55	NA	4.8e-1
GMRES with preconditioner	2	7.2e-1	8.3e-2

Fig. 7 (**a**) The original channel geometry. (**b**) The channel geometry with a locally refined segment highlighted in red. (**c**) The channel geometry with three interior holes added

only requires building the low-rank factorization of the blocks in **Q** and the operators associated with the $\mathbf{A}_{p,p}$ block and can be solved by an iterative solver such as GMRES. $\mathbf{A}_{\mathrm{ext}}^{inv}$ can be constructed with the extra cost of carrying out the Woodbury formula and applied as a preconditioner to (16). For a well-conditioned problem, where many different choices of local refinements and/or right-sides are considered, the total cost may be greatly reduced by using the fast direct solver in Section 3 as a preconditioner. Table 3 reports the performance of the preconditioner when it is applied to the boundary value problem on the fish geometry in Fig. 6 where the red region of the boundary is refined. The original discretization has two hundred 16-point Gaussian panels uniformly distributed in parameterization space; 8 panels discretize the red region and are replaced by 64 panels for the refinement. The number of discretization points kept was $N_k = 3072$, the number of discretization points cut was $N_c = 128$ and the number of discretization points added was $N_p = 1024$. The tolerance for HBS compression and low-rank approximations were set to 10^{-10}, and

the tolerance for GMRES was set to 10^{-11}. The average relative error of the solution at sampled locations is roughly 7×10^{-10} for both tests. Recall, we assume the HBS representation of $\mathbf{A}_{o,o}$ and its inverse are available. Thus the time needed to construct these is not included in our results. The results in the first row of Table 3 are for when the fast matrix-vector multiplication for $\tilde{\mathbf{A}}$ uses the precomputed HBS representation of $\mathbf{A}_{o,o}$. The time for constructing the efficient forward apply of the ELS $(\tilde{\mathbf{A}} + \mathbf{LR})$ is 0.53 second, which includes the construction of $\mathbf{A}_{p,p}$ and the low-rank factorization $\mathbf{Q} \approx \mathbf{LR}$. As expected the number of iterations is the same as in Table 2. The second row in Table 3 presents the results when the preconditioner is used. The extra time required to construct the preconditioner T_{pre}, i.e., for constructing $\mathbf{A}_{\text{ext}}^{inv}$, includes everything else that was not included in constructing the efficient forward apply of the ELS $(\tilde{\mathbf{A}} + \mathbf{LR})$ such as the construction and inversion of the Woodbury operator. Again the results are comparable to the results in the previous section. The preconditioner reduces the number of iterations from 55 to 2, resulting in an 82.7% reduction in solve time. And the extra cost for building the preconditioner is justified for problems involving more than one right-hand side.

If the problem is not well-conditioned, then the preconditioner may be necessary to obtain an accurate solution with a limited amount of computational resources.

5 Numerical experiments

This section illustrates the performance of the proposed solution techniques for Stokes problems involving locally refined discretizations. The fast direct solver scales linearly with respect to the number of points in the original discretization and is cheaper than building a fast direct solver from scratch for the new discretization. Section 5.1 illustrates the performance of the fast direct solver when applied to a locally refined channel. This example is from [16]. Section 5.2 reports on the performance of the fast direct solver as a preconditioner when the geometry is complex. Finally Section 5.3 illustrates the performance of the fast direct solver as a preconditioner when there are a sequence of local refinements for the same original geometry. Such an example arises in many applications including simulations of microfluidic devices.

For all test problems, the right-hand side of the BVPs is generated from a known flow and the solution error is the average of relative error at chosen target locations in the domain. All boundaries are discretized via the Nyström method with 16-point composite Gaussian quadrature, and generalized Gaussian quadrature corrections [28] are used to handle the weakly singular kernels. The solver also works with other quadrature corrections, such as [29–31].

All experiments were run on a dual 2.3 GHz Intel Xeon Processor E5-2695 v3 desktop workstation with 256 GB of RAM. The code is implemented in MATLAB, apart from the interpolatory decomposition routine, which is in FORTRAN.

To illustrate the performance of the solver, we introduce the following notations for reporting times and errors. For notation consistency, we use regular capital letters such as T and E for problems defined on the original discretization (or geometry) and letters with tilde, such as $\tilde{T}$ and $\tilde{E}$ for problems on the locally

refined discretization (or perturbed geometry). For the problem on the original discretization (or geometry), we define

- $T_{\text{HBS, comp}}$ and $T_{\text{HBS, inv}}$: the time in seconds for building the HBS compression of the discretized boundary integral operator and that for inverting the compression, i.e., building the HBS inverse, respectively.
- $T_{\text{HBS, Dsol}}$: the time in seconds for applying the HBS inverse to a given right-hand side vector. "Dsol" stands for "one direct solve".
- $T_{\text{HBS, Gsol}}$: the time in seconds for solving for one right-hand side vector using GMRES with HBS compression accelerated matrix-vector multiplication. "Gsol" stands for "one GMRES solve".
- $T_{\text{HBS, PGsol}}$: the time in seconds for solving for one right-hand side vector using a preconditioned GMRES with HBS compression accelerated matrix-vector multiplication, where the HBS inverse is used as the preconditioner. "PGsol" stands for "one preconditioned GMRES solve".
- $E_{\text{HBS, Dsol}}$, $E_{\text{HBS, Gsol}}$ and $E_{\text{HBS, PGsol}}$: the average relative error at sample domain locations for the three different solve options respectively.

For the problem on the locally refined discretization (or perturbed geometry), we define

- $\tilde{T}_{\text{HBS, comp}}$, $\tilde{T}_{\text{HBS, inv}}$, $\tilde{T}_{\text{HBS, Dsol}}$, $\tilde{T}_{\text{HBS, Gsol}}$, and $\tilde{T}_{\text{HBS, PGsol}}$: time in seconds similar to those categories for the original discretization (or geometry).
- $\tilde{E}_{\text{HBS, Dsol}}$, $\tilde{E}_{\text{HBS, Gsol}}$ and $\tilde{E}_{\text{HBS, PGsol}}$: error similar to those categories for the original discretization (or geometry).
- $\tilde{T}_{\text{ELS, comp}}$: the time in seconds for building $\mathbf{A}_{p,p}$ and $\mathbf{LR} \approx \mathbf{Q}$ in formulating the fast ELS approximation. Note we assume a HBS compression for $\mathbf{A}_{o,o}$ is available.
- $\tilde{T}_{\text{ELS, inv}}$: the time in seconds for building the operators needed in the Woodbury formula for applying the inverse approximation of the ELS: $\mathbf{A}_{p,p}^{-1}$, $\tilde{\mathbf{A}}^{-1}\mathbf{L}$, $\mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}$ and $\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right)^{-1}$. Note we assume a HBS inverse approximation for $\mathbf{A}_{o,o}$ is available.
- $\tilde{T}_{\text{ELS, Dsol}}$: the time in seconds for applying the approximate ELS inverse $(\tilde{\mathbf{A}} + \mathbf{LR})^{-1}$ via the Woodbury formula to a given right-hand side vector.
- $\tilde{T}_{\text{ELS, Gsol}}$: the time in seconds for solving the approximate ELS $(\tilde{\mathbf{A}} + \mathbf{LR})\tau_{\text{ext}} = \mathbf{g}_{\text{ext}}$ for one right-hand side vector $\mathbf{g}_{\text{ext}}$ using GMRES.
- $\tilde{T}_{\text{ELS, PGsol}}$: the time in seconds for solving the approximate ELS $(\tilde{\mathbf{A}} + \mathbf{LR})\tau_{\text{ext}} = \mathbf{g}_{\text{ext}}$ for one right-hand side vector $\mathbf{g}_{\text{ext}}$ using GMRES, where the approximate ELS inverse is used as the preconditioner.
- $\tilde{E}_{\text{ELS, Dsol}}$, $\tilde{E}_{\text{ELS, Gsol}}$ and $\tilde{E}_{\text{ELS, PGsol}}$: the average relative error at sample domain locations for the three different ELS solve options respectively.

The accuracy for HBS compression and low-rank approximation is set to 10^{-10} unless specified otherwise.

5.1 Asymptotic scaling experiments

This section illustrates the performance of the fast direct solver for the ELS (presented in Section 3) when applied to a Stokes problem with a confined geometry with two types of modifications: locally refining a part of the boundary and adding holes. Figure 7(a) illustrates the channel. Figure 7(b) and (c) illustrate the modification of local refinement (in red) and adding holes, respectively. The geometry is generated by applying cubic splines with periodic conditions to 121 spline knot locations (with the first and last knots giving the same physical point on the geometry) and was first seen in [16]. The channel is discretized by using the same number of Gaussian panels per subinterval in in the cubic spline geometry generation. For example, the total number of discretization points on the channel $N_{channel} = 1920$ corresponds to 120 Gaussian panels in total and 1 panel per subinterval. If there are two panels per subinterval, the number of discretization points doubles.

The circular holes are each discretized with 10 panels which means there are 160 quadrature points per circle.

Remark 5.1 The addition of holes is similar to the original examples used in [16] and fits in the definition of a locally perturbed geometry as defined in [7, 8]. However, the extended system is slightly different from the one given in Section 2.2 as we are only adding points for the new boundary and there is no deletion or cutting of points on the original geometry. The corresponding ESL formulation is given in Appendix B.

The Dirichlet boundary data for the interior channel BVP (Fig. 7(a) and (b)) is generated by 5 exterior Stokeslets outside of the channel geometry. For these two problem, the solution is represented with the double layer kernel (as discussed in Section 2.1). The Dirichlet boundary data for the BVP with holes (Fig. 7(c)) is generated by the same 5 exterior Stokeslets outside the channel geometry and five additional Stokeslets placed inside the added holes (two stokelets per hole for the bottom two holes and one stokelet in the top hole). The solution interior to the channel and exterior to the holes is represented as a double layer potential on the channel plus a combined field potential on the holes.

The observed condition number of the discretized integral operator for all the problems in this section is on the order of 10^5. The condition number of the Woodbury operator is on the order of 10^3 for the problem with the added holes and 10^5 for the problem with the local refinement. The observed rank numbers for the low-rank approximation of the update matrix $\mathbf{Q}$, which is also the size of the Woodbury system, is roughly 60 for the problem with the local refinement and 340 for the problem with the added holes.

Let $N_{channel}$ denote the number of discretization points on the original channel. Table 4 reports on the performance of the HBS solver applied to the original geometry (and discretization). For the locally refined discretization, let N_c and N_p denote the number of points removed and added, respectively. For the channel

Table 4 The time in seconds and error for using HBS compression and inversion to solve the BIE on the channel geometry with the original discretization (illustrated in Fig. 7(a)). The number of discretization points on the channel is $N_{channel}$. The size of the linear system is $2N_{channel} \times 2N_{channel}$

| $2N_{channel}$	$T_{HBS, comp}$	$T_{HBS, inv}$	$T_{HBS, sol}$	$E_{HBS, Dsol}$
30720	65.7	7.0	0.070	1.1e-10
61440	90.7	9.9	0.140	1.4e-10
122880	132.8	15.8	0.264	3.22e-10

Table 5 The time in seconds and error for using the Woodbury formula to solve the ELS for the boundary value problem on the channel with local refinement illustrated in Fig. 7(b). The number of discretization points cut and added on the red portion of the boundary are N_c and N_p respectively

$2N_{channel}, 2N_c, 2N_p$	$\tilde{T}_{ELS, comp}$	$\tilde{T}_{ELS, inv}$	$\tilde{T}_{ELS, Dsol}$	$\tilde{E}_{ELS, Dsol}$
30720, 192, 768	1.4	0.8	0.088	2.5e-10
61440, 384, 1536	3.0	1.2	0.113	5.8e-10
122880, 768, 3072	7.1	2.3	0.184	4.8e-10

Table 6 The time in seconds and error for using the Woodbury formula to solve the ELS for the boundary value problem on the channel with three interior holes illustrated in Fig. 7(c). The circular holes in Fig. 7(b) are each discretized with 10 panels and 160 quadrature points, resulting a total of $N_{holes} = 480$ points

$2N_{channel}, 2N_{holes}$	$\tilde{T}_{ELS, comp}$	$\tilde{T}_{ELS, inv}$	$\tilde{T}_{ELS, Dsol}$	$\tilde{E}_{ELS, Dsol}$
30720, 960	14.9	2.1	0.059	9.8e-11
61440, 960	28.8	4.3	0.103	2.3e-10
122880, 960	56.4	7.9	0.296	2.1e-10

Fig. 8 Partial Fallopian tube based on data extracted from the experiments in [32]. A small segment highlighted in red is chosen to be locally refined. The geometry is generated by [33]
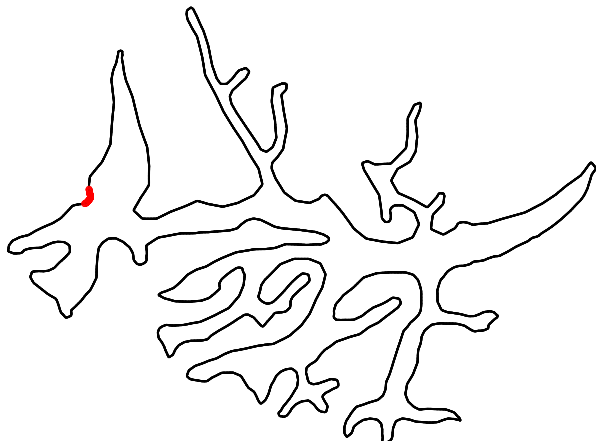
with holes geometry, let N_{holes} be the total number of discretization points placed on the three holes. The results in Tables 5 and 6 report on the performance of the proposed ELS formulation based fast direct solver applied to the geometries in Fig. 7(b) and (c). The size of each test case is given by the total degree of freedom, which is double the number of discretization points. To show the scaling of the ELS fast direct solver, the values for N_{channel}, N_c and N_p are all doubled as the test size increases. Both the HBS solver and the proposed fast direct solver scale linearly with respect to the number of points on the channel geometry. The cost of using Algorithm 1 is significantly less than building the original HBS solver. This means that Algorithm 1 is more computationally efficient than building a fast direct solver from scratch for the new discretization. It is worth noting that the time required for building the ELS compression for the addition of holes example is much higher than that for refining the channel boundary given the same N_{channel}. For example, when $2N_{\text{channel}} = 122880$, ELS compression for adding holes is about 8 times of that for refining a segment, although the points added for the holes N_{holes} is only 1/3 of the points added N_p due to the refinement. This is due to the fact that the change to the system for adding the three holes is "less local" than that for refining a segment of the channel, resulting in much higher rank numbers and more expensive compression of the update matrix **Q**. For the same reason, the time required for applying the inverse of the ELS when adding holes is also more than that for refining a segment of the channel.

The solution error for all test cases is maintained at 10^{-10} since the geometry is fully resolved and the tolerance for HBS compression and low-rank approximations is set to be 10^{-10}.

5.2 Complex geometry with local refinement

This section considers an interior problem on the complex Fallopian tube geometry illustrated in Fig. 8. The geometry is created by extracting data points from Fig. 1 of [32] and connecting them smoothly via the technique in [33]. The solution to the problem is generated by placing Stokeslets on the exterior of the geometry. The boundary data is generated via this known solution. Discretizing the complex geometry in Fig. 8 results in an integral equation with a high condition number. An iterative solver requires a large number of iterations in order to converge. The experiments in this section discretize the original Fallopian tube boundary (pre-refinement) with 1600 Gaussian panels (25600 points and 51200 degrees of freedom), which results in relative error of approximately 4×10^{-5}. To understand the conditioning of the linear system, we consider the smallest matrices that are inverted in the hierarchical tree using the HBS solver. These matrices (corresponding to the first three levels in the tree) have condition numbers on the order of 10^8 to 10^{11}.

For the refined discretization problem, the red portion of the boundary highlighted in Fig. 8 goes from having 6 panels to 24. Since a 16 point Gaussian quadrature is used, the number of points kept, cut and added are $N_k = 25,504$, $N_c = 96$, and $N_p = 384$, respectively. The iterative solver stops when the relative residual is on the order of 10^{-6}. For the boundary integral equation on the original discretization, we

Table 7 Time in seconds for solving (a) the original and (b) the refined problem defined on the Fallopian tube geometry (Fig. 8) via HBS inversion. (c) corresponds to the proposed fast direct solver for the ELS of the refined problem

(a)

$T_{HBS, comp}$	$T_{HBS, inv}$	$T_{HBS, Dsol}$
4.09e + 2	2.76e + 1	7.87e-2

(b)

$\tilde{T}_{HBS, comp}$	$\tilde{T}_{HBS, inv}$	$\tilde{T}_{HBS, Dsol}$
4.09e + 2	2.76e + 1	7.87e-2

(c)

$\tilde{T}_{ELS, comp}$	$\tilde{T}_{ELS, inv}$	$\tilde{T}_{ELS, Dsol}$
5.33e + 0	2.01e + 0	9.62e-2

Table 8 Time in seconds for solving (a) the original and (b) the refined problem defined on the Fallopian tube geometry (Fig. 8) via GMRES with HBS compression accelerated matrix-vector multiplication. (c) corresponds to solving the ELS of the refined problem via GMRES

(a)

$T_{HBS, comp}$	$T_{HBS, Gsol}$
4.09e + 2	5.95e + 1 ($n_{iter} = 519$)

(b)

$\tilde{T}_{HBS, comp}$	$\tilde{T}_{HBS, Gsol}$
4.09e + 2	6.59e + 1 ($n_{iter} = 519$)

(c)

$\tilde{T}_{ELS, comp}$	$\tilde{T}_{ELS, Gsol}$
5.33e + 0	7.07e + 1 ($n_{iter} = 520$)

The number of GMRES iterations n_{iter} required to converge to tolerance 10^{-6} is also reported

either build only HBS representation of the discretized boundary integral equation and couple it with GMRES or also build the HBS inverse and apply it directly to the given right-hand side. For the refined problem, we consider the discretized BIE and the equivalent ELS and a fast direct solver and an iterative solver for each. Additionally, we also use the direct solver for the ELS, built as described in Algorithm 1, to precondition the GMRES solve.

Table 7 reports the time in seconds required to solve the BIE on the original discretization, the BIE on the refined discretization, and the approximate ELS on the refined discretization using a fast direct solver. The total time for precomputation includes two parts: the forward compression indicated by subscript notation "comp" and the inversion indicated by the subscript notation "inv". Table 7(b) and (c) demonstrate that for this geometry the proposed direct solver for the ELS is more efficient than building a HBS solver from scratch for the refined problem. In fact, the cost for constructing a forward compression for the ELS for the refined problem is only 1.3% of the cost of constructing a HBS from scratch. The cost of constructing the inverse operator is only 7.3% of that of HBS inverse.

Table 8 reports the time in seconds for the unpreconditioned GMRES approach for the original and refined problems. The precomputation for this approach only involves the compression of the forward operator and is lower than that for the direct solution approach since an approximate inverse is not constructed. However, due to the poor conditioning of the problem, more than 500 GMRES iterations are required

to reach the desired tolerance of 10^{-6}. The time required to solve the integral equation via unpreconditioned GMRES is much higher for each new right-hand side than the direct solver. Table 8(b) and (c) show that solving the approximate ELS (16) is two orders of magnitude cheaper than building the HBS compression of the BIE for the refined problem. Applying the forward operator for the ELS is slightly more expensive than applying the HBS forward compression.

Table 9 reports the time in seconds for the preconditioned GMRES approach applied to the approximate ELS. Here Algorithm 1, i.e., the inverse of the ELS obtained by the Woodbury formula, is used to precondition the fast representation of the ELS. The precomputation time of this approach is equal to that of the direct solver approach for the ELS. The number of GMRES iterations required for the convergence criterion to be met is reduced from 520 to 6, leading to a significant reduction in total cost even for only one right-hand side solve when compared to the results in Table 8(c). The cost of solving one additional right-hand side vector via the preconditioned GMRES approach for the ELS is about 5.4% of that via unpreconditioned GMRES approach.

5.3 Relocating region of local refinement

This section illustrates the potential of using the fast direct solver presented in Algorithm 1 as preconditioner for many Stokes problems involving a body moving through a collection of star-shaped obstacles shown in Fig. 9. This example is representative of applications such as sorting with a microfluidic device. For the original discretization, 10 panels are placed on each star with less or equal to 5 prongs and 20 panels are placed on stars with more than 5 prongs. With the 16-point Gaussian quadrature, this results in a total of 42400 discretization points and 84800 degrees of freedom. For demonstration purposes, we do not simulate the true physics of any body moving in the domain; instead, we assume the body appears at certain locations at some time step, as illustrated in in Fig. 9. These can be viewed as snapshots of a body moving through the obstacles. The body moving through the obstacles is much smaller in scale than any of the stars. Thus the discretization of one or more obstacles will need to be locally refined as the body approaches those obstacles. Since the body is moving, the regions of local refinement are expected to be different for each snapshot. Previously refined regions may be coarsened back into the original discretization as the body moves away. In this example, 19 snapshot locations are chosen. In 12 of these snapshots, the body is close to an obstacle and local refinement is needed. In the other 7

Table 9 Time in seconds for solving the ELS of the refined problem defined on the Fallopian tube geometry (Fig. 8) via preconditioned GMRES, where Algorithm 1 is used as the preconditioner. The number of GMRES iterations n_{iter} required to converge to tolerance 10^{-6} is also reported

$\tilde{T}_{ELS, comp}$	$\tilde{T}_{ELS, inv}$	$\tilde{T}_{ELS, PGsol}$
5.33e + 0	2.01e + 0	3.80e + 0 ($n_{iter} = 6$)

snapshots, no local refinement is needed. We consider 5 different ways of solving the linear system for the 19 different boundary value problems. These solution techniques are:

(1) `GMRES-indy`: Treat each of the 13 different discretization as independent boundary value problems, building a forward HBS representation for each, and using this to accelerate the GMRES solve for each snapshot;
(2) `Direct-indy`: Treat each of the 13 different discretization as independent boundary value problems and build a HBS solver for each one;
(3) `GMRES-Local`: Build a HBS forward representation for the original discretization and use it to accelerate the GMRES solve for the ELS for each problem requiring local refinement;
(4) `Direct-Local`: Build a HBS solver for the original discretization and use it to build a fast direct solver for the ELS according to Algorithm 1 for each problem needing local refinement;
(5) `PGMRES-Local`: Build a HBS solver for the original discretization and use it to precondition the GMRES solve for the ELS for each problem requiring local refinement.

The tolerance for GMRES is set to 10^{-11}. For the boundary value problems that do not require local refinement, using the HBS matrix-vector acceleration of GMRES results in a relative error on the order of 10^{-9}. Using the HBS solver loses two digits; i.e., the relative error that results from this solver is on the order of 10^{-7}. Thus for the two techniques (2) and (4) where the direct solver is used as an actual solver and not a preconditioner, the accuracy is approximately 10^{-7}. When the HBS solver or the ELS fast direct solver in Algorithm 1 is used as the preconditioner, the error is approximately 10^{-9}.

To compare efficiency of the five approaches, we first report the time in seconds for solving the problem on the original discretization and that on one particular refined discretization, which corresponds to the first snapshot with the body located at the very bottom left of Fig. 9(b). The results are presented in Tables 10, 11 and 12 in the same format as the corresponding results for the Fallopian tube geometry in the previous section.

Table 10 Time in seconds for solving (a) the original and (b) the refined problem defined on the star-shape obstacle geometry in Fig. 9 via HBS inversion. (c) corresponds to the proposed fast direct solver for the ELS of the refined problem

(a)		
$T_{\text{HBS, comp}}$	$T_{\text{HBS, inv}}$	$T_{\text{HBS, Dsol}}$
8.23e + 2	3.38e + 1	2.57e-1
(b)		
$\tilde{T}_{\text{HBS, comp}}$	$\tilde{T}_{\text{HBS, inv}}$	$\tilde{T}_{\text{HBS, Dsol}}$
8.07e + 2	3.38e + 1	2.98e-1
(c)		
$\tilde{T}_{\text{ELS, comp}}$	$\tilde{T}_{\text{ELS, inv}}$	$\tilde{T}_{\text{ELS, Dsol}}$
8.39e + 0	6.43e + 0	4.18e-1

Table 11 Time in seconds for solving (a) the original and (b) the refined problem defined on the star-shape obstacle geometry in Fig. 9 via GMRES with HBS compression accelerated matrix-vector multiplication. (c) corresponds to the solving the ELS of the refined problem via GMRES. The number of GMRES iterations n_{iter} required to converge to tolerance 10^{-11} is also reported

(a)	
$T_{HBS, comp}$	$T_{HBS, Gsol}$
$8.23e + 2$	$3.27e + 1$ ($n_{iter} = 113$)
(b)	
$\tilde{T}_{HBS, comp}$	$\tilde{T}_{HBS, Gsol}$
$8.07e + 2$	$3.30e + 1$ ($n_{iter} = 113$)
(c)	
$\tilde{T}_{ELS, comp}$	$\tilde{T}_{ELS, Gsol}$
$8.39e + 0$	$3.46e + 1$ ($n_{iter} = 113$)

Table 12 Time in seconds for solving the ELS of the refined problem defined on the star-shape obstacle geometry in Fig. 9 via preconditioned GMRES, where Algorithm 1 is used as the preconditioner. The number of GMRES iterations n_{iter} required to converge to tolerance 10^{-11} is also reported

$\tilde{T}_{ELS, comp}$	$\tilde{T}_{ELS, inv}$	$\tilde{T}_{ELS, PGsol}$
$8.39e + 0$	$6.43e + 0$	$9.50e + 0$ ($n_{iter} = 6$)

Since the direct solver does not achieve the full possible accuracy of the discretization, using it as preconditioner is reasonable and it greatly decreases the number of iterations needed for an iterative solver to converge.

With the step-by-step cost summarized in Tables 10, 11 and 12, we can approximate the total cost for each of the five approaches handling all 19 snapshots by simple addition and multiplication, assuming that the cost for solving the ELS for each snapshot that requires a refinement is the same.

To get an idea of the speed up for solving problems involving the 19 multiple snapshots given in Fig. 9(a), Table 13 collects the time necessary for each part of the 5 solution techniques. The different times reported are:

- T_{static}: The time in seconds for constructing any of the operators needed for the solution technique on the original discretization. For techniques (1) and (3), only constructing an approximation of $\mathbf{A}_{o,o}$ via HBS is needed. For the other options, the construction of the approximate inverse of $\mathbf{A}_{o,o}$ is also needed. This is a "static" computation since it is independent of future time steps and potential local refinement.
- T_{Osol}: The time in seconds for solving a problem where local refinement is not needed. "Osol" stands for "solve for the original discretization"
- T_{Rsol}: The time in seconds for solving a problem where local refinement is needed. "Rsol" stands for "solve for one refined discretization".
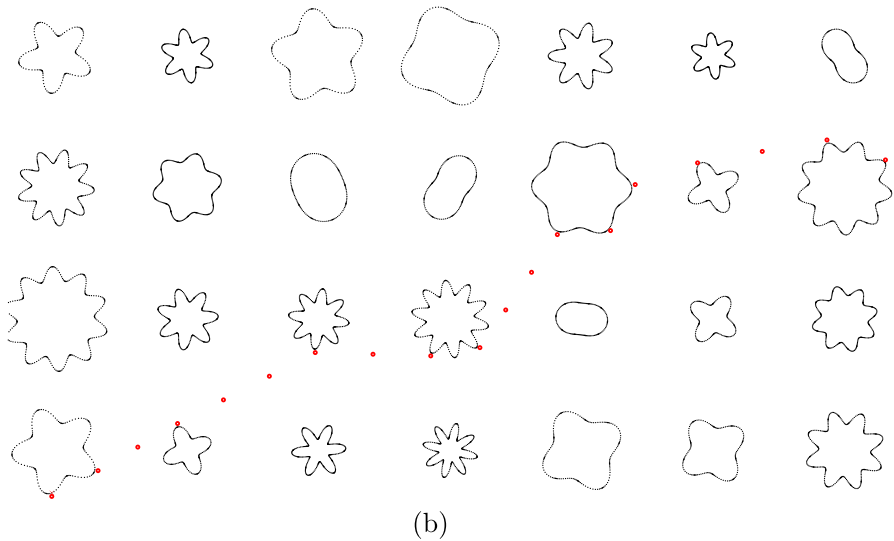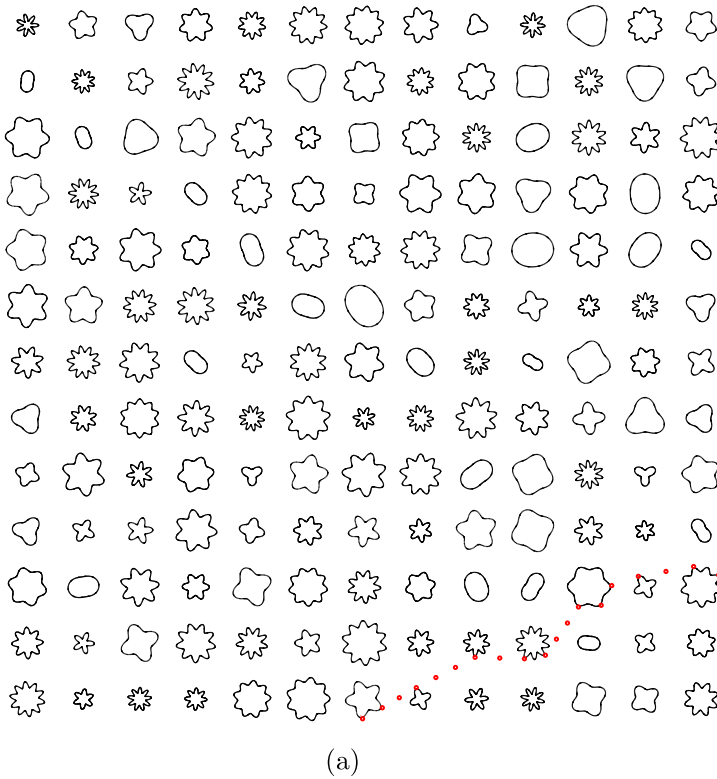
(a)

(b)

Fig. 9 (**a**) A collection of star-shape obstacles with different snapshots of body locations. (**b**) Zoomed-in in the region near the snapshots of the body locations. The locations are chosen artificially and do not represent any physical movement of body in Stokes flow. 19 locations are chosen, out of which 12 are close to certain part of the obstacle boundary and incur local refinement of the obstacle discretization

Table 13 Time in seconds of the construction of all necessary precomputed operators on the original discretization T_{static}, for solving a problem that does not need local refinement T_{Osol} and for solving a problem that requires local refinement T_{Rsol}. Here we assume that each of the locally refined discretization is the same in size and requires the same amount of calculations to solve

	T_{static}	T_{Osol}	T_{Rsol}
(1) `GMRES-indy`	8.23e + 2	3.27e + 1	8.40e + 2
(2) `Direct-indy`	8.56e + 2	2.57e-1	8.41e + 2
(3) `GMRES-Local`	8.23e + 2	3.27e + 1	4.29e + 1
(4) `Direct-Local`	8.56e + 2	2.57e-1	1.52e + 1
(5) `PGMRES-Local`	8.56e + 2	9.40e + 0	2.43e + 1

Approaches (3–5) which utilize the ELS are more efficient than building new HBS solver from scratch each time or only when there is local refinement. For these experiments Approach (4) is the most efficient but if the fully attainable accuracy is desired, Approach (5) should be used as it is both efficient and accurate. The previous standard solution technique for this type of problem was Approach (1). The proposed direct solver (4) and the proposed preconditioned solver (5) are 127 and 3.5 times faster than Approach (1) when local refinement is not needed. When refinement is needed, Approaches (4) and (5) are 55 and 34.6 times faster than Approach (1), respectively. Since the applications of interest (such as [1]) involve hundreds to thousands of solves, it is definitely worth using the ELS. If the user is okay losing a couple of digits, the fast direct solver is an ideal choice. If the digits are needed, then the preconditioned iterative solver is still going to be significantly faster than Approach (1).

Remark 5.2 The dominate cost T_{Rsol} for the ELS solution techniques is the cost of creating the low-rank factorization of **Q**. In most applications, several snapshots can use the same refinement and thus the same factorization of **Q**. The reuse of the factorization will decrease T_{Rsol} significantly. For example, in the experiments corresponding to the two body locations on the left bottom of Fig. 9, two different regions of the same five-prong star are refined in these two consecutive time steps. In practice it might be more efficient to group the two regions together and treat them as one locally refined region, thus leading to one refined discretization for the first two time steps.

6 Conclusions

This manuscript presented a fast direct solver for Stokes BIEs on locally refined discretizations. The technique makes use of an extended linear system that allows for precomputed fast direct solvers on the unrefined geometry to be utilized. The numerical results illustrate the new solver's performance on particulate flow simulations.

For general Stokes problems, two solution approaches are explored. Which solution technique should be used depends on the conditioning of the problem and how

many digits are desired. For well-conditioned problems, the proposed fast direct solver works extremely well. When the problem has poor conditioning, the fast direct solver may lose a couple of digits (relative to the compression accuracy). These digits can be recovered by using the second solution technique presented here, which is to utilize an iterative solver where the fast direct solver for the linear system serves as a preconditioner and the compressed representation of the ELS provides the fast matrix-vector multiply. Both solution techniques scale linearly with the size of the unrefined discretization. Linear scaling with respect to the number of unknowns added in the local refinement can also be achieved but is not necessary for the considered applications since a relatively low number of points are added. Numerical examples demonstrated significant speedups; in one test case, the proposed direct solver is roughly 55 times faster than the standard approach. For problems with large condition number, more accurate solution may be obtained by using the proposed preconditioner as compared to the direct solver. In another test example, the preconditioned GMRES solve for the ELS reduced the number of iterations by a factor of 19 (and total solve time by 3.6X). Our immediate future directions include incorporating close evaluation schemes and extension to three-dimensional problems.

Appendix : A. Stability of using the Woodbury formula

Woodbury formulas such as (12) are well-known in the linear algebra literature [34] and have been the cornerstone of recently developed fast direct solvers for applications including periodic Stokes flow [1] and quasi-periodic scattering problems [25, 26]. While the Woodbury formulas have been used in these applications, it was done so without any concern for the stability of the approach. This section will review the stability analysis of the Woodbury formula given in [35] and provides some empirical results in the case of Stokes problems.

The main concern in the stability of the Woodbury formula lies in the stable inversion of the matrix $\mathbf{W} = \mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}$. We will refer to the matrix $\mathbf{W}$ as the *Woodbury operator*. [35] states that in order to stably solve a linear system via the Sherman-Morrison-Woodbury formula, the following two conditions must be satisfied by the linear system:

- All the relevant matrix-matrix and matrix-vector multiplications in (12) involving $\tilde{\mathbf{A}}^{-1}$ are numerically stable.
- The Woodbury operator is well-conditioned.

For Stokes problems, the first condition is satisfied thanks to the choice of boundary integral formulation (in Section 2) and the use of a stable fast direct solver. Since Stokes problems tend to have a large condition number, we choose to modify the second condition to: (ii) The Woodbury operator is as "well-conditioned" as the full linear system $\tilde{\mathbf{A}} + \mathbf{LR}$.

The following lemma, which is a modified version of Lemma 1 in [35], provides an upper bound on the condition number of the Woodbury operator and

formally defines what we mean by the Woodbury operator being as "well-conditioned" as the full linear system. The lemma is stated in the context of discretized boundary integral operators and the ELS. Specifically it provides conditions on the low-rank approximation of the update matrix $\mathbf{Q} \approx \mathbf{LR}$ which must be satisfied (along with both the linear systems for the original and refined discretization being well-conditioned) for the Woodbury operator to be "well-conditioned."

Lemma A.1 (Upper bound on the condition number of the Woodbury operator) Assume the operator $\mathbf{A}_{o,o}$ and $\mathbf{A}_{p,p}$ as defined in (9) and (10) are all invertible. Additionally, $\hat{\mathbf{A}}_{\text{ext}} = \tilde{\mathbf{A}} + \mathbf{LR} \approx \mathbf{A}_{\text{ext}}$ is also invertible. If the k columns in the low-rank factor $\mathbf{L}$ and the k rows in $\mathbf{R}$ are linearly independent, then the condition number of the Woodbury operator is bounded above as follows:

$$\kappa\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right) \le \min\left\{\hat{\kappa}(\mathbf{L})^2, \hat{\kappa}(\mathbf{R})^2\right\}\kappa\left(\hat{\mathbf{A}}_{\text{ext}}\right)\kappa\left(\tilde{\mathbf{A}}\right), \tag{17}$$

where

$$\hat{\kappa}(\mathbf{L}) = \left\|\mathbf{L}^{\dagger}\right\|\|\mathbf{L}\| \text{ and } \hat{\kappa}(\mathbf{R}) = \left\|\mathbf{R}^{\dagger}\right\|\|\mathbf{R}\|$$

with

$$\mathbf{L}^{\dagger} = \left(\mathbf{L}^{T}\mathbf{L}\right)^{-1}\mathbf{L}^{T} \text{ and } \mathbf{R}^{\dagger} = \mathbf{R}^{T}\left(\mathbf{R}\mathbf{R}^{T}\right)^{-1}$$

defined as the pseudo-inverse for $\mathbf{L}$ and $\mathbf{R}$ in the standard sense.

Proof The matrix $\tilde{\mathbf{A}}$ is invertible since it is block diagonal with each block invertible by our assumption. Let the pseudo-inverses of $\mathbf{L}$ and $\mathbf{R}$ be defined as above.

It is easy to verify that $\mathbf{L}^{\ddagger}\mathbf{L} = \mathbf{R}\mathbf{R}^{\ddagger} = \mathbf{I}$ with dimension $k \times k$. By right-multiplying both sides of $\hat{\mathbf{A}}_{ext} = \tilde{\mathbf{A}} + \mathbf{LR}$ by $\tilde{\mathbf{A}}^{-1}\mathbf{L}$, we get

$$\hat{\mathbf{A}}_{ext}\tilde{\mathbf{A}}^{-1}\mathbf{L} = \left(\tilde{\mathbf{A}} + \mathbf{LR}\right)\tilde{\mathbf{A}}^{-1}\mathbf{L} = \mathbf{L} + \mathbf{LR}\tilde{\mathbf{A}}^{-1}\mathbf{L} = \mathbf{L}\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right).$$

Now the left-multiplication on both sides of the previous equality by $\mathbf{L}^{\ddagger}$ results in the following:

$$\mathbf{L}^{\dagger}\hat{\mathbf{A}}_{ext}\tilde{\mathbf{A}}^{-1}\mathbf{L} = \mathbf{L}^{\dagger}\mathbf{L}\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right) = \mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}.$$

Simplifying utilizing the basic properties of pseudo-inverse gives the following expression:

$$\mathbf{L}^{\dagger}\hat{\mathbf{A}}_{ext}^{-1}\tilde{\mathbf{A}}\mathbf{L} = \left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right)^{-1}.$$

Therefore, the condition number for the Woodbury operator is bounded above by

$$\kappa\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right) = \left\|\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right\|\left\|\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right)^{-1}\right\|$$

$$= \left\|\mathbf{L}^\dagger\hat{\mathbf{A}}_{ext}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right\|\left\|\mathbf{L}^\dagger\hat{\mathbf{A}}_{ext}^{-1}\tilde{\mathbf{A}}\mathbf{L}\right\|$$

$$\leq \left\|\mathbf{L}^\dagger\right\|^2\|\mathbf{L}\|^2\left\|\hat{\mathbf{A}}_{ext}^{-1}\right\|\left\|\hat{\mathbf{A}}_{ext}\right\|\left\|\tilde{\mathbf{A}}^{-1}\right\|\|\tilde{\mathbf{A}}\|$$

$$= \left\|\mathbf{L}^\dagger\right\|^2\|\mathbf{L}\|^2\kappa\left(\hat{\mathbf{A}}_{ext}\right)\kappa\left(\tilde{\mathbf{A}}\right).$$

A similar argument using $\mathbf{R}^\ddagger$ gives

$$\kappa\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right) \leq \left\|\mathbf{R}^\dagger\right\|^2\|\mathbf{R}\|^2\kappa\left(\hat{\mathbf{A}}_{ext}\right)\kappa\left(\tilde{\mathbf{A}}\right).$$

Combining the two bounds above gives equation (17). □

The lemma implies that if we construct $\mathbf{Q} \approx \mathbf{LR}$ so that $\hat{\mathbf{A}}_{ext} = \tilde{\mathbf{A}} + \mathbf{LR}$ is invertible, $\mathbf{L}$ and $\mathbf{R}$ are full rank, and additionally let $a^2 = \min\left\{\hat{\kappa}(\mathbf{L})^2, \hat{\kappa}(\mathbf{R})^2\right\}$, then $\kappa\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right) \leq a^2\kappa\left(\hat{\mathbf{A}}_{ext}\right)\kappa\left(\tilde{\mathbf{A}}\right)$. When the original problem and new problem have similar condition numbers, i.e., $\kappa\left(\hat{\mathbf{A}}_{ext}\right) \approx \kappa\left(\tilde{\mathbf{A}}\right) \approx \kappa$, the lemma and the low-rank approximation construction above together give the bound $\kappa\left(\mathbf{I} + \mathbf{R}\tilde{\mathbf{A}}^{-1}\mathbf{L}\right) \leq a^2\kappa^2$. The upper bound given by the lemma can be improved by building $\mathbf{L}$ and $\mathbf{R}$ so that at least one of $\hat{\kappa}(\mathbf{L})$ and $\hat{\kappa}(\mathbf{R})$ stay small. When the rank number of the low-rank factorization of $\mathbf{Q}$ ia large, a "re-compression" via constructing an ID or SVD for the concatenated $\mathbf{L}$ factor may be able to reduce the rank number and thus save some computations for the later steps. Take the refined fish test in Section 4.2 as an example, Table 14 reports the rank number of the concatenated low-rank factorization k and the one after re-compression k_{recomp} as well as the corresponding condition number of the Woodbury system. The tolerance for the ID is set to 10^{-10} and the condition numbers reported in the table are calculated via Matlab's `cond()` function.

For the problems considered in this manuscript, the upper bound in Lemma A.1 is overly pessimistic. In fact, the observed condition number is much smaller than $a^2\kappa^2$. In practice, when the low-rank approximation of $\mathbf{Q}$ is constructed with care as

Table 14 The observed rank (same as the size of the Woodbury system) and condition number for an interior BIE on the fish geometry illustrated in Fig. 14. k and κ are the size and condition number for the Woodbury operator defined for the concatenated factorization $\mathbf{Q} \approx \mathbf{LR}$. k_{recomp} and κ_{recomp} are the size and condition number for the Woodbury operator defined for the factorization with an extra SVD applied to $\mathbf{L}$ as a re-compression step. The condition numbers are all calculated by Matlab's `cond()` function. The condition number of the block-diagonal matrix $\tilde{\mathbf{A}}$ and the ELS $\mathbf{A}_{ext}$ are also reported. Finally the upper bound given by Lemma A.1 corresponding to $\mathbf{Q} \approx \mathbf{L}_{recomp}\mathbf{R}_{recomp}$ is also provided

N_k, N_c, N_p	k_{block}	κ_{block}	$k_{optimal}$	$\kappa_{optimal}$	$\kappa\left(\hat{\mathbf{A}}_{ext}\right)$	$\kappa\left(\tilde{\mathbf{A}}\right)$	Upper bound
752, 48, 384	175	1630.0	141	98.5	378.0	371.0	4.1e + 25
1520, 80, 640	158	407.3	124	77.3	376.7	371.0	1.2e + 25
3072, 128, 1024	143	523.5	113	77.0	375.0	371.0	1.6e + 25

described in Section 3.3, the observed condition number of the Woodbury system is comparable to the condition number of the original linear system. For example, Table 14 also reports the upper bound on the condition number given by Lemma A.1 for the Woodbury system. With or without the re-compression step, the condition numbers are well below the upper bound provided by the lemma.

Appendix : B. Extended system for the channel with added holes problem

Let Γ_k be the original channel boundary given in Fig. 7(a) and Γ_p be the union of the holes added in Fig. 7(b). Following this subscript notation, the discretized BIE on the "channel-with-holes" geometry can be reordered into the same format as in (10). Since no points are deleted, (10) itself serves as an ELS for this problem, and it can be written as

$$\mathbf{A}_{n,n} = \mathbf{A}_{\text{ext}} = \begin{bmatrix} \mathbf{A}_{k,k} & \mathbf{A}_{k,p} \\ \mathbf{A}_{p,k} & \mathbf{A}_{p,p} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{k,k} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{p,p} \end{bmatrix} + \mathbf{Q} \tag{18}$$

where the update matrix $\mathbf{Q}$ can be approximated by a low-rank factorization

$$\mathbf{Q} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{k,p} \\ \mathbf{A}_{p,k} & \mathbf{0} \end{bmatrix} \approx \mathbf{LR}.$$

If a fast direct solver is already constructed for the original channel geometry, i.e., an approximation to $\mathbf{A}_{k,k}^{-1}$ is available, then the solution to (18) can be quickly obtained by a Woodbury formula as described in Section 2.2. The construction of the low-rank approximation for the update matrix is also simpler for this particular problem, since only two sub-blocks need to be handled.

References

1. Marple, G., Barnett, A., Gillman, A., Veerapaneni, S.: A fast algorithm for simulating multiphase flows through periodic geometries of arbitrary shape. SIAM J. Scient. Comput. **38**(5), B740–B772 (2016)
2. Ghigliotti, G., Rahimian, A., Biros, G., Misbah, C.: Vesicle migration and spatial organization driven by flow line curvature. Phys Rev Lett **106**(2), 028101 (2011)
3. Lushi, E., Wioland, H., Goldstein, R.E.: Fluid flows created by swimming bacteria drive self-organization in confined suspensions. Proc. Natl. Acad. Sci. **111**(27), 9733–9738 (2014)
4. Yan, W., Corona, E., Malhotra, D., Veerapaneni, S., Shelley, M.: A scalable computational platform for particulate stokes suspensions. J. Comput. Phys. **416**, 109524 (2020)
5. Kabacaoğlu, G., Biros, G.: Sorting same-size red blood cells in deep deterministic lateral displacement devices. J. Fluid Mech. **859**, 433–475 (2019)

6. Wu, B., Zhu, H., Barnett, A., Veerapaneni, S.: Solution of stokes flow in complex nonsmooth 2d geometries via a linear-scaling high-order adaptive integral equation scheme. J. Comput. Phys. **410**, 109361 (2020)
7. Zhang, Y., Gillman, A.: A fast direct solver for boundary value problems on locally perturbed geometries. J. Comput. Phys. **356**, 356–371 (2018)
8. Zhang, Y., Gillman, A.: An alternative extended linear system for boundary value problems on locally perturbed geometries, vol. 433 (2021)
9. Martinsson, P., Rokhlin, V.: A fast direct solver for boundary integral equations in two dimensions. J. Comput. Phys. **205**(1), 1–23 (2005)
10. Gillman, A., Young, P., Martinsson, P.: A direct solver O(N) complexity for integral equations on one-dimensional domains. Front. Math. China **7**, 217–247 (2012)
11. Sheng, Z., Dewilde, P., Chandrasekaran, S: Algorithms to solve hierarchically semi-separable systems (2007)
12. Chandrasekaran, S., Gu, M.: A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices. Numer. Math. **96**(4), 723–731 (2004)
13. Ho, K., Ying, L.: Hierarchical interpolative factorization for elliptic operators: Integral equations. Commun. Pure Appl. Math. **69**(7), 1314–1353 (2015)
14. Hackbusch, W.: A sparse matrix arithmetic based on h-matrices. part i: Introduction to h-matrices. Computing **62**, 89–108, 04 (1999)
15. Minden, V., Damle, A., Ho, K.L., Ying, L.: A technique for updating hierarchical skeletonization-based factorizations of integral operators. Multiscale Model. Simul. **14**(1), 42–64 (2016)
16. Ryan, J.P., Damle, A.: Parallel skeletonization for integral equations in evolving multiply-connected domains. SIAM J. Sci. Comput. **43**(3), A2320–A2351 (2021)
17. Coulier, P., Pouransari, H., Darve, E.: The inverse fast multipole method: Using a fast approximate direct solver as a preconditioner for dense linear systems. SIAM J. Sci. Comput. **39**(3), A761–A796 (2017)
18. Bebendorf, M.B.M., Bratsch, M.: On the spectral equivalence of hierarchical matrix preconditioners for elliptic problems. Math. Comput. **85**, 2839–2861 (2016)
19. Chandrasekaran, S., Dewilde, P., Gu, M., Somasunderam, N.: On the numerical rank of the off-diagonal blocks of schur complements of discretized elliptic pdes. SIAM J. Matrix Anal. Appl. **31**(5), 2261–2290 (2010)
20. Bebendorf, M.: Efficient inversion of the galerkin matrix of general second-order elliptic operators with nonsmooth coefficients. Math. Comput. **74**(251), 1179–1199 (2005)
21. Bebendorf, M., Hackbusch, W.: Existence of ${\mathscr{H}},$H,-matrix approximants to the inverse fe-matrix of elliptic operators with $l^{\infty }$l∞-coefficients. Numer. Math. **95**, 1–28 (2003)
22. Barnett, A., Wu, B., Veerapaneni, S.: Spectrally accurate quadratures for evaluation of layer potentials close to the boundary for the 2d stokes and laplace equations. SIAM J. Scientif. Comput. **37**(4), B519–B542 (2015)
23. Hsiao, G.C., Wendland, W.L.: Boundary integral equations. Springer (2008)
24. Biros, G., Ying, L., Zorin, D.: A fast solver for the stokes equations with distributed forces in complex geometries. J. Comput. Phys. **193**(1), 317–348 (2004)
25. Gillman, A., Barnett, A.: A fast direct solver for quasi-periodic scattering problems. J. Comput. Phys. **248**, 309–322 (2013)
26. Zhang, Y., Gillman, A.: A fast direct solver for two dimensional quasi-periodic multilayered medium scattering problems. CoRR, 1907.06223 (2019)
27. Saad, Y., Schultz, M.H.: Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**(3), 856–869 (1986)
28. Hao, S., Barnett, A.H., Martinsson, P.G., Young, P.: High-order accurate Nystrom discretization of integral equations with weakly singular kernels on smooth curves in the plane. Adv. Comput. Math. **40**, 245–272 (2013)
29. Alpert, B.K.: Hybrid gauss-trapezoidal quadrature rules. SIAM J. Sci. Comput. **20**, 1551–1584 (1999)
30. Helsing, J., Ojala, R.: Corner singularities for elliptic problems: Integral equations, graded meshes, quadrature, and compressed inverse preconditioning. J. Comput. Phys. **227**, 8820–8840 (2008)
31. Kapur, S., Rokhlin, V.: High-order corrected trapezoidal quadrature rules for singular functions. SIAM J. Numer. Anal. **34**(4), 1331–1356 (1997)

32. Guo, H., Zhu, H., Veerapaneni, S.: Simulating cilia-driven mixing and transport in complex geometries. Phys. Rev. Fluids **5**, 053103 (2020)
33. Beylkin, D., Rokhlin, V.: Fitting a bandlimited curve to points in a plane. SIAM J. Sci. Comput. **36**(3), A1048–A1070 (2014)
34. Golub, G.H., Van Loan, C.F.: Matrix Computations. The Johns Hopkins University Press third ed. (1996)
35. Yip, E.L.: A note on the stability of solving a rank-p modification of a linear system by the sherman–morrison–woodbury formula. SIAM J. Sci. Stat. Comput. **7**(2), 507–513 (1986)