NOTE: We colored the changes from the original manuscript as follows:

- **Red:** Changes due to updating the numerical benchmarks (Section 4), using an optimized computer implementation; see Page 1 of “Reply to review report”.
- **Blue:** Changes due the comments from Reviewer #1.
- **Green:** Changes due the comments from Reviewer #2.
- **Magenta:** Changes due the comments from Reviewer #3.
- **Orange:** Changes due the comments from Reviewer #4.

Application of the inverse fast multipole method as a preconditioner in a 3D Helmholtz boundary element method

Toru Takahashi^{a,*}, Pieter Coulier^{b,c}, Eric Darve^c

^aDepartment of Mechanical Science and Engineering, Nagoya University, Furo-cho, Nagoya, Aichi, 464-8603 Japan

^bDepartment of Civil Engineering, KU Leuven, Kasteelpark Arenberg 40, 3001 Leuven, Belgium

^cDepartment of Mechanical Engineering, Stanford University, 496 Lomita Mall, 94305, Stanford, CA, USA

Abstract

We investigate an efficient preconditioning of iterative methods (such as GMRES) for solving dense linear systems $\mathbf{Ax} = \mathbf{b}$ that follow from a boundary element method (BEM) for the 3D Helmholtz equation, focusing on the low-frequency regime. While matrix-vector products in GMRES can be accelerated through the low-frequency fast multipole method (LFFMM), the BEM often remains computationally expensive due to the large number of GMRES iterations. We propose the application of the inverse fast multipole method (IFMM) as a preconditioner to accelerate the convergence of GMRES. The IFMM is in essence an approximate direct solver that uses a multilevel hierarchical decomposition and low-rank approximations. The proposed IFMM-based preconditioning has a tunable parameter ε that balances the cost to construct a preconditioner $\mathbf{M}$, which is an approximation of $\mathbf{A}^{-1}$, and the cost to perform the iterative process by means of $\mathbf{M}$. Namely, using a small (respectively, large) value of ε takes a long (respectively, short) time to construct $\mathbf{M}$, while the number of iterations can be small (respectively, large). A comprehensive set of numerical examples involving various boundary value problems with complicated geometries and mixed boundary conditions is presented to validate the efficiency of the proposed method. We show that the IFMM preconditioner (with a nearly optimal ε of 10^{-2}) clearly outperforms some common preconditioners for the BEM, achieving 1.2–10.8 times speed-up of the computations, in particular when the scale of the underlying scatterer is about five wavelengths or more. In addition, the IFMM preconditioner is capable of solving complicated problems (in a reasonable amount of time) that BD preconditioner can not.

Keywords: Fast Multipole Method, Boundary Element Method, Inverse Fast Multipole

10 1. Introduction

11 In the context of the boundary element method (BEM) [1, 2], it is becoming popular to
12 use iterative methods to solve discretized boundary integral equations, i.e., a linear system of
13 equations, say $\mathbf{Ax} = \mathbf{b}$, because the matrix-vector products $\mathbf{Ax}$ can be computed quickly by
14 using the fast multipole method (FMM) [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] or theoretically
15 related methods based on $\mathcal{H}$ -matrices [15]. However, the development of robust, computationally
16 inexpensive, implementation-easy, and/or general-purpose preconditioners (say, $\mathbf{M}$ such that
17 $\mathbf{M} \simeq \mathbf{A}^{-1}$) is still an open research question. This challenge is addressed in the current paper.

18 We briefly review recent developments of preconditioners in the context of FMM-accelerated
19 BEM (FMBEM) or methods of moment (MoM) for oscillatory problems, involving the Helmholtz
20 equation as well as the Maxwell and Navier-Cauchy equations. A more comprehensive review
21 (especially for the Maxwell equations) is found in [16]. General preconditioning techniques [17]
22 (e.g., point-Jacobi, ILU, etc.) can be applied to BEM matrices and in some cases can lead to
23 a marked improvement (e.g., see [18] for ILU). In the context of the FMM, better convergence
24 than with a simple point-Jacobi approach is usually obtained by means of a block-Jacobi preconditioner,
25 which is based on the diagonal blocks of $\mathbf{A}$ that are associated with the leaf clusters, and
26 widely used in the field of FMBEM [13]. These diagonal and block-diagonal preconditioners
27 can be considered as a special type of sparse approximate inverse (SPAI or SAI) preconditioners.

28 More general SPAI preconditioners $\mathbf{M} \simeq \mathbf{A}^{-1}$ are obtained by minimizing $\|\mathbf{I} - \mathbf{MA}\|_{\text{F}}^2$ (where
29 $\|\cdot\|_{\text{F}}$ stands for the Frobenius norm), assuming a certain predefined sparsity pattern in $\mathbf{M}$ so that
30 its application is inexpensive. Such SPAI preconditioners can be efficient in some cases (e.g.,
31 see [19, 20, 21, 22]), but their efficiency tends to deteriorate as the problem size increases,
32 because a SPAI preconditioner essentially exploits only local (neighboring) rather than global
33 information of the dense matrix [20]. The aforementioned preconditioners are classified as algebraic
34 preconditioners in the sense that they directly utilize the matrix data. Strictly speaking, the
35 block-diagonal preconditioner utilizes information of the leaf clusters and can also be seen as a
36 kind of geometric preconditioning.

37 Geometric preconditioners exploit additional information specific to the geometry of the underlying
38 problem and can be more special-purpose. For example, it is reported that scaling with
39 the edge lengths of RWG bases [23] is effective in the case of three-dimensional (3D) Maxwell
40 equations [24].

41 For some specific cases (i.e., for certain choices of integral equations and boundary conditions),
42 analytic preconditioners have been developed to improve the spectral property of the
43 hypersingular integral operator, which should be handled in order to avoid fictitious (spurious)
44 eigenvalues in external problems. Very efficient preconditioners based on the Calderón relation
45 have been proposed by Steinbach et al [25]. It is mentioned in [25] that Calderón-based preconditioners
46 can handle not only Dirichlet and Neumann problems but also mixed boundary value
47 problems (BVPs) by some modifications. However, this has not been investigated in the literature
48 at this point. The recent progress on this class of preconditioners is mentioned in [26].

---

*Corresponding author

Email address: ttaka@nuem.nagoya-u.ac.jp (Toru Takahashi)

Some of the other analytic preconditioning strategies that can be incorporated in the FMM use the pseudo-inverse of the hypersingular operator [27, 28, 29].

Another preconditioning technique is the inner-outer scheme, which exploits a fast matrix-vector product (typically, the FMM) in two levels of precisions. Since the left-preconditioning, which the present study chooses, is represented as $\mathbf{M}\mathbf{A}\mathbf{x} = \mathbf{M}\mathbf{b}$, we have to apply $\mathbf{M}$ to a certain vector $\mathbf{r}$, where $\mathbf{r} := \mathbf{A}\mathbf{x}$ during the iterative process and $\mathbf{r} := \mathbf{b}$ at the initial stage. Computing $\mathbf{M}\mathbf{r}$ means solving the equation $(\mathbf{A}\mathbf{z} \simeq) \mathbf{M}^{-1}\mathbf{z} = \mathbf{r}$ with respect to a vector $\mathbf{z}$. We never solve this equation or compute $\mathbf{M}$ directly. Instead, we solve the equation with a fast iterative solver, which is typically GMRES [30] employing the FMM. Therefore, we exploit such a fast iterative solver to solve $\mathbf{M}^{-1}\mathbf{z} = \mathbf{r}$ within another fast iterative solver (typically, GMRES using the FMM) to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$. The former and latter solvers are called as inner and outer solvers, respectively. The inner solution should be computed faster and thus may be less accurate than the outer solution. To this end, a relatively large stopping criterion (tolerance) and/or low-accurate setting (small number of expansion orders in the case of the FMM) is used for the inner solver. This type of preconditioning was proposed in [31] and then applied to some cases [20, 32, 33, 22]. This approach is often combined with the flexible GMRES [34].

In the aforementioned inner-outer scheme, fast direct solvers (e.g., direct FMMs [35, 36, 37, 38], $\mathcal{H}$ -LU factorization [15, 39], and inverse FMM [40, 41]) can be also applied to solve $\mathbf{M}^{-1}\mathbf{z} = \mathbf{r}$. As the term “direct” indicates, these solvers perform Gaussian elimination or LU factorization on an approximated (compressed) representation of $\mathbf{A}$, without any iterative procedure. This allows solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ approximately with a computational complexity that is typically $O(N \log N)$ or even $O(N)$, which is significantly less than the $O(N^3)$ complexity for a classical Gaussian elimination on a dense matrix. These direct solvers work well at high accuracy. On the other hand, at low accuracy, they can be used as preconditioners or as the aforementioned inner solvers.

One such fast direct solver that can be used as a preconditioner (in the inner-outer scheme) is the inverse FMM (IFMM), which was recently presented in [40, 41]. While the FMM computes the matrix-vector product $\mathbf{A}\mathbf{x}$ in $O(N)$ operations, the IFMM uses the hierarchical structure and low-rank representations to find an approximation $\mathbf{M}\mathbf{r}$ in $O(N)$. To date, the IFMM has already been successfully applied as a preconditioner in a variety of problems (including radial basis function interpolation [42], Stokes flow in porous media [43], etc.), illustrating that the number of iterations and the total computation time can be reduced considerably in comparison with some commonly used preconditioners. Similarly, $\mathcal{H}$ -LU factorization has been used successfully to compute $\mathbf{M}$ in both low- and high-frequency regimes for the 2D Helmholtz equation [44]. Note that the IFMM is similar to $\mathcal{H}$ -LU but different. Namely, $\mathcal{H}$ -LU is based on a block LU factorization, while the IFMM is based on a multilevel elimination. Formally, both methods use the FMM tree. However, they eliminate the nodes of the tree in a different order, i.e., depth-first for $\mathcal{H}$ -LU and level by level for IFMM.

The purpose of this paper is to apply the IFMM as a preconditioner in GMRES for solving a dense linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ that follows from a BEM for the 3D Helmholtz equation. Here, the matrix-vector products required by GMRES are accelerated with the low-frequency FMM (LFFMM) [11]. Note that, as the name implies, the LFFMM is computationally efficient (truly a fast method) when the size of the model is comparable to the wavelength of the field. The application of low-frequency acoustic analysis includes simulations of acoustic metamaterials [45], which have recently drawn much attention in physical science and engineering because they possess unconventional acoustical properties, e.g., negative refractive index. In the numerical tests in Section 4, we consider models (scatterers) whose sizes are about ten wavelengths or less.

We emphasize that the present study focusing on the Helmholtz equation offers special difficulties. First, we should point out that there are very few robust and black-box preconditioners for general indefinite systems. For example, the multigrid method can work in some cases but generally has some great limitations and assumptions [46]. Therefore this paper is an essential contribution. Second, the present study is not just a repeat of the references [40, 41]. The issue of low-rank compression, on which both the IFMM and $\mathcal{H}$ -matrix methods are based, is still poorly understood theoretically. However we know that in the high frequency regime, $\mathcal{H}$ -matrix methods fail in a general setting [47]. This made it unclear before the start of this study whether the method would work, if at all, in the low and mid frequency regime. The presented benchmarks however show that the method works better than expected.

To exploit the IFMM as a preconditioner for our purpose, we express all the operations of the LFFMM, i.e., the P2M, M2M, M2L, L2L, L2P, and P2P operators, in matrix format. Once this is done, we can adapt an existing implementation of the IFMM [41, 42] to incorporate these LFFMM operators. The theoretical background of this paper hence resembles that of previous papers [40, 41], although the latter have mainly considered discrete models (particle simulations) rather than boundary integral equations. We will therefore focus on the numerical aspects of applying the IFMM.

This paper is organized as follows. Section 2 introduces the BEM for the 3D Helmholtz equation and the LFFMM. We also express the LFFMM operators in matrix format. In Section 3, these expressions are integrated into the IFMM, and a concise overview of the IFMM algorithm is given. Section 4 presents the numerical results to validate the performance of the IFMM as a preconditioner in comparison with other preconditioners, for various BVPs involving complicated geometries. Section 5 concludes the present study, with multiple directions for future research.

2. Fast-multipole accelerated BEM for 3D acoustics

2.1. Problem statement and boundary integral equation

Let D be a domain (or scatterer) in $\mathbb{R}^3$ and S be its boundary which is assumed to be piecewise smooth. Then, we consider the following external boundary value problem (BVP) or scattering problem for the 3D Helmholtz equation in D :

$$\Delta u + k^2 u = 0 \quad \text{in } D \subset \mathbb{R}^3, \quad (1a)$$

$$u = \bar{u} \quad \text{on } S_u, \quad (1b)$$

$$q = \bar{q} \quad \text{on } S_q, \quad (1c)$$

$$u(\mathbf{x}) \rightarrow u^I(\mathbf{x}) \quad \text{as } |\mathbf{x}| \rightarrow \infty, \quad (1d)$$

where a real number k denotes a given wavenumber, $q := \frac{\partial u}{\partial n}$ denotes the flux, i.e., the normal derivative of the (total) field u , and u^I denotes the incident field. Also, u (respectively, q) is given on S_u (S_q) as $\bar{u}$ ($\bar{q}$) as the boundary condition, where $S = S_u \cup S_q$ and $S_u \cap S_q = \emptyset$. The above BVP is reduced to solving the following combined boundary integral equation (CBIE) [48]:

$$\begin{aligned} & \frac{1}{2}u(\mathbf{x}) + \int_S \left(\frac{\partial \Gamma}{\partial n_y}(\mathbf{x}, \mathbf{y})u(\mathbf{y}) - \Gamma(\mathbf{x} - \mathbf{y})q(\mathbf{y}) \right) dS_y \\ & + \beta \left\{ \frac{1}{2}q(\mathbf{x}) + \int_S \left(\frac{\partial^2 \Gamma}{\partial n_x \partial n_y}(\mathbf{x}, \mathbf{y})u(\mathbf{y}) - \frac{\partial \Gamma}{\partial n_x}(\mathbf{x}, \mathbf{y})q(\mathbf{y}) \right) dS_y \right\} \\ & = u^I(\mathbf{x}) + \beta q^I(\mathbf{x}) \quad \text{for } \mathbf{x} \in S, \end{aligned} \quad (2)$$

where $\Gamma(\mathbf{x}) := \frac{e^{ik|\mathbf{x}|}}{4\pi|\mathbf{x}|}$ is the fundamental solution of the Helmholtz equation, and $q^l(= \frac{\partial u^l}{\partial n})$ represents the normal derivative of the incident field. Also, the coupling coefficient β must be a complex number whose imaginary part is non-zero so that the CBIE is free from the fictitious (spurious) eigenfrequency problem [48]. In this study, we let $\beta = \frac{i}{k}$, which is the standard choice [49].

2.2. Discretization

For simplicity, we discretize the CBIE in Eq. (2) with N triangular piecewise-constant elements, i.e., u and q are assumed to be constants u_i and q_i , respectively, on the i^{th} element S_i ¹. Then, the CBIE is reduced to the following algebraic system of linear equations:

$$\mathbf{Ax} = \mathbf{b}, \quad (3)$$

where $\mathbf{A} \in \mathbb{C}^{N \times N}$ is the coefficient matrix², $\mathbf{x} \in \mathbb{C}^N$ is the unknown vector whose i^{th} component is either the unknown u_i or q_i according to the boundary condition on S_i , and $\mathbf{b} \in \mathbb{C}^N$ is a known vector whose i^{th} component is determined by either the known u_i or q_i as well as the incident field $u^l(\mathbf{x}_i) + \beta q^l(\mathbf{x}_i)$ at $\mathbf{x}_i$. Here, $\mathbf{x}_i$ denotes the i^{th} collocation point located at the center of S_i .

Since $\mathbf{A}$ is usually non-symmetric and dense, as N becomes larger, it becomes harder to solve Eq. (3) by means of a direct solver (Gaussian elimination, LU factorization, etc.) owing to its $O(N^3)$ computational cost.

2.3. Matrix-vector representations of LFFMM operations

The details of the (multilevel) LFFMM in terms of the boundary element method for the 3D Helmholtz equation are well described in the literature [7, 11, 13, 52]. The key formulae of the LFFMM, i.e., the P2M, M2M, M2L, L2L, L2P, and P2P operators are presented in Appendix A. In this subsection, we transform these analytical formulae to an algebraic form, i.e., matrix-vector products, in order to incorporate the LFFMM into the IFMM [41] in Section 3.

Each FMM operation can be represented as the following matrix-vector products. Following the reference [53], let us consider the matrix representation of the M2L operation in Eq. (A.4)

¹It is possible to use high-order elements (e.g., linear, quadratic [2], or isogeometric elements [50]) together with the IFMM as well as LFFMM. The IFMM as well as LFFMM is nearly unchanged. However, we avoided some complicated mathematical/numerical procedures resulting from a high-order discretization (e.g., treatments of singular integrals and corners on a boundary) by using the piecewise-constant elements. A drawback of using piecewise-constant elements is that the discretization error usually converges slowly, which was actually observed in our numerical tests; see Subsection 4.2.

²Considering the singularity as well as nearly-singularity, we use the add-and-subtract technique in the evaluation of the boundary integrals over each element. Namely, we first represent the boundary integrals in Eq. (2) in terms of an element S_j as follows:

$$\begin{aligned} \int_{S_j} \frac{\partial \Gamma}{\partial n_x}(\mathbf{x}, \mathbf{y}) dS_y &= \int_{S_j} \left(\frac{\partial \Gamma}{\partial n_x}(\mathbf{x}, \mathbf{y}) - \frac{\partial G}{\partial n_x}(\mathbf{x}, \mathbf{y}) \right) dS_y + \int_{S_j} \frac{\partial G}{\partial n_x}(\mathbf{x}, \mathbf{y}) dS_y, \\ \int_{S_j} \frac{\partial^2 \Gamma}{\partial n_x \partial n_y}(\mathbf{x}, \mathbf{y}) dS_y &= \int_{S_j} \left(\frac{\partial^2 \Gamma}{\partial n_x \partial n_y}(\mathbf{x}, \mathbf{y}) - \frac{\partial^2 G}{\partial n_x \partial n_y}(\mathbf{x}, \mathbf{y}) - \frac{(ik)^2}{8\pi} \frac{\partial^2 |\mathbf{x} - \mathbf{y}|}{\partial n_x \partial n_y} \right) dS_y \\ &\quad + \int_{S_j} \left(\frac{\partial^2 G}{\partial n_x \partial n_y}(\mathbf{x}, \mathbf{y}) + \frac{(ik)^2}{8\pi} \frac{\partial^2 |\mathbf{x} - \mathbf{y}|}{\partial n_x \partial n_y} \right) dS_y, \end{aligned}$$

where $G(\mathbf{x}) := \frac{1}{4\pi|\mathbf{x}|}$ denotes the fundamental solution of the Laplace equation. Then, since the integrands in the first terms of the right-hand sides (RHSs) are always non-singular (even when $\mathbf{x} = \mathbf{y}$), the first terms are evaluated numerically (we use four points triangular Gaussian quadrature [51]). On the other hand, the second terms are integrated analytically.

in the case that the expansion order p (see Eq. (6)) is 2, for example. Then, we can write down the M2L operation explicitly as follows:

$$\begin{aligned}\ell_0^0 &= \sum_{n'=0}^1 \sum_{m'=-n'}^{n'} K_{0,0,n',m'} m_{n'}^{m'} = K_{0,0,0,0} m_0^0 + K_{0,0,1,-1} m_1^{-1} + K_{0,0,1,0} m_1^0 + K_{0,0,1,1} m_1^1, \\ \ell_1^{-1} &= \sum_{n'=0}^1 \sum_{m'=-n'}^{n'} K_{1,-1,n',m'} m_{n'}^{m'} = K_{1,-1,0,0} m_0^0 + K_{1,-1,1,-1} m_1^{-1} + K_{1,-1,1,0} m_1^0 + K_{1,-1,1,1} m_1^1, \\ \ell_1^0 &= \sum_{n'=0}^1 \sum_{m'=-n'}^{n'} K_{1,0,n',m'} m_{n'}^{m'} = K_{1,0,0,0} m_0^0 + K_{1,0,1,-1} m_1^{-1} + K_{1,0,1,0} m_1^0 + K_{1,0,1,1} m_1^1, \\ \ell_1^1 &= \sum_{n'=0}^1 \sum_{m'=-n'}^{n'} K_{1,1,n',m'} m_{n'}^{m'} = K_{1,1,0,0} m_0^0 + K_{1,1,1,-1} m_1^{-1} + K_{1,1,1,0} m_1^0 + K_{1,1,1,1} m_1^1.\end{aligned}$$

Clearly, the operation can be represented by a matrix-vector product $\mathbf{z} = \mathbf{K}\mathbf{y}$, where

$$\mathbf{K} := \begin{bmatrix} K_{0,0,0,0} & K_{0,0,1,-1} & K_{0,0,1,0} & K_{0,0,1,1} \\ K_{1,-1,0,0} & K_{1,-1,1,-1} & K_{1,-1,1,0} & K_{1,-1,1,1} \\ K_{1,0,0,0} & K_{1,0,1,-1} & K_{1,0,1,0} & K_{1,0,1,1} \\ K_{1,1,0,0} & K_{1,1,1,-1} & K_{1,1,1,0} & K_{1,1,1,1} \end{bmatrix}, \quad \mathbf{y} := (m_0^0, m_1^{-1}, m_1^0, m_1^1)^T, \quad \mathbf{z} := (\ell_0^0, \ell_1^{-1}, \ell_1^0, \ell_1^1)^T.$$

In the same way, by regarding a pair of indices n and m (respectively, n' and m') as the “row” index $i := n^2 + n + m \in [0, p^2)$ (respectively, the “column” index $j := n'^2 + n' + m' \in [0, p^2)$), all the FMM operations can be represented as the following matrix-vector products:

$$\text{P2M} : \quad \mathbf{y}_i^{(l)} = \mathbf{V}_i^{(l)*} \mathbf{x}_i^{(l)}, \quad (4a)$$

$$\text{M2M} : \quad \mathbf{y}_{\mathbf{p}(i)}^{(l-1)} = \mathbf{V}_i^{(l-1)*} \mathbf{y}_i^{(l)}, \quad (4b)$$

$$\text{M2L} : \quad \mathbf{z}_i^{(l)} = \mathbf{K}_{ij}^{(l)} \mathbf{y}_j^{(l)} \quad (\text{where } \Omega_j^{(l)} \in \mathcal{I}(\Omega_i^{(l)})), \quad (4c)$$

$$\text{L2L} : \quad \mathbf{z}_i^{(l)} = \mathbf{U}_i^{(l-1)} \mathbf{z}_{\mathbf{p}(i)}^{(l-1)}, \quad (4d)$$

$$\text{L2P} : \quad \mathbf{f}_i^{(l)} = \mathbf{U}_i^{(l)} \mathbf{z}_i^{(l)}, \quad (4e)$$

$$\text{P2P} : \quad \mathbf{f}_i^{(l)} = \mathbf{S}_{ij}^{(l)} \mathbf{x}_j^{(l)} \quad (\text{where } \Omega_j^{(l)} \in \mathcal{N}(\Omega_i^{(l)})), \quad (4f)$$

where $\Omega_i^{(l)}$ denotes a cluster i at a level l , $\mathbf{x}_i^{(l)}$ consists of candidates (or an initial guesses) of the unknown values on the elements belonging to $\Omega_i^{(l)}$, $\mathcal{I}$ and $\mathcal{N}$ denote the interaction and neighbor lists, respectively (see their definitions in Appendix A), and $*$ denotes the adjoint operator. Also, $\mathbf{y}_i^{(l)}$ and $\mathbf{z}_i^{(l)}$ are p^2 -dimensional vectors consisting of the moments m_n^m and local-coefficients ℓ_n^m of $\Omega_i^{(l)}$, respectively, i.e.,

$$\begin{aligned}\mathbf{y}_i^{(l)} &= \left(\underbrace{m_0^0}_{n=0}, \underbrace{m_1^{-1}, m_1^0, m_1^1}_{n=1}, \dots, \underbrace{m_{p-1}^{-(p-1)}, \dots, m_{p-1}^{p-1}}_{n=p-1} \right)^T \in \mathbb{C}^{p^2}, \\ \mathbf{z}_i^{(l)} &= \left(\underbrace{\ell_0^0}_{n=0}, \underbrace{\ell_1^{-1}, \ell_1^0, \ell_1^1}_{n=1}, \dots, \underbrace{\ell_{p-1}^{-(p-1)}, \dots, \ell_{p-1}^{p-1}}_{n=p-1} \right)^T \in \mathbb{C}^{p^2}.\end{aligned}$$

Also, a matrix (or block in the terminology of the IFMM) $\mathbf{K}_{ij}^{(l)}$ corresponds to the M2L operator between two clusters $\Omega_i^{(l)}$ and $\Omega_j^{(l)}$. A matrix $\mathbf{V}_i^{(l)*}$ corresponds to the P2M operator if $\Omega_i^{(l)}$ is a leaf cluster, or the M2M operator if not. Similarly, $\mathbf{U}_i^{(l)}$ corresponds to the L2P or L2L operator. The mapping $p(i)$ denotes the parent cluster of i .

The matrix representations in Eq. (4) make it easy to apply the IFMM to the LFFMM because the IFMM handles each FMM operator algebraically, as will be explained in Section 3.

3. The inverse fast multipole method (IFMM)

The IFMM [40, 41] is a fast approximate direct method to solve a linear system of equations that results from, for example, discretizing a boundary integral equation, as the present paper considers. It uses the same multilevel hierarchical format and low-rank approximations as the (forward) FMM, but rather than accelerating matrix-vector products, it provides an approximate solution $\hat{\mathbf{x}} \simeq \mathbf{A}^{-1}\mathbf{b}$. The key ideas of the algorithm are concisely described in the following paragraphs; the reader is referred to [40, 41] for a detailed discussion. Note that the current IFMM implementation uses its own (uniform) tree structure, rather than the adaptive one of the FMM.

The first key idea of the IFMM is to transform the original dense system $\mathbf{Ax} = \mathbf{b}$ into an extended sparse system, using the P2M, M2M, M2L, L2L, L2P, and P2P operators described in Subsection 2.3. The equivalent sparse system contains auxiliary “unknown variables”, i.e., the multipole moments ($\mathbf{y}^{(3)}$ and $\mathbf{y}^{(2)}$) and local coefficients ($\mathbf{z}^{(3)}$ and $\mathbf{z}^{(2)}$) defined before³. More precisely, rearranging the relations defined in Eq. (4) provides the following sparse system:

$$\begin{bmatrix} \mathbf{S}^{(3)} & \mathbf{U}^{(3)} & & & \\ & \mathbf{V}^{(3)*} & & & \\ & & -\mathbf{I} & & \\ & & -\mathbf{I} & \mathbf{K}^{(3)} & \mathbf{U}^{(2)} \\ & & & \mathbf{V}^{(2)*} & \\ & & & & -\mathbf{I} & \mathbf{K}^{(2)} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{(3)} \\ \mathbf{z}^{(3)} \\ \mathbf{y}^{(3)} \\ \mathbf{z}^{(2)} \\ \mathbf{y}^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{f}^{(3)} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad (5)$$

where the vector $\mathbf{x}^{(3)}$ consists of all the 8^3 unknown subvectors for level 3, i.e., $\mathbf{x}^{(3)} := (\mathbf{x}_1^{(3)}, \dots, \mathbf{x}_{512}^{(3)})^T \in \mathbb{C}^N$, which is equivalent to $\mathbf{x}$ in Eq. (3) except for the order of elements. Similarly, $\mathbf{f}^{(3)}$ is equivalent to $\mathbf{b}$ in Eq. (3). The vectors $\mathbf{y}^{(l)}$ and $\mathbf{z}^{(l)}$ consist of all the moments and local-coefficients at level l , e.g., $\mathbf{y}^{(2)} := (\mathbf{y}_1^{(2)}, \dots, \mathbf{y}_{64}^{(2)})^T$. Correspondingly, the matrices $\mathbf{S}^{(l)}$, $\mathbf{K}^{(l)}$, $\mathbf{U}^{(l)}$, and $\mathbf{V}^{(l)*}$ are defined with $\mathbf{S}_{ij}^{(l)}$, $\mathbf{K}_{ij}^{(l)}$, $\mathbf{U}_i^{(l)}$, and $\mathbf{V}_i^{(l)*}$, respectively, for all the relevant clusters $\Omega_i^{(l)}$ and/or $\Omega_j^{(l)}$. Also, $\mathbf{I}$ corresponds to the identity matrix. The variables and equations are ordered in such a way that a sparse matrix with a symmetric fill-in pattern is obtained. The coefficient matrix in Eq. (5), denoted by $\mathbf{E}$, is moderately larger than the dense matrix $\mathbf{A}$, but sparse, making it more attractive from a computational point of view. The idea of extended sparsification has been applied before, see for example [54, 55, 56].

Note that Eq. (5) is an approximation of Eq. (3) because the number (highest order) of the multipole-moments and local-coefficients is truncated to a finite number. The very first step of the IFMM, referred to as the initialization, thus consists of nothing more than generating the P2M, M2M, M2L, L2L, L2P, and P2P operators, and creating the sparse structure of $\mathbf{E}$.

³We consider only two levels, i.e., the 2nd and 3rd (leaf) levels, for illustrative purposes, but we can consider multiple levels in a recursive manner.

Transforming the original dense system into an extended sparse system of Eq. (5) lowers the computational cost from $O(N^3)$ to $O(N^2)$, but not to $O(N)$. The quadratic cost comes from the fact that the system does not remain sparse when using a sparse direct solver. The second key idea of the IFMM is therefore to maintain the sparsity pattern of $\mathbf{E}$ throughout the Gaussian elimination; this process is also referred to as the factorization. To this end, newly created fill-in matrix blocks are approximated and redirected by means of low-rank compressions (e.g., singular value decomposition, adaptive cross approximation, etc.), provided that they represent far-field interactions. More precisely, the factorization creates interactions (i.e., fill-in matrix blocks) between clusters that are well-separated (e.g., P2P, P2L, or M2P operators⁴), implying that these newly created interactions can be treated at the parent level rather than at the current level of the tree. To this end, a low-rank compression of the fill-in is computed. For example, one can represent a newly created P2L (respectively, M2P) fill-in block as a product of two conventional P2M and M2L low-rank blocks (respectively, M2L and L2P). This resembles the decomposition of a direct interaction (a P2P operator for far-field) into the product of P2M, M2L, and L2P operators in the FMM. The fill-in can then be accounted for by updating the existing low-rank operators in $\mathbf{E}$, without the need of sacrificing its sparsity pattern. Maintaining the sparsity is a key ingredient for obtaining a fast factorization, and hence, a fast solver.

It is worth pointing out that the proposed factorization methodology is related to that of ILU preconditioners [17, Chapter 10] in the sense that the fill-in is controlled to avoid that the matrix becomes dense again. However, most of the fill-in in ILU (below a certain tolerance) is simply discarded, whereas here it is approximated using low-rank blocks, with a prescribed tolerance ε . One can intuitively understand that this will lead to a more accurate solver than ILU, as useful information is not discarded but approximated.

It can be demonstrated that the IFMM has an asymptotic complexity of $O(N \log^2 1/\varepsilon)$, under the assumption that the rank of the fill-in blocks remains bounded (which is indeed the case in many practical applications). More specifically, the singular values of these blocks need to decay faster than a^n , where $0 < a < 1$ and n denotes the singular value index.

Performing low-rank compressions throughout the factorization makes the IFMM inexact, but the accuracy can be tuned by varying the parameter ε . Although machine accuracy can be reached if needed, the computational cost of obtaining a highly accurate direct solver often becomes prohibitive. Our investigations confirm that using the IFMM as a stand-alone direct solver is not beneficial (see Subsection 4.2). It is therefore in many cases more efficient to construct an IFMM solver with a low accuracy and to apply it as a preconditioner in an (FMM-accelerated) iterative method such as GMRES [30]. This leads to a rather cheap preconditioner that is still accurate enough to reduce the number of GMRES iterations significantly, and hence the overall computation time.

The major difference between the current and previous studies [40, 41, 42] is that the underlying PDE of the problem is different, i.e., the IFMM has not been applied yet to solve the Helmholtz equation. The current study also differs from others in the way that the initial matrix operators in the IFMM are constructed. While previous studies have employed a black-box approach using interpolation based on Chebyshev polynomials (with real arithmetic) [57], the current paper considers the LFFMM operators (with complex arithmetic) introduced in Subsection 2.3 for initializing the IFMM. More details about the IFMM implementation for the current investigation can be found in Appendix B.

---

⁴A P2L operator is an unconventional FMM operator created during the factorization, and is defined as the operator that translates an unknown vector $\mathbf{x}_j^{(l)}$ to a local-coefficient $\mathbf{z}_i^{(l)}$; $i \neq j$. An M2P operator is defined similarly.

4. Numerical benchmarks

We assess the efficiency of the proposed IFMM preconditioner, starting from a simple test and moving on towards more complicated ones. The IFMM preconditioner as well as conventional preconditioners (see Subsubsection 4.1.1) are integrated into a BEM code based on GMRES employing the LFFMM.

The BEM code is partially parallelized. Namely, the FMM (i.e., the fast matrix-vector product routine) is parallelized with OpenMP; in both upward and downward passes, the loop over clusters at each level is parallelized by inserting a pragma “#omp parallel for”. A time-consuming part of the IFMM (i.e., the factorization phase) is not parallelized⁵, while a part of the initialization phase is parallelized with respect to the loop over clusters at each level in a similar way to the FMM.

In the following tests, we consider the wavenumbers k of 1, 2, 4, 8, 16, 32, and 64. These correspond to wavelengths $\lambda(= \frac{2\pi}{k})$ of 6.28, 3.14, 1.57, 0.785, 0.393, 0.196, and 0.0982, respectively. Larger wavenumbers could not be investigated due to the memory requirements of the IFMM (as well as the limitation of computation time) in our computer (see below). Such large wavenumbers can be handled by implementing a parallel version of the IFMM code in a distributed memory environment, on which the FMM can be parallelized and highly accelerated (e.g., [58, 59, 60]).

All the computations are performed with double-precision complex arithmetic. A computing node (of the Sherlock System at the Stanford Research Computing Center) with quad Intel Xeon E5-4640 processors (2.40GHz; 32 cores in total) and 1.5TB memory is employed. As a compiler, gcc-5.3.0 is used with the options “-O3 -mavx -fopenmp”.

Note that a single job (analysis) cannot run over the maximum runtime (i.e., 16 hours) on the computing node, which can be used through a batch job system. In practice, we usually limited the runtime of a job to 10 hours, because a job requesting a long runtime can wait in the queue for an unnecessarily long time. If a job could not finish within 10 hours (typically, because of the non-convergence of the GMRES iterations) and had the possibility to finish within 16 hours, we ran the job again with setting the runtime to its maximum.

4.1. Parameter settings

The linear system $\mathbf{Ax} = \mathbf{b}$, resulting from the discretization of the CBIE in Eq. (2), is solved by means of GMRES, using the FMM for the matrix-vector products. The efficiency of the IFMM (presented in Section 3) as preconditioner is compared to that of the block-diagonal (BD) preconditioner, which is commonly used in studies on FMBEM [13, Page 64]. This subsection summarizes the parameters used in the following numerical benchmarks.

4.1.1. GMRES

- Tolerance (denoted by tol): GMRES is terminated if the relative residual drops below a predefined tolerance, i.e., if $\frac{\|\mathbf{Ax} - \mathbf{b}\|_2}{\|\mathbf{b}\|_2} < \text{tol}$. We use $\text{tol} = 10^{-5}$.
- The initial guess of the solution is set to zero.

---

⁵The Schur complement update could easily be parallelized, but this is not implemented yet.

271 • The restart parameter is set to 2000, but restart did not occur in any case. We note that
 272 the computational cost of the GMRES is proportional to the square of the number of it-
 273 erations. It is thus very important to reduce the number of iterations by using a proper
 274 preconditioner.

275 • Preconditioning: we need a good left-preconditioner $\mathbf{M} \approx \mathbf{A}^{-1}$ in order to accelerate the it-
 276 erative process. We compare the proposed IFMM preconditioner (see Subsubsection 4.1.3
 277 for parameter settings) with the (leaf-based) BD preconditioner. Since the diagonal of $\mathbf{A}$ is
 278 covered by non-overlapping submatrices (or diagonal blocks), each of which corresponds
 279 to a leaf cluster of the FMM, $\mathbf{A}$ can be approximated with the union of the diagonal blocks.
 280 Then, $\mathbf{M} (\approx \mathbf{A}^{-1})$ can be computed by inverting each submatrix independently. Note that
 281 the inverses of those submatrices are precomputed at the initial stage in order to reduce the
 282 computation time of the BD preconditioner as much as possible. Also, the loop over leaf
 283 clusters are parallelized with OpenMP.

284 Other typical scenarios are to turn off the preconditioning, where $\mathbf{M}$ is interpreted as the
 285 identity matrix, and to use the point-Jacobi (PJ) preconditioner, where $\mathbf{M}$ is the diagonal
 286 matrix that contains the inverse of the diagonal elements of $\mathbf{A}$. However, in our preliminary
 287 examinations, these preconditioning strategies mostly resulted in poorer performance than
 288 the BD preconditioner with respect to both the number of iterations and the computation
 289 time until convergence. We have therefore excluded these results.

290 In addition, we compared the proposed IFMM preconditioner with an inner-outer preconditioner.
 291 In this preconditioner, $\mathbf{A}$ is approximated with the union of all the P2P matrices,
 292 i.e., all the near-neighbor interactions of all the leaf clusters. Then, it is no longer possible
 293 to invert the approximated matrix easily. To invert it, we applied the inner-outer scheme
 294 (recall Section 1). Unfortunately, the resulting computation time was usually larger than
 295 that of the BD preconditioner because of the additional computation cost to perform the
 296 inner solution every iteration. We have hence also excluded the results of this approach.

297 4.1.2. Forward FMM (for matrix-vector products)

298 • The maximum number of collocation points or boundary elements per leaf cluster (denoted
 299 by maxepc): this parameter determines the depth of the hierarchy of clusters (i.e., an
 300 adaptive octree) and thus the amount of far- and near-fields calculations in the FMM. It
 301 is well-known that this parameter affects the computation time of the FMM-accelerated
 302 matrix-vector products. However, our prime focus is not optimally tuning this parameter
 303 but rather on investigating the efficiency of the IFMM preconditioner. Hence, we use
 304 maxepc = 100 in this paper.

• The expansion order p , determining the accuracy of the FMM, is determined through a
 parameter δ and the following empirical formula [6]:

$$p := kD + \delta \log(kD + \pi), \quad (6)$$

305 where L is the edge length of clusters at the level under concern and $D := \sqrt{3}L$. In what
 306 follows, we use $\delta = 5$ (a relatively low accurate FMM) and compare it with $\delta = 7$ (a
 307 relatively high accurate FMM) in Subsubsection 4.5.1. Table 1 lists the range of p for the
 308 four models, which we will consider below.

Table 1: Range of the expansion order p in the FMM.

Model	δ	Wavenumber k						
		1	2	4	8	16	32	64
Sphere	5	5–6	6–7	6–9	6–12	7–18	9–28	12–44
	7	8–9	8–10	8–12	9–16	10–23	12–33	16–51
Horse	5	5–6	5–7	5–9	6–12	6–18	6–28	7–44
	7	8–9	8–10	8–12	8–16	8–23	9–33	10–51
Head	5	5–6	5–7	6–9	6–12	6–18	7–28	9–44
	7	8–9	8–10	8–12	8–16	9–23	10–33	12–51
House	5	6–8	6–9	6–13	8–19	9–29	13–48	19–81
	7	8–10	8–13	9–17	10–24	13–35	17–55	24–90

Note that, when the expansion order varies level by level as in Eq. (6), performing a matrix-vector product $\mathbf{A}\mathbf{x}$ in Eq. (3) by means of the LFFMM leads to a higher computational complexity than $O(N)$, which is not truly “fast”. However, this is not a concern because we are focusing on reducing the number of iterations through application of the IFMM preconditioner, rather than on the cost of a single matrix-vector product.

4.1.3. Inverse FMM (for preconditioning)

- The parameter ε , determining the relative cut-off and hence the accuracy of the low-rank compressions, is varied from $\varepsilon = 10^{-1}$ (cheap but approximate preconditioner) down to 10^{-2} , 10^{-3} , and 10^{-4} (expensive but more accurate preconditioner). For comparison, much smaller values of ε are examined in Subsection 4.3.
- A variety of numerical techniques can be applied for computing low-rank approximations, such as a singular value decomposition (SVD) [61], randomized SVD (RSVD) [62], adaptive cross approximation [63], rank revealing QR [61], etc [64]. We use RSVD in all numerical benchmarks because of its computational efficiency and robustness [41].
- The depth of the IFMM tree: The IFMM uses its own tree structure, which is non-adaptive (uniform), i.e., all the leaves exist at the deepest level. We set the depth of the IFMM’s tree to that of the FMM’s tree, i.e., ℓ .

4.2. Canonical example — Sphere model

We consider a scattering problem for a sphere of diameter one centered at the origin. The sphere is supposed to be acoustically hard, i.e., $q = 0$ is imposed on the surface. The sphere is irradiated by a planewave of the prescribed wavenumber k propagating in the $+x_3$ direction, i.e., $u^I(\mathbf{x}) = e^{ikx_3}$. This external Neumann BVP is not practical but valuable because the exact solution is available [65] to check the accuracy of the corresponding numerical solution. Table 2 summarizes the characteristics of the model. In this table, L_0 stands for the edge length of a cube that encloses the model. This length corresponds to the size of the root cluster of the FMM and IFMM hierarchies. The sphere is discretized with 10580, 50000, 100820, 151380, or 200000 triangular piecewise-constant boundary elements. For these N s, the depths of octree are 3, 4, 5, 5, and 5, respectively. We compare the efficiency of the proposed IFMM preconditioner (using $\varepsilon = 10^{-1}$, 10^{-2} , 10^{-3} , or 10^{-4}) with the **BD preconditioner**.

Table 2: Characteristics of the underlying four kinds of BVPs.

Model	Boundary condition(s)	Incident field [†]	N (#elements)	Root size L_0
Sphere	$q = 0$ everywhere	$u^I(\mathbf{x}) = e^{ikx_3}$	10580–200000	1.0
Horse	$q = 0$ everywhere	$u^I(\mathbf{x}) = e^{ikx_1}$	32206–1019036	1.0
Head	$q = 0$ everywhere	$u^I(\mathbf{x}) = \frac{e^{ik \mathbf{x}-\mathbf{s} }}{ \mathbf{x}-\mathbf{s} }$ [‡]	238598	1.0
House	$u = 1$ on ‘TV’, $q = 0$ everywhere else	N/A	147168	2.2

[†] The i^{th} component of $\mathbf{x}$ is denoted by x_i . [‡] The point $\mathbf{s}$ denotes a source point given at $(1, \frac{1}{2}, \frac{1}{2})^T$, which is close to the left-hand side “ear” of the **Head** model.

To see the efficiency of the IFMM preconditioner at glance, the **IFMM** and **BD** preconditioners are compared for the largest N i.e., $N = 200k$ in Table 3, where the meaning of the column headings is as follows:

- **k**: wavenumber.
- **niter**: number of GMRES iterations until convergence is achieved for a given tolerance, i.e., $\text{tol} = 10^{-5}$.
- **nsp**: speedup relative to the BD preconditioner in terms of the number of iterations.
- **total**: total computation time (in seconds), which includes everything from the start to the end of an analysis. This is decomposed into the following dominant components:
  - **mv**: time spent performing matrix-vector products with the FMM.
  - **prec**: time spent applying the preconditioner in GMRES.
  - **init**: initialization cost of the IFMM preconditioner.
  - **fact**: factorization (elimination) cost of the IFMM preconditioner.
- **tsp**: speedup relative to the BD preconditioner in terms of the total computation time.
- **l2err**: ℓ_2 -error of the solution for the wave field, denoted by $\mathbf{u}_{\text{target}} \in \mathbb{C}^N$ and computed with a target preconditioner (i.e., IFMM or BD preconditioner), relative to the exact solution, denoted by $\mathbf{u}_{\text{exact}} \in \mathbb{C}^N$, i.e., $\text{l2err} := \frac{\|\mathbf{u}_{\text{target}} - \mathbf{u}_{\text{exact}}\|_2}{\|\mathbf{u}_{\text{exact}}\|_2}$. Note that the exact solution is available only for the Sphere model.
- **disc**: Relative discrepancy of the solution $\mathbf{u}_{\text{IFMM}} \in \mathbb{C}^N$ computed with the IFMM preconditioner from the reference solution $\mathbf{u}_{\text{BD}}$ computed with the BD preconditioner, i.e., $\text{disc} := \frac{\|\mathbf{u}_{\text{IFMM}} - \mathbf{u}_{\text{BD}}\|_2}{\|\mathbf{u}_{\text{BD}}\|_2}$. This is used to measure the relative accuracy of the IFMM preconditioner for cases that don’t have an exact reference solution.

Table 3 shows that the IFMM preconditioner can reduce the number of iterations significantly in all cases and becomes faster than the BD preconditioner as k becomes larger. More precisely, the IFMM preconditioner is useful when the size (L_0) of the model is about five wavelengths or more, which corresponds to $k \gtrsim 16$, for ε of 10^{-1} and 10^{-2} .

On the other hand, the BD preconditioner can reduce the number of iterations to some extent at low-frequencies (compared to the case without preconditioning), but the number of iterations

increases substantially as k increases. The main advantage of the standard preconditioners is that they are very cheap, i.e., the time to construct and apply them is negligible.

Figure 1 plots the ℓ_2 -error of the two preconditioners relative to the exact solution against N . We will show the results for $k = 1$ and 16 because their tendencies are similar to the others for low- and high-frequencies, respectively. We can observe that the solutions obtained by all the preconditioners agree with the exact solution in two or three digits. The figure also shows the relative discrepancy from the BD preconditioner. These discrepancies are smaller than the former errors. This indicates that the choice of the preconditioner does not affect the solution.

Note that, in general, the ℓ_2 -error should decrease as N increases. This is not the case when $k = 1$ in Figure 1. To explain this, we increased the parameter δ of the “log” formula in Eq. (6). In the case of $k = 16$, the error did not change significantly. Meanwhile, in the case of $k = 1$, the error clearly improved; more precisely, the error did not increase monotonically but fluctuated around 10^{-3} as N increased. These experiments indicate that the accuracy is sensitive to the value of δ when k is small: the “log” formula can underestimate the expansion order p when kD is small. As a result, the error can increase as N increases. This is an important issue if one needs to control the accuracy exactly. However, our prime interest lies in the discrepancy between the two preconditioners rather than the concerning behavior of the error with respect to N . Hence, we decided to exploit the “log” formula as is.

Figure 2 shows the contours of the speed-ups “nsp” (in the LHS column) and “tsp” (in the RHS column) on the $N - k$ plane. Note that the data for $k = 64$ is not included in this figure because the BD preconditioner was unable to achieve convergence of GMRES for $N = 11k$ and 50k at $k = 64$ within the limited time, i.e., 10 hours. Probably, this is because each leaf cluster has more global interactions at long-range as N becomes smaller, and thus the BD preconditioner becomes inadequate. Nevertheless, the IFMM preconditioner converged after several iterations at $k = 64$. As a result, the IFMM preconditioner can significantly outperform the BD preconditioner for $N \sim 10^4$ and $k \sim 32$ in the figure.

As seen in Table 3, which is for $N = 200k$, the number of iterations for the BD preconditioner increases as k increases and reaches 351, while the IFMM preconditioner requires at most 12 iterations. Hence, “nsp” is of 10^1 order in Figure 2. However, since the IFMM preconditioner requires a long precomputation time, i.e., the initialization and factorization processes, the resulting “tsp” seems relatively moderate. Nevertheless, a few times acceleration is achieved as k increases for any N . In this test, $\varepsilon = 10^{-1}$ is the best choice in terms of the performance. A larger ε can lead to a larger speed-up. However, we will see that $\varepsilon = 10^{-1}$ is too large to maintain the accuracy of the solution in another problem (see 4.4). Hence, we do not pursue to seek the best value of ε larger than 10^{-1} .

Regarding the ℓ_2 -error in Table 3, the error is basically independent of k . To explain it, we would like to point out that the number of boundary elements $N = 200k$ used in the table is sufficiently large for the underlying wavenumbers $k = 1$ to $k = 64$. Actually, assuming that all the N triangles are equilateral, the number of boundary elements per wavelength is estimated as 16 for the largest wavenumber $k = 64$. In general, this number is enough to represent the wave field. Therefore, the error in Table 3 can reflect the discretization error for the Laplace operator, which is independent of k .

Figure 3 shows the memory usage, which indicates the peak amount of physical memory used by the underlying code. The memory usage of the IFMM preconditioner as well as the BD preconditioner is almost linear⁶. The IFMM preconditioner requires about ten times more

⁶This is not the case in the case of the BD preconditioner with $k = 16$. The reason is that the coefficients of $W_{n',n,m',m,l}$

memory than the BD preconditioner. In the worst case of $k = 64$, the IFMM preconditioner can require more memory than that to store a whole double complex N -by- N matrix, i.e., $(16N)^2 B$, which is drawn as the line indicated with “Full”. The huge memory consumption can restrict the size of the BVP to be solved with the help of the IFMM preconditioner.

We close this subsection with a comment on the IFMM-based direct solver. As a follow-up this study, we implemented a prototype of the IFMM-based direct solver and tested it for the Sphere model in the case of $k \leq 32$ and $N \leq 200k$. As a tentative result, we observed that the solver had $O(N)$ complexity in the cases of $\varepsilon = 10^{-3}$, 10^{-4} , and 10^{-5} . Roughly speaking, the direct solver was about ten times slower than the iterative solver using the IFMM preconditioner. Regarding the accuracy, the relative ℓ_2 -errors were $O(10^{-2})$ at most. This indicates that even $\varepsilon = 10^{-5}$ was not sufficiently small. However, a smaller ε would require a prohibiting computation time and memory in the current implementation. Further investigations (including the parallelization of the IFMM code and improvements of the algorithm) of the direct solver are necessary in the future. At this point, it is much more efficient to use the IFMM as a preconditioner than as a stand-alone direct solver.

Figure 1: The ℓ_2 -error of the IFMM preconditioner (using $\varepsilon = 10^{-1}$, 10^{-2} , 10^{-3} , and 10^{-4}) and BD preconditioner relative to the exact solution for the **Sphere** model. These errors are marked with ‘*’, while the errors without the mark represent the relative discrepancy of the IFMM preconditioner from the BD preconditioner. Refer to Subsection 4.2.

4.3. Complicated geometry — Horse model

We test the proposed IFMM preconditioner in a model with a more complicated geometry, i.e., a “Horse” model⁷ as shown in Figure 4. As well as the Sphere model, this model is supposed to be an acoustically hard scatterer and subjected to a planewave; see the characteristics in Table 2.

(see Appendix A), which is independent of N but dependent on k , takes the majority in the memory for the underlying $N \lesssim 200k$. Note that, if N becomes much larger, the memory usage of the BD preconditioner becomes linear.

⁷This model is available from <http://archive3d.net/?a=download&id=aac49a59>.

Table 3: Numerical results for the **Sphere** model ($N = 200k$) for the BD and IFMM preconditioners. The last column “l2err” denotes the ℓ_2 -error relative to the **exact solution**. Refer to [Subsection 4.2](#).

Precon.	k	niter	nsp	total	tsp	mv	prec	init	fact	l2err
BD	1	51	1.0	180	1.0	143	1	0	0	2.05×10^{-2}
	2	48	1.0	85	1.0	50	0	0	0	6.07×10^{-3}
	4	44	1.0	86	1.0	52	0	0	0	4.34×10^{-3}
	8	53	1.0	124	1.0	79	0	0	0	1.78×10^{-3}
	16	69	1.0	284	1.0	234	1	0	0	8.69×10^{-4}
	32	131	1.0	2181	1.0	2109	1	0	0	1.42×10^{-3}
	64	351	1.0	35 738	1.0	35 482	3	0	0	2.81×10^{-3}
IFMM $\varepsilon = 10^{-1}$	1	6	8.5	150	1.2	6	17	27	67	2.05×10^{-2}
	2	8	6.0	168	0.5	9	19	34	72	6.07×10^{-3}
	4	12	3.7	189	0.5	15	29	35	74	4.34×10^{-3}
	8	5	10.6	185	0.7	9	16	34	82	1.78×10^{-3}
	16	5	13.8	264	1.1	25	16	54	119	8.69×10^{-4}
	32	5	26.2	653	3.3	96	17	182	291	1.42×10^{-3}
	64	4	87.8	3322	10.8	541	28	1174	1430	2.81×10^{-3}
IFMM $\varepsilon = 10^{-2}$	1	5	10.2	166	1.1	5	19	26	82	2.05×10^{-2}
	2	5	9.6	204	0.4	13	26	33	97	6.07×10^{-3}
	4	4	11.0	195	0.4	6	20	34	100	4.34×10^{-3}
	8	4	13.3	227	0.5	7	23	36	118	1.78×10^{-3}
	16	3	23.0	343	0.8	13	25	59	195	8.70×10^{-4}
	32	5	26.2	834	2.6	93	44	179	452	1.42×10^{-3}
	64	5	70.2	3804	9.4	614	70	1142	1824	2.81×10^{-3}
IFMM $\varepsilon = 10^{-3}$	1	4	12.8	224	0.8	5	24	28	133	2.05×10^{-2}
	2	4	12.0	261	0.3	5	28	34	158	6.07×10^{-3}
	4	5	8.8	286	0.3	7	39	36	169	4.34×10^{-3}
	8	4	13.3	343	0.4	7	42	36	214	1.78×10^{-3}
	16	5	13.8	578	0.5	21	69	61	375	8.70×10^{-4}
	32	6	21.8	1310	1.7	111	113	183	833	1.42×10^{-3}
	64	4	87.8	4660	7.7	509	149	1170	2675	2.81×10^{-3}
IFMM $\varepsilon = 10^{-4}$	1	4	12.8	369	0.5	4	42	28	261	2.05×10^{-2}
	2	4	12.0	431	0.2	5	44	35	310	6.07×10^{-3}
	4	3	14.7	466	0.2	5	50	35	340	4.34×10^{-3}
	8	3	17.7	695	0.2	6	89	37	516	1.78×10^{-3}
	16	3	23.0	1341	0.2	13	164	63	1048	8.70×10^{-4}
	32	3	43.7	3035	0.7	64	242	184	2478	1.42×10^{-3}
	64	3	117.0	8472	4.2	405	414	1174	6320	2.81×10^{-3}

Figure 2: Contour plots of the speed-up of the IFMM preconditioner (using $\varepsilon = 10^{-1}$, 10^{-2} , 10^{-3} , and 10^{-4}) over the BD preconditioner with respect to “number of iterations until convergence” (left column; “nsp”) and “total computation time” (right column; “tsp”) for the **Sphere** model. The contours are smoothed by interpolating the speed-ups obtained on the grid of $N \in \{10580, 50000, 100820, 151380, 200000\}$ and $k \in \{1, 2, 4, 8, 16, 32\}$ with the B-spline in order to improve the visibility. Refer to [Subsection 4.2](#).

Figure 3: Memory usage (in GB) of the IFMM (using $\epsilon = 10^{-1}$, 10^{-2} , 10^{-3} , and 10^{-4}) and BD preconditioners (“BD”) for different N in the **Sphere** model. The legend “Full” shows the memory requirement to store a double-complex N -by- N matrix. In the cases of $k = 64$, the BD preconditioner could not perform the analyses of the first and second smallest N in 10 hours. Refer to Subsection 4.2.

Figure 4: Boundary element models: “Horse”, “Head”, “House”.

We compare the the IFMM and BD preconditioners in Table 4 in the case of $N = 190k$, for which the depth of octree is 7 in terms of the parameter `maxepc` = 100 (see Subsubsection 4.1.2). Here, the results of $k \in \{1, 2, 4\}$, for which the scale of the model (i.e., 1.0) is smaller than the corresponding wavelengths, are not highlighted because the IFMM preconditioner could not outperform the BD preconditioner similarly to the previous model. In comparison with the previous result in Table 3 for a simpler geometry, the complexity of boundary shape seems to increase the number of iterations until convergence. We can observe that the proposed IFMM preconditioner substantially outperforms the BD preconditioner in terms of the number of iterations. Regarding the total computation time, the IFMM preconditioner works better for relatively high wavenumbers, i.e., $k = 32$ and 64 , for which the scale of the model is equal to about five and ten wavelengths, respectively, similarly to the previous model.

A smaller value for ε clearly makes the preconditioner more accurate to $\mathbf{A}^{-1}$ and hence leads to a smaller number of iterations, at the expense of a more costly initialization and factorization phases. This is a trade-off that needs to be considered on a case-by-case basis to obtain an optimal value for ε that minimizes the total computation time. From the result (in Figure 5) for an auxiliary test using much smaller values of ε (i.e., $\varepsilon = 10^{-5}$ and 10^{-6}), the best choice is $\varepsilon = 10^{-1}$ in terms of the speed-up for computation time, while $\varepsilon = 10^{-2}$ is a proper choice in terms of the speed-up for number of iterations. There is no significant difference between these values of ε in terms of the relative discrepancy as well as the memory usage.

Next, we vary N up to one million, i.e., $N = 32206, 86470, 190156, 491040$, and 1019036 . For these N s, the depth of octree is 5, 6, 7, 7, and 8, respectively, in terms of the parameter `maxepc` = 100 (see Subsubsection 4.1.2).

Figure 6 shows the total computation time (“total”) and the number of iterations altogether for an optimal $\varepsilon = 10^{-2}$ and $k = 16, 32, 64$. The figure also shows the sum of initialization and factorization processes “init+fact”, which is the dominant part of the IFMM preconditioner. Clearly, the number of iterations for the BD preconditioner increases rapidly as the mesh size and frequency increase.

Figure 7 shows the relative discrepancy of the IFMM preconditioner from the BD preconditioner. We can observe that the discrepancy resulted in 10^{-5} order in any N and ε . The invariance for N is caused by the tolerance of the GMRES (`tol`) rather than the other parameters, i.e., δ and ε . As a matter of fact, the accuracy is reduced to 10^{-8} order when `tol` = 10^{-8} is used instead of 10^{-5} . On the other hand, the invariance for ε is reasonable because the preconditioning generally does not affect the accuracy of the solution $\mathbf{x}$ of $\mathbf{Ax} = \mathbf{b}$ but the number of iterations, although a

large ε can spoil the accuracy as we will see in the next subsection.

Figure 8 shows the memory usage of the IFMM and BD preconditioners. Again, the memory usage scales linearly in these preconditioners.

Figure 5: Results of the IFMM preconditioner using very small values of ε for the **Horse** model of $N = 190k$. Here, the speed-up over the BD preconditioner in terms of total computation time (top-left), the speed-up in terms of number of iterations (top-right), the relative discrepancy from the BD preconditioner (bottom-left), and the memory usage (bottom-right) are plotted. Refer to Subsection 4.3.

Figure 6: Total computation time (“total”) of the IFMM (using $\varepsilon = 10^{-2}$) and BD preconditioners in the case of $k = \{16, 32, 64\}$ for the **Horse** model. Regarding the IFMM preconditioner, the summation of the initialization and factorization processes “init+fact” is also plotted. Refer to Subsection 4.3.

4.4. More complicated problems — Head and House models

We consider more complicated models, i.e., “Head”⁸ and “House” models, as shown in Figure 4. The Head model ($N = 239k$) is topologically more complicated than the previous models,

⁸This model, which is a sculpture of a woman’s head, is available from <http://archive3d.net/?a=download&id=cf2f54ab>.

Table 4: Numerical results for the **Horse** model ($N = 190k$) for the **BD** and **IFMM** preconditioners. The last column “disc” denotes the **relative discrepancy of the IFMM preconditioner from the BD preconditioner**. Refer to [Subsection 4.3](#).

Precon.	k	niter	nsp	total	tsp	mv	prec	init	fact	disc
BD	8	68	1.0	154	1.0	90	1	0	0	N/A
	16	72	1.0	210	1.0	145	1	0	0	N/A
	32	114	1.0	962	1.0	875	1	0	0	N/A
	64	288	1.0	14 645	1.0	14 416	3	0	0	N/A
IFMM $\varepsilon = 10^{-1}$	8	18	3.8	445	0.3	76	106	72	125	1.19×10^{-5}
	16	14	5.1	361	0.6	29	52	85	128	1.55×10^{-5}
	32	10	11.4	635	1.5	154	77	124	204	1.64×10^{-5}
	64	10	28.8	1729	8.5	551	40	476	520	1.84×10^{-5}
IFMM $\varepsilon = 10^{-2}$	8	9	7.6	429	0.4	13	67	76	208	1.41×10^{-5}
	16	7	10.3	478	0.4	16	63	86	245	1.38×10^{-5}
	32	9	12.7	790	1.2	77	81	173	379	1.51×10^{-5}
	64	11	26.2	2632	5.6	598	136	892	863	1.75×10^{-5}
IFMM $\varepsilon = 10^{-3}$	8	3	22.7	582	0.3	5	55	73	383	1.36×10^{-5}
	16	3	24.0	757	0.3	8	67	81	533	1.33×10^{-5}
	32	4	28.5	1377	0.7	39	117	172	969	1.45×10^{-5}
	64	4	72.0	3473	4.2	255	158	932	1972	1.67×10^{-5}
IFMM $\varepsilon = 10^{-4}$	8	3	22.7	866	0.2	5	71	74	647	1.18×10^{-5}
	16	3	24.0	1245	0.2	8	105	85	979	1.34×10^{-5}
	32	3	38.0	2649	0.4	31	191	175	2170	1.42×10^{-5}
	64	3	96.0	7046	2.1	200	336	898	5457	1.67×10^{-5}

Figure 7: The **relative discrepancy** of the IFMM preconditioner (using $\varepsilon = 10^{-1}$ and 10^{-2}) from the BD preconditioner for the **Horse** model in the cases of $k = 16, 32$, and 64 . Refer to [Subsection 4.3](#).

Figure 8: Memory usage (in GB) of the IFMM (using $\varepsilon = 10^{-1}$ and 10^{-2}) and BD preconditioners (“BD”) for the **Horse** model in the cases of $k = 16, 32$, and 64 . The line of “Full” shows the memory requirement to fully store a double-complex N -by- N matrix. Refer to [Subsection 4.3](#).

470 i.e., it has two holes in the curly hair. Meanwhile, the House model ($N = 147k$) contains mul-
 471 tiple windows and doors (thus, topologically much more complicated). These two models are
 472 supposed to be acoustically hard scatterers, i.e., $q = 0$ on the surfaces of the models (see Ta-
 473 ble 2), whereas $u = 1$ (instead of $q = 0$) is given on a part of the boundary, which represents a
 474 “television (TV)”, in the House model (see Figure 9, where the TV is drawn with the solid lines).
 475 This is thus a mixed BVP.

Figure 9: Transparent view of the House model. The framed box in the front side of the ground floor models a “television”, the upper half of whose surface that faces the inside of the ground floor is subject to the boundary condition $u = 1$, while $q = 0$ is given everywhere else.

476 The results in Tables 5 and 6 are basically similar to those in the previous models.

477 The House model, where mixed boundary conditions are considered, seems the hardest one
 478 among the four models in terms of the number of iterations obtained by the BD preconditioner.
 479 As a matter of fact, the BD preconditioner could not perform $k = 32$ within the maximum run-
 480 time, i.e., 16 hours. In contrast, even for the hardest model, the IFMM preconditioner can reduce
 481 the number of iterations significantly, and the speed-up improves as k increases. In particular,
 482 the speed-up of 2.3 was obtained with $\varepsilon = 10^{-2}$ when $k = 8$, for which the length of the model
 483 is 2.8 wavelengths. Note that the BD preconditioner was unable to solve the Head model for the
 484 case of $k = 64$ within 16 hours.

485 Regarding accuracy, it is important to note that the relative discrepancy of the IFMM precon-
 486 ditioner is relatively large for the House model if $\varepsilon = 10^{-1}$ (see the underlined discrepancies in
 487 Table 6). Correspondingly, the IFMM preconditioner is unable to reduce the number of iterations
 488 significantly in those cases. This result indicates that a too large value of ε is not desirable.

489 4.5. Some auxiliary results

490 4.5.1. Influence of the parameter δ

491 We investigate the influence of the parameter δ that determines the expansion order p (cfr. Eq. (6)).
 492 In this subsection, δ is increased from 5 to 7. The computation time and accuracy of the BD
 493 and IFMM preconditioners are compared in Table 7 for the Head model, where $k \in \{4, 8, 16, 32\}$
 494 were computed.

495 Similarly to $\delta = 5$ in Table 4, there is a trade-off between the time spent constructing the
 496 IFMM preconditioner (initialization and factorization phases), and the time spent on matrix-
 497 vector products in GMRES. It should be noted that the speed-up with respect to time (“tsp”) is
 498 better than $\delta = 5$. This result is reasonable because increasing δ has a similar effect to increasing
 499 k in the sense that the expansion order p increases as δ and k increase. This result indicates that

Table 5: Numerical results for the **Head** model ($N = 239k$) for the BD and IFMM preconditioners. The last column “disc” denotes the **relative discrepancy of the IFMM preconditioner** relative to the BD preconditioner. **The BD preconditioner could not perform $k = 64$ within the max runtime, i.e., 16 hours; the number of iterations was 546 and the residual was 0.51 in the end. Because of this failure, the speed-ups “nsp” and “tsp” of the IFMM preconditioner are displayed as “ ∞ ”. Refer to Subsection 4.4.**

Precon.	k	niter	nsp	total	tsp	mv	prec	init	fact	disc
BD	8	72	1.0	285	1.0	186	1	0	0	N/A
	16	127	1.0	593	1.0	475	1	0	0	N/A
	32	292	1.0	5101	1.0	4867	4	0	0	N/A
	64	<i>546</i>	1.0	<i>16h</i>	1.0	-	-	0	0	N/A
IFMM $\varepsilon = 10^{-1}$	8	9	8.0	344	0.8	19	36	69	120	1.23×10^{-5}
	16	15	8.5	480	1.2	63	57	86	171	2.07×10^{-5}
	32	26	11.2	1242	4.1	452	92	165	411	6.89×10^{-5}
	64	14	∞	4965	∞	1566	63	1222	1905	N/A
IFMM $\varepsilon = 10^{-2}$	8	5	14.4	547	0.5	28	70	73	279	1.70×10^{-5}
	16	7	18.1	680	0.9	30	82	88	382	1.53×10^{-5}
	32	7	41.7	1327	3.8	134	103	218	757	2.01×10^{-5}
	64	8	∞	5203	∞	995	183	1214	2622	N/A
IFMM $\varepsilon = 10^{-3}$	8	4	18.0	842	0.3	10	106	71	554	1.16×10^{-5}
	16	4	31.8	1346	0.4	19	159	88	977	1.47×10^{-5}
	32	4	73.0	2507	2.0	97	227	171	1889	1.99×10^{-5}
	64	4	∞	7202	∞	538	341	1240	4860	N/A
IFMM $\varepsilon = 10^{-4}$	8	3	24.0	1111	0.3	7	134	72	759	1.15×10^{-5}
	16	3	42.3	2223	0.3	15	300	83	1720	1.47×10^{-5}
	32	3	97.3	5358	1.0	67	525	225	4414	1.99×10^{-5}
	64	3	∞	13 817	∞	414	830	1228	11 120	8.43×10^{80}

Table 6: Numerical results for the **House** model ($N = 147k$) for the BD and IFMM preconditioners. The last column “disc” denotes the relative discrepancy of the IFMM preconditioner relative to the BD preconditioner. The BD preconditioner could not perform $k = 32$ within the maximum runtime, i.e., 16 hours; the number of iterations was 317 and the residual was 0.66 in the end. Because of this failure, the speed-ups “nsp” and “tsp” of the IFMM preconditioner are displayed as “ ∞ ”. Refer to Subsection 4.4.

Precon.	k	niter	nsp	total	tsp	mv	prec	init	fact	disc
BD	4	190	1.0	384	1.0	316	1	0	0	N/A
	8	400	1.0	2253	1.0	2064	2	0	0	N/A
	16	974	1.0	25 979	1.0	25 034	5	0	0	N/A
	32	317	1.0	16h	1.0	-	-	0	0	N/A
IFMM $\varepsilon = 10^{-1}$	4	107	1.8	699	0.5	180	291	48	128	1.19×10^{-3}
	8	233	1.7	2330	1.0	1220	708	109	192	2.15×10^{-2}
	16	156	6.2	6193	4.2	4649	799	191	467	1.44×10^{-2}
	32	108	∞	25 070	∞	20 041	633	1715	2482	N/A
IFMM $\varepsilon = 10^{-2}$	4	5	38.0	546	0.7	10	77	48	368	1.74×10^{-4}
	8	10	40.0	988	2.3	57	151	109	620	4.48×10^{-5}
	16	18	54.1	2398	10.8	493	317	262	1257	1.01×10^{-4}
	32	42	∞	15 649	∞	8038	1308	1779	4326	N/A
IFMM $\varepsilon = 10^{-3}$	4	10	19.0	1092	0.4	19	252	50	728	1.75×10^{-4}
	8	4	100.0	2040	1.1	26	291	109	1562	4.48×10^{-5}
	16	5	194.8	4345	6.0	155	546	198	3374	1.05×10^{-4}
	32	7	∞	13 164	∞	1458	1421	1752	8328	N/A
IFMM $\varepsilon = 10^{-4}$	4	4	47.5	1617	0.2	29	294	57	1195	1.74×10^{-4}
	8	3	133.3	4079	0.6	20	745	113	3150	4.47×10^{-5}
	16	4	243.5	10 261	2.5	129	1675	266	8119	1.04×10^{-4}
	32	5	∞	23 610	∞	1088	3566	1875	16 884	N/A

the IFMM preconditioner could work better than the BD preconditioner as the precision level becomes higher, provided that the IFMM preconditioner would not use up the memory available; it required 151 and 367 GB for $\delta = 5$ and 7, respectively, for $k = 32$ and $\varepsilon = 10^{-2}$, for example. Note that the issue of the memory requirement can be addressed by implementing a parallel code.

4.5.2. Final rank of clusters

It is instructive to have a closer look at the final ranks of the clusters in the IFMM preconditioner, i.e., after the factorization has been performed. Figure 10 plots the rank of all the clusters (at all levels) for all wavenumbers $k \in \{1, 2, 4, 8, 16, 32, 64\}$ in the case of the Head model, for example, as a function of the dimensionless variable $\frac{L}{\lambda}$. Here, L denotes the edge length of a cluster, while $\lambda = \frac{2\pi}{k}$ corresponds to the wavelength. As a reference, the square of the expansion number p is superimposed on the figures, where p is represented as a function of $\frac{L}{\lambda}$ as follows (cfr. Eq. (6)):

$$p = \frac{2\pi\sqrt{3}L}{\lambda} + \delta \log\left(\frac{2\pi\sqrt{3}L}{\lambda} + \pi\right).$$

Here, p^2 corresponds to the dimensions of the M2L, M2M, and L2L matrices in Eq. (4), and thus to initial rank of each of the clusters.

We can observe from Figure 10 that the final rank of the clusters tends to increase as ε is decreased: a higher rank is required if we want a more accurate preconditioner. The tendency is similar to those of the Sphere, Horse, and House models. This confirms that the low-rank (re)compressions in the IFMM are controlled by the parameter ε . Note that the final rank of a cluster can either be smaller or larger than the initial rank p^2 , due to the compressions performed throughout the factorization process. For the large majority of the clusters, the rank remains reasonably small.

4.5.3. Visualization of pressure field u

For illustrative purposes, the pressure fields $\Re u(\mathbf{x})$ on the surface of the models are shown in Figure 11 in the case of $k = 16$. One can observe that a good correspondence is achieved between the pressure distributions obtained with the different preconditioners, except for the House model with $\varepsilon = 10^{-1}$ (recall Subsection 4.4 and Table 6). The bottom row of Figure 11 shows a somewhat strange pressure distribution on the right hand side of the outer wall of the house in that case.

5. Conclusion

In this paper, we propose to use the IFMM as a preconditioner in an iterative method (GMRES) in order to solve a linear system of equations, i.e., $\mathbf{Ax} = \mathbf{b}$ in Eq. (3), that results from the discretized CBIE for a 3D Helmholtz boundary element method. Here, the computation of the matrix-vector product $\mathbf{Ax}$ is accelerated by means of the FMM (specifically, the low-frequency FMM).

Despite the cost for constructing the IFMM preconditioner (i.e., the initialization and factorization phases), we demonstrated that the IFMM can outperform some typical preconditioners, such as a (leaf-based) block diagonal (BD) and a point Jacobi (PJ) preconditioner. This is especially the case if the BD or PJ preconditioner requires a large number of iterations until

Table 7: Numerical results for the **Head** model ($N = 239k$) for the BD and IFMM preconditioners, using $\delta = 7$ instead of $\delta = 5$. The last column “disc” denotes the **relative discrepancy of the IFMM preconditioner from the BD preconditioner**. Refer to [Subsubsection 4.5.1](#).

Precon.	k	niter	nsp	total	tsp	mv	prec	init	fact	disc
BD	8	72	1.0	512	1.0	410	1	0	0	N/A
	16	127	1.0	1868	1.0	1745	2	0	0	N/A
	32	292	1.0	14 571	1.0	14 322	5	0	0	N/A
IFMM $\varepsilon = 10^{-1}$	8	10	7.2	550	0.9	60	44	135	201	1.29×10^{-5}
	16	14	9.1	926	2.0	203	61	219	323	1.86×10^{-5}
	32	11	26.5	2076	7.0	661	84	355	820	2.20×10^{-5}
IFMM $\varepsilon = 10^{-2}$	8	7	10.3	925	0.6	44	109	141	519	1.34×10^{-5}
	16	10	12.7	1377	1.4	149	156	221	729	2.05×10^{-5}
	32	9	32.4	2652	5.5	494	142	487	1373	2.03×10^{-5}
IFMM $\varepsilon = 10^{-3}$	8	4	18.0	1850	0.3	73	231	138	1296	1.42×10^{-5}
	16	4	31.8	2394	0.8	68	225	222	1756	1.47×10^{-5}
	32	3	97.3	3908	3.7	195	221	369	2961	2.14×10^{-5}
IFMM $\varepsilon = 10^{-4}$	8	3	24.0	2983	0.2	23	320	143	2381	1.13×10^{-5}
	16	3	42.3	4766	0.4	54	483	198	3905	1.48×10^{-5}
	32	3	97.3	8054	1.8	196	608	481	6607	2.20×10^{-5}

Figure 10: Final rank of the clusters in the IFMM preconditioner for the **Head** model, as a function of the dimensionless variable $\frac{L}{\lambda}$. Superimposed is the initial rank p^2 , as defined in [Eq. \(6\)](#). Refer to [Subsubsection 4.5.2](#).

Figure 11: Pressure field $\Re u(x)$ on the surface of the Sphere, Horse, Head, and House models, for $k = 16$, as obtained with the BD and IFMM preconditioners. The figures in the bottom row magnify the right hand side of the outer wall of the House model, illustrating that the IFMM preconditioner with $\varepsilon = 10^{-1}$ leads to a relatively large error in the dotted circles. Note that the results of the IFMM preconditioner for $\varepsilon = 10^{-3}$ and 10^{-4} look completely the same as that for $\varepsilon = 10^{-2}$. Refer to Subsubsection 4.5.3.

convergence, which is usually the case for complicated boundary shapes, boundary conditions (especially, mixed boundary conditions), and/or high wavenumbers. The efficiency of the IFMM is governed by a prescribed tolerance ε for the low-rank compressions. Choosing a proper value for ε is essential for achieving numerically efficient results. In the examples discussed in Section 4, a value of 10^{-2} has yielded good results, leading to a speed-up of the computations without spoiling the accuracy of the solution. More precisely, an important observation about Tables 3–6 shows that the number of iterations was reduced by a factor of 7.6–70.2 relative to the BD preconditioner with using $\varepsilon = 10^{-2}$. Correspondingly, we observed 0.4–10.8 times speed-up with respect to the total runtime. Here, a speed-up less than one means a slow-down in practice. This is the case when the wavenumber k is relatively small, i.e., $k \lesssim 8$. Basically, the speed-up increased as k increased. The IFMM preconditioner was 1.2 times or more faster than the BD preconditioner when a scatterer was about five wavelengths or more; three wavelengths or more in the case of the House model. In addition, the IFMM preconditioner could solve the complicated problems (in a reasonable amount of time) that BD preconditioner could not. Such achievements are very promising and pave the way to deal with realistic large scale problems involving complicated geometries. On the other hand, it should be noted that large values of ε (e.g., $\varepsilon = 10^{-1}$ for this paper) can result in an inaccurate solution $\mathbf{x}$.

A future direction of the present study is to parallelize the IFMM implementation, although a part of the code (i.e., the initialization process) is already parallelized in the current version. It is one of our goals to tackle large problems, i.e., $N \gtrsim 10^6$ with using a fully parallelized code, which is dedicated for a shared-memory environment in order to relax the memory requirements of the IFMM. In addition, the use of an adaptive octree for the IFMM will undoubtedly increase its performance. Moreover, extending the present Helmholtz case to the Maxwell case is important in view of many important applications in computational science and engineering. It is further interesting to develop an IFMM preconditioner for other types of the FMMs such as the directional FMM [66], the Fourier-based FMM [67], wideband FMMs [68, 69], etc.

Acknowledgments

The work of the first author is partially supported by JSPS KAKENHI Grant Number 15K06683. The second author is a post-doctoral fellow of the Research Foundation Flanders (FWO) and a Francqui Foundation fellow of the Belgian American Educational Foundation (BAEF). The financial support is gratefully acknowledged.

Part of this research was done at Stanford University, and was supported in part by the U.S. Army Research Laboratory, through the Army High Performance Computing Research Center, Cooperative Agreement W911NF-07-0027. This material is also based upon work supported by the Department of Energy under Award Number DE-NA0002373-1.

Parts of the computing for this project was performed on the Sherlock cluster. We would like to thank Stanford University and the Stanford Research Computing Center for providing computational resources and support that have contributed to these research results.

References

- [1] R. Kress, Linear Integral Equations (Second Edition), no. 82 in Applied Mathematical Sciences, Springer Verlag, New York, 1999.
- [2] M. Bonnet, Boundary Integral Equation Methods for Solids and Fluids, John Wiley & Sons, West Sussex, 1999.

- [3] V. Rokhlin, Rapid solution of integral equations of classical potential theory, *Journal of Computational Physics* 60 (2) (1985) 187–207. doi:10.1016/0021-9991(85)90002-6.
- [4] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, *Journal of Computational Physics* 73 (2) (1987) 325–348. doi:10.1016/0021-9991(87)90140-9.
- [5] V. Rokhlin, Diagonal forms of translation operators for the Helmholtz equation in three dimensions, *Applied and Computational Harmonic Analysis* 1 (1993) 82–93. doi:10.1006/acha.1993.1006.
- [6] R. Coifman, V. Rokhlin, S. Wandzura, The fast multipole method for the wave equation: A pedestrian prescription, *Antennas and Propagation Magazine, IEEE* 35 (3) (1993) 7–12. doi:10.1109/74.250128.
- [7] M. A. Epton, B. Dembart, Multipole translation theory for the three-dimensional Laplace and Helmholtz equations, *SIAM J. Sci. Comput.* 16 (4) (1995) 865–897. doi:10.1137/0916051.
- [8] E. Darve, The fast multipole method: Numerical implementation, *Journal of Computational Physics* 160 (1) (2000) 195–240. doi:10.1006/jcph.2000.6451.
- [9] E. Darve, The fast multipole method I: Error analysis and asymptotic complexity, *SIAM Journal on Numerical Analysis* 38 (1) (2000) 98–128. doi:10.1137/S0036142999330379.
- [10] W. Chew, E. Michielssen, J. M. Song, J. M. Jin (Eds.), *Fast and Efficient Algorithms in Computational Electromagnetics*, Artech House, Inc., Norwood, MA, USA, 2001.
- [11] N. Nishimura, Fast multipole accelerated boundary integral equation methods, *Applied Mechanics Reviews* 55 (4) (2002) 299–324. doi:10.1115/1.1482087.
- [12] E. Darve, P. Havé, Efficient fast multipole method for low-frequency scattering, *Journal of Computational Physics* 197 (1) (2004) 341–363. doi:10.1016/j.jcp.2003.12.002.
- [13] Y. Liu, *Fast Multipole Boundary Element Method: Theory and Applications in Engineering*, Cambridge University Press, Cambridge, 2009.
- [14] Y. J. Liu, S. Mukherjee, N. Nishimura, M. Schanz, W. Ye, A. Sutradhar, E. Pan, N. A. Dumont, A. Frangi, A. Saez, Recent advances and emerging applications of the boundary element method, *Applied Mechanics Reviews* 64 (3) (2011) 030802. doi:10.1115/1.4005491.
- [15] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2008.
- [16] B. Carpentieri, Fast preconditioned Krylov methods for boundary integral equations in electromagnetic scattering, in: V. Barsan, R. Lungu (Eds.), *Trends in Electromagnetism from Fundamentals to Applications*, InTech, 2012, pp. 155–176.
- [17] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.
- [18] J. Lee, J. Zhang, C.-C. Lu, Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems, *J. Comput. Phys.* 185 (1) (2003) 158–175. doi:10.1016/S0021-9991(02)00052-9.
- [19] P. J. Harris, K. Chen, On efficient preconditioners for iterative solution of a Galerkin boundary element equation for the three-dimensional exterior Helmholtz problem, *Journal of Computational and Applied Mathematics* 156 (2) (2003) 303–318. doi:10.1016/S0377-0427(02)00918-4.
- [20] B. Carpentieri, I. S. Duff, L. Giraud, G. Sylvand, Combining fast multipole techniques and an approximate inverse preconditioner for large electromagnetism calculations, *SIAM Journal on Scientific Computing* 27 (3) (2005) 774–792. doi:10.1137/040603917.
- [21] S. Li, Q. Huang, A fast multipole boundary element method based on the improved Burton-Miller formulation for three-dimensional acoustic problems, *Engineering Analysis with Boundary Elements* 35 (5) (2011) 719–728. doi:10.1016/j.enganabound.2010.12.004.
- [22] D. R. Wilkes, A. J. Duncan, A low frequency elastodynamic fast multipole boundary element method in three dimensions, *Computational Mechanics* 56 (5) (2015) 829–848. doi:10.1007/s00466-015-1205-7.
- [23] S. Rao, D. Wilton, A. Glisson, Electromagnetic scattering by surfaces of arbitrary shape, *Antennas and Propagation, IEEE Transactions on* 30 (3) (1982) 409–418. doi:10.1109/TAP.1982.1142818.
- [24] M. G. Araujo, J. M. Bertolo, F. Obelleiro, J. L. Rodriguez, J. M. Taboada, L. Landesa, Geometry based preconditioner for radiation problems involving wire and surface basis functions, *Progress In Electromagnetics Research* 93 (2009) 29–40. doi:10.2528/PIER09042104.
- [25] O. Steinbach, W. Wendland, The construction of some efficient preconditioners in the boundary element method, *Advances in Computational Mathematics* 9 (1) (1998) 191–216. doi:10.1023/A:1018937506719.
- [26] K. Niino, N. Nishimura, Preconditioning based on Calderon’s formulae for periodic fast multipole methods for Helmholtz’ equation, *J. Comput. Phys.* 231 (1) (2012) 66–81. doi:10.1016/j.jcp.2011.08.019.
- [27] O. Bruno, T. Elling, C. Turc, Regularized integral equations and fast high-order solvers for sound-hard acoustic scattering problems, *International Journal for Numerical Methods in Engineering* 91 (10) (2012) 1045–1072. doi:10.1002/nme.4302.
- [28] M. Darbas, E. Darrigrand, Y. Lafranche, Combining analytic preconditioner and fast multipole

- method for the 3-D Helmholtz equation, *Journal of Computational Physics* 236 (2013) 289–316. doi:10.1016/j.jcp.2012.10.059.
- [29] E. van't Wout, P. Glat, T. Betcke, S. Arridge, A fast boundary element method for the scattering analysis of high-intensity focused ultrasound, *The Journal of the Acoustical Society of America* 138 (5) (2015) 2726–2737. doi:10.1121/1.4932166.
- [30] Y. Saad, M. H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM Journal on Scientific and Statistical Computing* 7 (1986) 856–869. doi:10.1137/0907058.
- [31] A. Grama, V. Kumar, A. Sameh, Parallel hierarchical solvers and preconditioners for boundary element methods, *SIAM Journal on Scientific Computing* 20 (1) (1998) 337–358. doi:10.1137/S1064827596313322.
- [32] N. A. Gumerov, R. Duraiswami, A broadband fast multipole accelerated boundary element method for the three dimensional Helmholtz equation, *The Journal of the Acoustical Society of America* 125 (1) (2009) 191–205. doi:10.1121/1.3021297.
- [33] S. Chaillat, J.-F. Semblat, M. Bonnet, A preconditioned 3-D multi-region fast multipole solver for seismic wave propagation in complex geometries, *Commun. Comput. Phys.* 11(2) (2012) 594–609. doi:10.4208/cicp.231209.030111s.
- [34] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, *SIAM Journal on Scientific Computing* 14 (2) (1993) 461–469. doi:10.1137/0914028.
- [35] P. Martinsson, V. Rokhlin, A fast direct solver for boundary integral equations in two dimensions, *Journal of Computational Physics* 205 (1) (2005) 1–23. doi:10.1016/j.jcp.2004.10.033.
- [36] E. Corona, P.-G. Martinsson, D. Zorin, An direct solver for integral equations on the plane, *Applied and Computational Harmonic Analysis* 38 (2) (2015) 284 – 317. doi:http://dx.doi.org/10.1016/j.acha.2014.04.002. URL http://www.sciencedirect.com/science/article/pii/S1063520314000529
- [37] J. Bremer, A. Gillman, P.-G. Martinsson, A high-order accurate accelerated direct solver for acoustic scattering from surfaces, *BIT Numerical Mathematics* 55 (2) (2015) 367–397. doi:10.1007/s10543-014-0508-y. URL http://dx.doi.org/10.1007/s10543-014-0508-y
- [38] K. L. Ho, L. Ying, Hierarchical interpolative factorization for elliptic operators: Differential equations, *Communications on Pure and Applied Mathematics* 69 (8) (2016) 1415–1451. doi:10.1002/cpa.21582. URL http://dx.doi.org/10.1002/cpa.21582
- [39] B. Liz, Fast direct solver for the boundary element method in electromagnetism and acoustics: $\mathcal{H}$ -matrices. parallelism and industrial applications, Ph.D. thesis, Universit Paris 13 (2014).
- [40] S. Ambikasaran, E. Darve, The inverse fast multipole method, *ArXiv e-prints* arXiv:1407.1572.
- [41] P. Coulier, H. Pouransari, E. Darve, The inverse fast multipole method: using a fast approximate direct solver as a preconditioner for dense linear systems, *ArXiv e-prints* arXiv:1508.01835.
- [42] P. Coulier, E. Darve, Efficient mesh deformation based on radial basis function interpolation by means of the inverse fast multipole method, *Computer Methods in Applied Mechanics and Engineering* 308 (2016) 286–309. doi:10.1016/j.cma.2016.05.029.
- [43] P. Coulier, B. Quaife, E. Darve, An efficient preconditioner for the fast simulation of a 2D Stokes flow in porous media, *International Journal for Numerical Methods in Engineering* In preparation.
- [44] L. Banjai, W. Hackbusch, Hierarchical matrix techniques for low- and high-frequency Helmholtz problems, *IMA Journal of Numerical Analysis* 28 (1) (2008) 46–79. doi:10.1093/imanum/drm001.
- [45] S. A. Cummer, J. Christensen, A. Al, Controlling sound with acoustic metamaterials, *Nature Reviews Materials* 1 (3) (2016) 16001+. doi:10.1038/natrevmats.2016.1.
- [46] Y. A. Erlangga, Advances in iterative methods and preconditioners for the Helmholtz equation, *Archives of Computational Methods in Engineering* 15 (1) (2008) 37–66.
- [47] M. Messner, M. Schanz, E. Darve, Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation, *Journal of Computational Physics* 231 (4) (2012) 1175–1196.
- [48] A. J. Burton, G. F. Miller, The application of integral equation methods to the numerical solution of some exterior boundary-value problems, *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 323 (1553) (1971) 201–210. doi:10.1098/rspa.1971.0097.
- [49] R. Kress, Minimizing the condition number of boundary integral operators in acoustic and electromagnetic scattering, *The Quarterly Journal of Mechanics and Applied Mathematics* 38 (2) (1985) 323–341. doi:10.1093/qjmam/38.2.323.
- [50] T. Takahashi, T. Matsumoto, An application of fast multipole method to isogeometric boundary element method for laplace equation in two dimensions, *Engineering Analysis with Boundary Elements* 36 (12) (2012) 1766–1775. doi:http://dx.doi.org/10.1016/j.enganabound.2012.06.004. URL http://www.sciencedirect.com/science/article/pii/S0955799712001373
- [51] M. Abramowitz, I. Stegun (Eds.), *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables* (eighth Dover printing), Dover, New York, 1972.

- [52] T. Takahashi, C. Cecka, E. Darve, An acceleration of multipole-to-local operation for low-frequency fast multipole BIEM with GPU, in: Extended Abstracts for Symposium of the International Association for Boundary Element Methods, IABEM 2011, 2011, pp. 303–308.
- [53] T. Takahashi, Y. Shimba, H. Isakari, T. Matsumoto, An efficient blocking M2L translation for low-frequency fast multipole method in three dimensions, *Computer Physics Communications* 202 (2016) 151–164. doi:10.1016/j.cpc.2016.01.008.
- [54] T. P. Pals, Multipole for scattering computations: Spectral discretization, stabilization, fast solvers, Ph.D. thesis, University of California Santa Barbara (December 2004).
- [55] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, T. Pals, A fast solver for HSS representations via sparse matrices, *SIAM Journal on Matrix Analysis and Applications* 29 (1) (2007) 67–81. doi:10.1137/050639028.
- [56] K. Ho, L. Greengard, A fast direct solver for structured linear systems by recursive skeletonization, *SIAM Journal on Scientific Computing* 34 (5) (2012) 2507–2532.
- [57] W. Fong, E. Darve, The black-box fast multipole method, *Journal of Computational Physics* 228 (2009) 8712–8725. doi:10.1016/j.jcp.2009.08.031.
- [58] E. Darve, C. Cecka, T. Takahashi, The fast multipole method on parallel clusters, multicore processors, and graphics processing units, *Comptes Rendus Mcanique* 339 (2-3) (2011) 185–193. doi:10.1016/j.crme.2010.12.005.
- [59] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner, T. Takahashi, Task-based FMM for multicore architectures, *SIAM Journal on Scientific Computing* 36 (1) (2014) C66–C93. doi:10.1137/130915662.
- [60] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner, T. Takahashi, Task-based FMM for heterogeneous architectures, *Concurrency and Computation-Practice & Experience* 28 (9) (2016) 2608–2629. doi:10.1002/cpe.37.
- [61] G. H. Golub, C. F. Van Loan, *Matrix Computations* (3rd Ed.), Johns Hopkins University Press, Baltimore, MD, USA, 1996.
- [62] N. Halko, P. G. Martinsson, J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, *SIAM Review* 53 (2) (2011) 217–288. doi:10.1137/090771806.
- [63] M. Bebendorf, S. Rjasanow, Adaptive low-rank approximation of collocation matrices, *Computing* 70 (2003) 1–24.
- [64] N. Kishore Kumar, J. Shneider, Literature survey on low rank approximation of matrices.
- [65] J. J. Bowman, T. B. A. Senior, P. L. E. Uslenghi, *Electromagnetic and Acoustic Scattering by Simple Shapes* (Revised edition), Hemisphere Publishing Corp., New York, 1987.
- [66] M. Messner, M. Schanz, E. Darve, Fast directional multilevel summation for oscillatory kernels based on chebyshev interpolation, *Journal of Computational Physics* 231 (4) (2012) 1175–1196. doi:10.1016/j.jcp.2011.09.027.
- [67] C. Cecka, E. Darve, Fourier-based fast multipole method for the Helmholtz equation, *SIAM Journal on Scientific Computing* 35 (1) (2013) A79–A103. doi:10.1137/11085774X.
- [68] E. Darve, P. Havé, A fast multipole method for Maxwell equations stable at all frequencies, *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 362 (1816) (2004) 603–628. doi:10.1098/rsta.2003.1337.
- [69] T. Takahashi, A wideband fast multipole accelerated boundary integral equation method for time-harmonic elastodynamics in two dimensions, *International Journal for Numerical Methods in Engineering* 91 (5) (2012) 531–551. doi:10.1002/nme.4288.
- [70] A. Messiah, *Quantum Mechanics: Two Volumes Bound as One*, Dover, Mineola, New York, 1999.
- [71] G. Guennebaud, B. Jacob, et al., *Eigen v3*, <http://eigen.tuxfamily.org> (2010).
- [72] N. Tessore, *RedSVD-h*, <https://github.com/ntessore/redsvd-h> (2014).

739 A. Low-frequency fast multipole method (LFFMM)

740 We will present the key formulae of the LFFMM, i.e., the P2M, M2M, M2L, L2L, L2P, and
 741 P2P operators. To carry out the LFFMM, we use the adaptive octree. According to the usual
 742 manner, we let the root cluster (at level 0) of the octree surround the boundary S of the given
 743 model. Then, the root cluster is divided into eight sub-clusters at level 1. Such divisions are
 744 repeated for every non-empty cluster recursively in order to create the clusters at higher levels.
 745 When a cluster contains a predefined number of boundary elements or less (see the parameter
 746 `maxepc` in Subsubsection 4.1.2), the cluster is no longer divided and called a leaf cluster.

747 If $|\mathbf{x} - \mathbf{s}| > |\mathbf{y} - \mathbf{s}|$ holds for any $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{s}$ in $\mathbb{R}^3$, the multipole expansion of the fundamental

748 solution Γ is given as follows:

$$\Gamma(\mathbf{x} - \mathbf{y}) = \frac{ik}{4\pi} \sum_{n=0}^{\infty} \sum_{m=-n}^n (2n+1) O_n^m(\mathbf{x} - \mathbf{s}) \overline{I_n^m(\mathbf{y} - \mathbf{s})}, \quad (\text{A.1})$$

749 where $\overline{(\cdot)}$ denotes the complex conjugate, and O_n^m and I_n^m are the following inner and outer
750 functions of order n and degree m :

$$O_n^m(\mathbf{x}) := \frac{h_n^{(1)}(k|\mathbf{x}|)}{s_n} Y_n^m\left(\frac{\mathbf{x}}{|\mathbf{x}|}\right), \quad I_n^m(\mathbf{x}) := \overline{s_n} j_n(k|\mathbf{x}|) Y_n^m\left(\frac{\mathbf{x}}{|\mathbf{x}|}\right).$$

751 Here, j_n and $h_n^{(1)}$ are the spherical Bessel and Hankel functions of order n , respectively [51], and
752 Y_n^m denotes the spherical harmonics of order n and degree m (see [70] for the definition of Y_n^m).
753 Also, $s_n := h_n^{(1)}(k\alpha)$ stands for the scaling (normalization) factor, where α denotes a constant
754 given at each level of FMM hierarchy; we use $\alpha = \frac{\sqrt{3}L}{2}$ by letting L be the edge length of the
755 clusters at a level of interest.

756 In what follows, we truncate the infinite summation over n in Eq. (A.1) with the first p orders.
757 Here, we determine the value of p by the empirical formula in Eq. (6), as discussed in Section 4.

758 The multipole expansion in Eq. (A.1), where the infinite summation $\sum_{n=0}^{\infty}$ is truncated as
759 $\sum_{n=0}^{p-1}$, allows to compute the matrix-vector product $\mathbf{f} := \mathbf{A}\mathbf{x}$ (where $\mathbf{x}$ represents a candidate or
760 an initial guess of the solution) that corresponds to the left-hand side (LHS) of Eq. (2)⁹ by the
761 following operations:

- 762 1. P2M: Creation of the multipole moments m_n^m of a leaf cluster S , whose center is denoted
763 by $\mathbf{s}$:

$$m_n^m(\mathbf{s}) = \sum_{\{j \mid S_j \in S\}} \int_{S_j} \left(\frac{\partial I_n^m(\mathbf{y} - \mathbf{s})}{\partial n_y} u_j - \overline{I_n^m(\mathbf{y} - \mathbf{s})} q_j \right) dS_y \quad (\text{A.2})$$

764 for $0 \leq n < p$ and $-n \leq m \leq n$.

- 765 2. M2M: Translation of the multipole moments of a cluster of S' to its parent (thus, non-leaf)
766 cluster S :

$$m_n^m(\mathbf{s}) = \sum_{n'=0}^{p'-1} \sum_{m'=-n'}^{n'} \sum_{\substack{l=|n-n'| \\ n'+n+l:\text{even}}}^{n+n'} (2n'+1)(-1)^{-m'} W_{n,n',m,-m',l} I_l^{-m+m'}(\mathbf{s} - \mathbf{s}') m_{n'}^{m'}(\mathbf{s}') \quad (\text{A.3})$$

767 for $0 \leq n < p$ and $-n \leq m \leq n$. Here, p (respectively, p') denotes the truncated number
768 determined for the level of S (S') and

$$W_{n',n,m',m,l} := (2l+1)i^{n-n'+l} \begin{pmatrix} n' & n & l \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} n' & n & l \\ m' & m & -m'-m \end{pmatrix}$$

769 with $\begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}$ denoting the Wigner-3j symbol [70, p.1056]. To create the multipole moments
770 of S , the above formula must be additively repeated for all the child clusters S' (up to 8
771 clusters).

⁹In our code, to compute $\mathbf{A}\mathbf{x}$, the i^{th} component x_i is the unknown u_i or q_i on S_i , while all the known data prescribed by the boundary conditions are set to zero. On the other hand, we can compute the RHS vector $\mathbf{b}$ in Eq. (3) as the product $\mathbf{f}$ by letting the unknown boundary data be zero and substituting the boundary conditions whose signs are reversed to u_i or q_i .

3. M2L: Translation of the multipole moments m_n^m of S to the local coefficient ℓ_n^m of O (whose center is denoted by $\mathbf{o}$):

$$\ell_n^m(\mathbf{o}) = \sum_{n'=0}^{p-1} \sum_{m'=-n'}^{n'} K_{n,m,n',m'}(\mathbf{o}-\mathbf{s}) m_{n'}^{m'}(\mathbf{s}) \quad (\text{A.4})$$

for $0 \leq n < p$ and $-n \leq m \leq n$. Here, the coefficient $K_{n,m,n',m'}(\mathbf{v})$ is given by

$$K_{n,m,n',m'}(\mathbf{v}) := \frac{2n'+1}{s_n s_{n'}} \sum_{\substack{l=|n-n'| \\ n'+n+l:\text{even}}}^{n+n'} W_{n',n,m',m,l} \tilde{O}_l^{-m-m'}(\mathbf{v}).$$

where $\tilde{O}_n^m(\mathbf{v}) := \frac{h_n^{(1)}(k|\mathbf{v}|)}{s_n} \overline{Y_n^m\left(\frac{\mathbf{v}}{|\mathbf{v}|}\right)}$. This operation should be repeated for all the (at most 189) clusters S that interact with O at the underlying level; we denote a set of such clusters S , i.e., the interaction-list of O , by $\mathcal{I}(O)$. Note that the local coefficient of a certain cluster can be computed via both the M2L formula and the following L2L formula in the so-called downward pass of the FMM.

4. L2L: Translation of the local coefficients of a cluster O' to its child cluster O :

$$\ell_n^m(\mathbf{o}) = \sum_{n'=0}^{p'-1} \sum_{m'=-n'}^{n'} \sum_{\substack{l=|n-n'| \\ n'+n+l:\text{even}}}^{n+n'} (2n'+1)(-1)^{-m} W_{n',n,m',-m,l} \mathcal{I}_l^{m-m'}(\mathbf{o}-\mathbf{o}') \ell_{n'}^{m'}(\mathbf{o}'). \quad (\text{A.5})$$

This operation must be repeated to all the child clusters O of O' .

5. L2P: Evaluation of the field (say f) at all the collocation points $\mathbf{x}_i \in O$ with the local-coefficients of O :

$$f(\mathbf{x}_i) = \frac{ik}{4\pi} \sum_{n=0}^{p-1} \sum_{m=-n}^n (2n+1) \left(\overline{\mathcal{I}_n^m(\mathbf{x}_i - \mathbf{o})} + \beta \frac{\partial \overline{\mathcal{I}_n^m(\mathbf{x}_i - \mathbf{o})}}{\partial n_x} \right) \ell_n^m(\mathbf{o}). \quad (\text{A.6})$$

The RHS denotes the far-field component of the field. The near field component is computed by the following P2P operation.

6. P2P: Evaluation of the contribution from a neighbor cluster S at all the collocation points $\mathbf{x}_i \in O$:

$$f(\mathbf{x}_i) = \sum_{\{j \mid S_j \in \mathcal{S}\}} \left[\left\{ \frac{I_{ij}}{2} + \int_{S_j} \left(\frac{\partial \Gamma}{\partial n_y}(\mathbf{x}_i, \mathbf{y}) + \beta \frac{\partial^2 \Gamma}{\partial n_x \partial n_y}(\mathbf{x}_i, \mathbf{y}) \right) dS_y \right\} u_j \right. \\ \left. + \left\{ \beta \frac{I_{ij}}{2} - \int_{S_j} \left(\Gamma(\mathbf{x}_i - \mathbf{y}) + \beta \frac{\partial \Gamma}{\partial n_x}(\mathbf{x}_i, \mathbf{y}) \right) dS_y \right\} q_j \right]. \quad (\text{A.7})$$

This operation must be repeated for all the clusters S that are adjacent to O ; a set of such S is referred to as the neighbor list of O , which is denoted by $\mathcal{N}(O)$. Here, f in Eq. (A.7) should be superimposed to that in Eq. (A.6).

To diminish the runtime, we precompute all the P2P matrices, which represent the P2P operation in A.7, at the initial stage of each analysis. This optimization can lead to a substantial speed-up for all the preconditioners. In particular, the conventional preconditioners can be

highly accelerated by this optimization because they usually require a large number of GMRES iterations until convergence, and the same P2P matrices used to be recomputed every iteration. Meanwhile, this P2P precomputation does not lead to a significant speed-up for the proposed IFMM preconditioner because it usually converges within a few iterations.

In addition, the coefficients $W_{n',n,m',m,l}$, which are necessary for the M2M, M2L, and L2L operations, are precomputed because they depend only on the values of the indexes. To this end, we determine an expansion order $\bar{p}$ so that it can cover all the expansion orders p under consideration. Then, we compute the coefficients $W_{n',n,m',m,l}$ for $n, n' \in [0, \bar{p})$, $m, m' \in (-\bar{p}, \bar{p})$, and $l \in [0, 2\bar{p})$, and store them into a file. Once this is done, we can load the coefficients of any expansion order p such as $p \leq \bar{p}$ from the file rapidly. As a drawback, this optimization requires a larger amount of memory. This is because the current implementation needs to load the whole of the data into a memory temporarily even when the actual order p is less than $\bar{p}$. Note that the coefficients thus loaded are reused by the IFMM preconditioner when it constructs the sparse matrix E , which is mentioned in Section 3, at the initial stage.

Note that the above two optimizations are independent of the choice of preconditioner. Hence, they can decrease the baseline of the computation time of the LFFMM.

B. Implementation details

We implemented the proposed IFMM preconditioner in C++, based on an existing implementation used in previous studies [41, 42]. In the latter, the bbFMM [57] is used to construct the initial low-rank operators, employing the C++ template library Eigen [71] to handle the dense matrix algebra. The RedSVD-h library [72] is used for the randomized SVD. The main modification in the current study is that the initial IFMM operators are based on the LFFMM described in Section 2 (instead of the bbFMM). The code has also been adjusted to deal with complex arithmetic. This implies that the SVD is expressed through the adjoint operator rather than the transpose one. In addition, the SVD is implemented with LAPACK's `zgesvd`.

The current implementation differs from the algorithm described in [40, 41] on two minor points. Firstly, we use the unscaled orthonormalized basis vectors when performing low-rank recompressions and do not account for the weights (i.e., the singular values) of each basis vector. Although accounting for these weights is the more rigorous strategy in the IFMM, we omitted this because the improved accuracy does not lead to reduced computational costs (because of the added cost of implementing this more correct approach). This had a negligible effect on the accuracy of the preconditioner.

Secondly, for this work, the IFMM is applied to non-symmetric matrices. As a result, there is a special difficulty to guarantee that the pivot $\mathbf{V}_k^{(l)*} \mathbf{S}_{kk}^{(l)-1} \mathbf{U}_k^{(l)}$ of each cluster remains non-singular (where $\mathbf{S}_{kk}^{(l)}$ denotes the P2P operator of a cluster k at level l with itself, while $\mathbf{U}_k^{(l)}$ and $\mathbf{V}_k^{(l)*}$ are its L2P and P2M operators, respectively). Although this rarely occurs, the near-singularity of a pivot could have a detrimental effect on the accuracy of the IFMM solver. We therefore implemented the following simple procedure: if the norm of the inverse of the pivot exceeds a certain threshold η (e.g., $\eta = 10^5$ in this study), we avoid the elimination of this cluster by pushing all the variables to the parent level. This implies that we replace $\mathbf{U}_k^{(l)}$ and $\mathbf{V}_k^{(l)*}$ with identity matrices, and consider a full-rank instead of a low-rank interaction for this cluster. This is a very simple approach that increases the robustness of the IFMM if applied to non-symmetric matrices, at a negligible additional computational cost (as this only occurs for very few clusters). Note that it is also possible to guarantee that the pivot is non-singular by using a symmetric

⁸³⁸ projection operator. This is a more rigorous procedure, but it might also be more expensive. This
⁸³⁹ is included in the topics of future research.