MULTISCALE MODEL. SIMUL.
Vol. 15, No. 2, pp. 768–796

A RECURSIVE SKELETONIZATION FACTORIZATION BASED ON STRONG ADMISSIBILITY[*]

VICTOR MINDEN[†], KENNETH L. HO[‡], ANIL DAMLE[§], AND LEXING YING[¶]

Abstract. We introduce the strong recursive skeletonization factorization (RS-S), a new approximate matrix factorization based on recursive skeletonization for solving discretizations of linear integral equations associated with elliptic partial differential equations in two and three dimensions (and other matrices with similar hierarchical rank structure). Unlike previous skeletonization-based factorizations, RS-S uses a simple modification of skeletonization, strong skeletonization, which compresses only far-field interactions. This leads to an approximate factorization in the form of a product of many block unit-triangular matrices that may be used as a preconditioner or moderate-accuracy direct solver, with dramatically reduced rank growth. We further combine the strong skeletonization procedure with alternating near-field compression to obtain the hybrid recursive skeletonization factorization (RS-WS), a modification of RS-S that exhibits reduced storage cost in many settings. Under suitable rank assumptions both RS-S and RS-WS exhibit linear computational complexity, which we demonstrate with a number of numerical examples.

Key words. hierarchical factorizations, integral equations, preconditioners, strong admissibility, fast direct solvers, fast multipole method

AMS subject classifications. 65R20, 65F08, 65F05

DOI. 10.1137/16M1095949

1. Introduction.

Given a kernel function $K(z)$, we consider the integral equation

$$a(x)u(x) + b(x)\int_{\Omega} K(x-y)c(y)u(y)\,dy = f(x), \quad x \in \Omega \subset \mathbb{R}^d, \tag{1}$$

in dimension $d = 2$ or 3. Here, $a(x)$, $b(x)$, and $c(y)$ are given functions that typically represent material parameters, $f(x)$ is some known right-hand side, and $u(x)$ is the unknown function to be determined.

We focus in this paper on the case where $K(z)$ is associated with some underlying elliptic partial differential equation (i.e., it is the Green's function or its derivative). For optimal complexity of our methods the kernel $K(z)$ should not exhibit significant oscillation away from the origin, though this is not strictly necessary to apply the

[*]Received by the editors September 27, 2016; accepted for publication (in revised form) January 17, 2017; published electronically April 25, 2017.
http://www.siam.org/journals/mms/15-2/M109594.html

Funding: The first author was supported by a Stanford graduate fellowship in science and engineering and a U.S. Department of Energy computational science graduate fellowship (grant DE-FG02-97ER25308). The second author was supported by a National Science Foundation mathematical sciences postdoctoral research fellowship (grant DMS-1203554). The third author was supported by a National Science Foundation mathematical sciences postdoctoral research fellowship (grant DMS-1606277). The fourth author was supported by the National Science Foundation (grant DMS-1521830) and the U.S. Department of Energy Advanced Scientific Computing Research program (grant DE-FC02-13ER26134 and DE-SC0009409).

[†]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305 (vminden@stanford.edu).
[‡]Department of Mathematics, Stanford University, Stanford, CA 94305. Current address: TSMC Technology, Inc., San Jose, CA 95134 (klho@alumni.caltech.edu).
[§]Department of Mathematics, University of California, Berkeley, Berkeley, CA 94720 (damle@berkeley.edu).
[¶]Department of Mathematics and Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305 (lexing@math.stanford.edu).

basic machinery. In this setting, (1) remains rather general and includes problems such as the Laplace equation, the Lippmann–Schwinger equation, and the Helmholtz equation in the low- to moderate-frequency regime. Further, while we concentrate on the case where $u(x)$ is scalar-valued, extension to the vector-valued case (e.g., the Stokes or elasticity equations) is straightforward.

Discretization of (1) using typical approaches such as collocation, the Nyström method, or the Galerkin method leads to a linear system with N degrees of freedom (DOFs)

$$\mathsf{K}u = f, \tag{2}$$

where the entries of the matrix $\mathsf{K} \in \mathbb{C}^{N\times N}$ are dictated by the kernel $K(z)$ and the discretization scheme. For example, in the case where our domain is the unit square $\Omega = [0,1]^2$ a simple Nyström approximation to the integral using a regular grid with $\sqrt{N}$ points in each direction yields the discrete system

$$[a(x_i) + w_i]\, u_i + \frac{b(x_i)}{N} \sum_{i\neq j} K(x_i - x_j) c(x_j) u_j = f(x_i), \quad i = 1, \ldots, N, \tag{3}$$

where the discrete solution $\{u_i\} \approx \{u(x_i)\}$ approximates the continuous solution on the grid and each term $w_i u_i$ corresponds to some discretization of diagonal entries of K. Because $K(z)$ is frequently singular at the origin, this discretization may be more involved than that of the off-diagonal entries. While more complicated and higher-order discretization schemes exist, (3) illustrates the key feature that off-diagonal entries of K are given essentially by *kernel interactions between distinct points in space*. In this paper we develop a method exploiting this fact and its consequences to efficiently solve (2).

1.1. Background and previous work.

Because K in (2) is dense and generally large in practice, traditional direct factorizations of K such as the LU factorization are typically too expensive due to the associated $O(N^3)$ time complexity and $O(N^2)$ storage cost.

Given the availability of fast schemes for applying K such as fast multipole methods (FMMs) [11, 14, 15, 33], iterative methods such as the conjugate gradient (CG) method [20] form a tempting alternative to direct methods. For first-kind integral equations or problems where $a(x)$, $b(x)$, or $c(x)$ exhibit high contrast, however, convergence is typically slow leading to a lack of robustness. In other words, while each iteration is relatively fast, the number of iterations necessary to attain reasonable accuracies can be unreasonably large.

The above considerations have led to the development of a plethora of alternative methods for solving (2) approximately by exploiting properties of the kernel $K(z)$ and the underlying physical structure of the problem. In particular, such methods take advantage of the fact that K exhibits *hierarchical block low-rank structure.*

A large body of work pioneered by Hackbusch and collaborators on the algebra of $\mathcal{H}$-matrices (and $\mathcal{H}^2$-matrices) provides an important and principled theoretical framework for obtaining linear or quasi-linear complexity when working with matrices exhibiting such structure [17, 18, 19]. Inside the asymptotic scaling of this approach, however, lurk large constant factors that hamper practical performance, particularly in the three-dimensional (3D) case.

The $\mathcal{H}$-matrix literature classifies matrices with hierarchical block low-rank structure into two categories based on which off-diagonal blocks of the matrix are compressed. Given a quadtree or octree data structure partitioning the domain into small

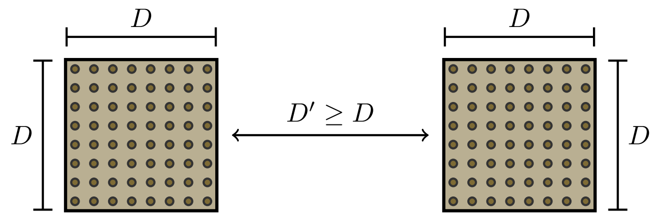


FIG. 1. *Given two boxes in $\mathbb{R}^2$ each with sidelength D and with corresponding DOF sets $\mathcal{B}_1$ and $\mathcal{B}_2$, in the strong admissibility setting the associated off-diagonal blocks $\mathsf{K}_{\mathcal{B}_1\mathcal{B}_2}$ and $\mathsf{K}_{\mathcal{B}_2\mathcal{B}_1}$ are assumed to be numerically low rank as long as the boxes are separated by a distance of at least D. In contrast, in the weak admissibility setting the boxes need only be nonoverlapping.*

boxes, let $\mathcal{B}_1$ and $\mathcal{B}_2$ be sets of DOFs corresponding to distinct boxes at the same level of the tree each with sidelength D. For *strongly admissible* hierarchical matrices, the off-diagonal block $\mathsf{K}_{\mathcal{B}_1\mathcal{B}_2}$ is compressed only if $\mathcal{B}_1$ and $\mathcal{B}_2$ are *well-separated* as in the FMM—that is, if $\mathcal{B}_1$ and $\mathcal{B}_2$ are separated by a distance of at least D as in Figure 1. In contrast, *weakly admissible* hierarchical matrices compress not only well-separated interactions but also interactions corresponding to DOFs in *adjacent* boxes. The inclusion of nearby interactions under weak admissibility typically increases the required approximation rank, but it also affords a much simpler geometric and algorithmic structure.

A number of more recent methods have been developed for hierarchically rank-structured matrices with the aim of more efficient practical performance based on weakly admissible rank structure. Examples include algorithms for hierarchical semi-separable matrices [4, 5, 32], hierarchical off-diagonal low-rank matrices [1, 24], and methods based on recursive skeletonization (RS) [25, 12, 21], among other related schemes [3, 6]. In general, methods based strictly on weak admissibility require allowing ranks of off-diagonal blocks to grow nonnegligibly with N to attain a fixed target accuracy. This has led to the development of more involved methods such as the hierarchical interpolative factorization (HIF) of Ho and Ying [22] and the method of Corona, Martinsson, and Zorin [8], which combine RS with additional compression steps based on geometric considerations to obtain greater efficiency at the cost of a more complicated algorithm.

There has been much less work on improved algorithms for solving (2) based directly on strong admissibility. The stand-out example is the recent "inverse fast multipole method" (IFMM) of Coulier, Pouransari, and Darve [9] and Ambikasaran and Darve [2], which assumes a general $\mathcal{H}^2$-matrix is given and provides a framework for approximately applying the inverse operator in the language of the FMM. Further, a factorization based on block elimination and strong admissibility has been recently introduced by Sushnikova and Oseledets [31] for the "sparse analogue" of our integral equation setting (that is, discretizations of elliptic partial differential equations).

1.2. Contributions.

Based on the RS process of Martinsson and Rokhlin [25] in the block elimination form of Ho and Ying [22, section 3], we introduce *strong skeletonization*, an extension of skeletonization for the strong admissibility setting. Using this in a recursive fashion like the original RS factorization, we develop the *strong recursive skeletonization factorization* (RS-S), an approximate factorization of K into the product of many block unit-triangular matrices and a block-diagonal matrix with time complexity linear in the number of DOFs, under suitable rank-scaling assumptions. Using low-accuracy approximations to off-diagonal blocks yields an effective preconditioner for iterative methods applied to (2), whereas at higher accuracies the resulting factorization can be used as a direct solver.

Like the IFMM [9, Appendix C], our factorization uses a bottom-up traversal of the quadtree or octree decomposition of space to compress well-separated interactions on a level-by-level basis. This allows efficient on-the-fly construction of a nested "skeleton" basis for representing far-field interactions at different levels during the factorization process, in contrast to the typical recursive $\mathcal{H}$-matrix inversion algorithm. Using skeletonization to maintain problem structure and exploit accelerated compression techniques (see subsection 3.2), we obtain what may be thought of as a multiplicative analogue of the FMM, using the same strong admissibility structure. This gives a factorization of K or K^{-1} with simple constituent factors that is easy to understand and implement.

As an extension to our approach, we combine the original weak-admissibility-based skeletonization process with our strong-admissibility-based skeletonization and introduce the *hybrid recursive skeletonization factorization* (RS-WS), which uses additional compression steps like HIF or the method of Corona, Martinsson, and Zorin but does so without the need for spatial geometry beyond the boxes of the tree decomposition. This additional compression reduces memory usage for practical performance gains in many cases.

2. Preliminaries.

In the remainder of this paper we adopt the following notation. For a positive integer N, the index set $\{1, 2, \ldots, N\}$ is denoted by $[N]$. We write matrices or matrix-valued functions in the sans serif font (e.g., $\mathsf{A} \in \mathbb{C}^{N\times N}$) but make no such distinction for vectors (e.g., $x \in \mathbb{C}^N$). Given a vector or matrix, the norms $\|x\|$ or $\|\mathsf{A}\|$ refer to the standard Euclidean vector norm and corresponding induced matrix norm, respectively. The math-calligraphic font is used to indicate index sets (e.g., $\mathcal{I} = \{i_1, i_2, \ldots, i_r\}$ with each i_j a positive integer) that we use to index blocks of a matrix (e.g., $\mathsf{A}_{\mathcal{I}\mathcal{J}} = \mathsf{A}(\mathcal{I}, \mathcal{J}) \in \mathbb{C}^{|\mathcal{I}|\times|\mathcal{J}|}$, using MATLAB notation). Therefore, each index set has an implicit ordering, though we use the term "set" as opposed to "vector" to avoid conflation. Because we are working with matrices discretizing integral equations, indices in an index set are typically associated with points in $\mathbb{R}^d$ (e.g., Nyström or collocation points or centroids of elements). As such, we will use the more general term "DOF sets" to refer to both the index set $\mathcal{B}$ and the corresponding points $\{x_i\}_{i\in\mathcal{B}}$ in $\mathbb{R}^d$. Finally, to denote ordered sets of positive integers that are not associated with points in the domain nor used to index matrices we use the math-script font (e.g., $\mathscr{L}$).

2.1. Block-structured elimination.

We begin with a brief review of block-structured elimination and its efficiency, which is central to the skeletonization algorithm.

Let $\mathsf{A} \in \mathbb{C}^{N\times N}$ be an $N \times N$ matrix and suppose $[N] = \mathcal{I} \cup \mathcal{J} \cup \mathcal{K}$ is a partition of the index set of A such that both $\mathsf{A}_{\mathcal{I}\mathcal{K}} = 0$ and $\mathsf{A}_{\mathcal{K}\mathcal{I}} = 0$, i.e., we have the block structure

$$\mathsf{A} = \left[\begin{array}{c|c|c} \mathsf{A}_{\mathcal{I}\mathcal{I}} & \mathsf{A}_{\mathcal{I}\mathcal{J}} & \\ \hline \mathsf{A}_{\mathcal{J}\mathcal{I}} & \mathsf{A}_{\mathcal{J}\mathcal{J}} & \mathsf{A}_{\mathcal{J}\mathcal{K}} \\ \hline & \mathsf{A}_{\mathcal{K}\mathcal{J}} & \mathsf{A}_{\mathcal{K}\mathcal{K}} \end{array}\right],$$

up to permutation. Assuming that the block $\mathsf{A}_{\mathcal{I}\mathcal{I}}$ is invertible, the DOFs $\mathcal{I}$ can be decoupled as follows. First, define the matrices L and U as

$$\mathsf{L} \equiv \left[\begin{array}{c|c|c} \mathsf{I} & & \\ \hline -\mathsf{A}_{\mathcal{J}\mathcal{I}}\mathsf{A}_{\mathcal{I}\mathcal{I}}^{-1} & \mathsf{I} & \\ \hline & & \mathsf{I} \end{array}\right], \quad \mathsf{U} \equiv \left[\begin{array}{c|c|c} \mathsf{I} & -\mathsf{A}_{\mathcal{I}\mathcal{I}}^{-1}\mathsf{A}_{\mathcal{I}\mathcal{J}} & \\ \hline & \mathsf{I} & \\ \hline & & \mathsf{I} \end{array}\right] \tag{4}$$

with the same block partitioning as A. Then, applying these operators on the left and right of A yields

$$\mathsf{LAU} = \left[\begin{array}{c|c|c} \mathsf{A}_{\mathcal{I}\mathcal{I}} & & \\ \hline & \mathsf{S}_{\mathcal{J}\mathcal{J}} & \mathsf{A}_{\mathcal{J}\mathcal{K}} \\ \hline & \mathsf{A}_{\mathcal{K}\mathcal{J}} & \mathsf{A}_{\mathcal{K}\mathcal{K}} \end{array}\right], \tag{5}$$

where $\mathsf{S}_{\mathcal{J}\mathcal{J}} = \mathsf{A}_{\mathcal{J}\mathcal{J}} - \mathsf{A}_{\mathcal{J}\mathcal{I}}\mathsf{A}_{\mathcal{I}\mathcal{I}}^{-1}\mathsf{A}_{\mathcal{I}\mathcal{J}}$ is the only nonzero block of the resulting matrix that has been modified.

We say that $\mathsf{S}_{\mathcal{J}\mathcal{J}}$ is related to $\mathsf{A}_{\mathcal{J}\mathcal{J}}$ through a *Schur complement update*. Note that while we choose here to write block elimination in its simplest form, in practice it can be numerically advantageous to work with a factorization of $\mathsf{A}_{\mathcal{I}\mathcal{I}}$ as is done by Ho and Ying [22, Lemma 2.1] as opposed to inverting the submatrix directly. Either way, the cost of computing $\mathsf{S}_{\mathcal{J}\mathcal{J}}$ is $O(|\mathcal{I}|^3 + |\mathcal{I}| \cdot |\mathcal{J}|^2)$.

2.2. Compression via the interpolative decomposition. Another key linear algebra tool of which we will make heavy use is the *interpolative decomposition* (ID) [7].

DEFINITION 2.1. *Given both a matrix* $\mathsf{A}_{\mathcal{I}\mathcal{J}} \in \mathbb{C}^{|\mathcal{I}|\times|\mathcal{J}|}$ *with rows indexed by* $\mathcal{I}$ *and columns indexed by* $\mathcal{J}$ *and a tolerance* $\epsilon > 0$*, an* ϵ*-accurate ID of* $\mathsf{A}_{\mathcal{I}\mathcal{J}}$ *is a partitioning of* $\mathcal{J}$ *into DOF sets associated with so-called skeleton columns* $\mathcal{S} \subset \mathcal{J}$ *and redundant columns* $\mathcal{R} = \mathcal{J} \setminus \mathcal{S}$ *and a corresponding interpolation matrix* T *such that*

$$\|\mathsf{A}_{\mathcal{I}\mathcal{R}} - \mathsf{A}_{\mathcal{I}\mathcal{S}}\mathsf{T}\| \le \epsilon \|\mathsf{A}_{\mathcal{I}\mathcal{J}}\|,$$

or equivalently, assuming $\mathsf{A}_{\mathcal{I}\mathcal{J}} = [\ \mathsf{A}_{\mathcal{I}\mathcal{R}} \quad \mathsf{A}_{\mathcal{I}\mathcal{S}}\ ]$,

$$\left\|\mathsf{A}_{\mathcal{I}\mathcal{J}} - \mathsf{A}_{\mathcal{I}\mathcal{S}}[\ \mathsf{T} \quad \mathsf{I}\ ]\right\| \le \epsilon \|\mathsf{A}_{\mathcal{I}\mathcal{J}}\|.$$

In other words, the redundant columns are approximated as a linear combination of the skeleton columns to within the prescribed relative accuracy, leading to a low-rank factorization of $\mathsf{A}_{\mathcal{I}\mathcal{J}}$.

Note that while the ID error bound can be attained trivially by taking $\mathcal{S} = \mathcal{J}$, it is desirable to keep $|\mathcal{S}|$ as small as possible. The typical algorithm to compute an ID uses a strong rank-revealing QR factorization as detailed by Gu and Eisenstat [16], though in practice a standard greedy column-pivoted QR tends to be sufficient. In either case, the computational complexity is $O(|\mathcal{I}| \cdot |\mathcal{J}|^2)$.

3. The strong recursive skeletonization factorization.

We work in the context of a given tree decomposition of the domain (quadtree or octree) such that each leaf-level box of the tree contains a bounded number of DOFs independent of N. When the tree is uniformly refined (i.e., each box has either 0 or 2^d children and all leaf boxes are at the same level), it is straightforward to use the strong admissibility criterion illustrated in Figure 1 to identify which pairs of boxes are strongly admissible and which are not, as in Figure 2: given a box, the *near-field* region for that box is the region corresponding to adjacent boxes at the same level of tree, and the *far-field* region is the remainder of the domain. The case where the tree is not uniformly refined is similar, though we will illustrate our method with uniform trees in what follows.

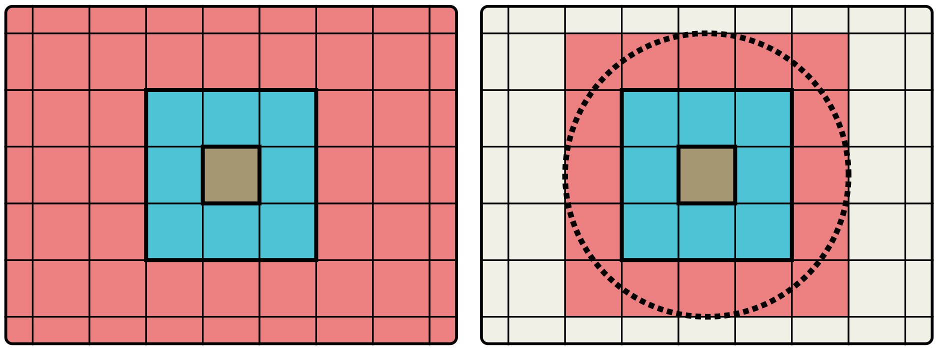


FIG. 2. *Considering the DOFs $\mathcal{B}$ interior to the brown box on the left, the near-field DOFs $\mathcal{N}$ are those interior to the blue boxes and the far-field DOFs $\mathcal{F}$ are those interior to the red boxes. Note that not all boxes in the far-field are shown. On the right, we draw a "proxy surface" Γ (the dashed circle) around the DOFs $\mathcal{B}$ such that the far-field $\mathcal{F}$ can be further decomposed into DOFs belonging to boxes inside the proxy surface, $\mathcal{O}$ (still red), and DOFs belonging to boxes outside the proxy surface, $\mathcal{P}$.*

3.1. Strong skeletonization. Given a matrix $\mathsf{A} \in \mathbb{C}^{N\times N}$ indexed by points in our domain, we begin by selecting DOFs $\mathcal{B}$ corresponding to a leaf-level box at the finest level of the tree.

Letting $\mathcal{N}$ and $\mathcal{F}$ be the sets of near- and far-field DOFs as in Figure 2 (left), an appropriate permutation P gives the block structure

$$\mathsf{P}^*\mathsf{A}\mathsf{P} = \left[\begin{array}{c|c|c} \mathsf{A}_{\mathcal{B}\mathcal{B}} & \mathsf{A}_{\mathcal{B}\mathcal{N}} & \mathsf{A}_{\mathcal{B}\mathcal{F}} \\ \hline \mathsf{A}_{\mathcal{N}\mathcal{B}} & \mathsf{A}_{\mathcal{N}\mathcal{N}} & \mathsf{A}_{\mathcal{N}\mathcal{F}} \\ \hline \mathsf{A}_{\mathcal{F}\mathcal{B}} & \mathsf{A}_{\mathcal{F}\mathcal{N}} & \mathsf{A}_{\mathcal{F}\mathcal{F}} \end{array}\right].$$

By assumption, the blocks $\mathsf{A}_{\mathcal{B}\mathcal{F}}$ and $\mathsf{A}_{\mathcal{F}\mathcal{B}}$ corresponding to the far-field interactions of $\mathcal{B}$ are numerically low-rank and thus compressible. Given some tolerance ϵ, we partition $\mathcal{B}$ into its redundant and skeleton DOFs $\mathcal{B} = \mathcal{R} \cup \mathcal{S}$ via the ID

$$\begin{bmatrix} \mathsf{A}_{\mathcal{F}\mathcal{B}} \\ \mathsf{A}^*_{\mathcal{B}\mathcal{F}} \end{bmatrix} = \begin{bmatrix} \mathsf{A}_{\mathcal{F}\mathcal{R}} & \mathsf{A}_{\mathcal{F}\mathcal{S}} \\ \mathsf{A}^*_{\mathcal{R}\mathcal{F}} & \mathsf{A}^*_{\mathcal{S}\mathcal{F}} \end{bmatrix} \approx \begin{bmatrix} \mathsf{A}_{\mathcal{F}\mathcal{S}} \\ \mathsf{A}^*_{\mathcal{S}\mathcal{F}} \end{bmatrix} \begin{bmatrix} \mathsf{T} & \mathsf{I} \end{bmatrix}, \tag{6}$$

which yields a skeleton set $\mathcal{S}$ and interpolation matrix T that can be used to represent both the columns of $\mathsf{A}_{\mathcal{F}\mathcal{B}}$ and the rows of $\mathsf{A}_{\mathcal{B}\mathcal{F}}$, i.e., $\mathsf{A}_{\mathcal{F}\mathcal{R}} \approx \mathsf{A}_{\mathcal{F}\mathcal{S}}\mathsf{T}$ and $\mathsf{A}_{\mathcal{R}\mathcal{F}} \approx \mathsf{T}^*\mathsf{A}_{\mathcal{S}\mathcal{F}}$. Note that in (6) we have assumed for clarity of exposition that the redundant DOFs $\mathcal{R}$ are ordered first within $\mathcal{B}$ such that no further permutation is necessary. We now partition blocks of $\mathsf{P}^*\mathsf{A}\mathsf{P}$ according to this ID to obtain

$$\mathsf{P}^*\mathsf{A}\mathsf{P} \approx \left[\begin{array}{cc|c|c} \mathsf{A}_{\mathcal{R}\mathcal{R}} & \mathsf{A}_{\mathcal{R}\mathcal{S}} & \mathsf{A}_{\mathcal{R}\mathcal{N}} & \mathsf{T}^*\mathsf{A}_{\mathcal{S}\mathcal{F}} \\ \mathsf{A}_{\mathcal{S}\mathcal{R}} & \mathsf{A}_{\mathcal{S}\mathcal{S}} & \mathsf{A}_{\mathcal{S}\mathcal{N}} & \mathsf{A}_{\mathcal{S}\mathcal{F}} \\ \hline \mathsf{A}_{\mathcal{N}\mathcal{R}} & \mathsf{A}_{\mathcal{N}\mathcal{S}} & \mathsf{A}_{\mathcal{N}\mathcal{N}} & \mathsf{A}_{\mathcal{N}\mathcal{F}} \\ \hline \mathsf{A}_{\mathcal{F}\mathcal{S}}\mathsf{T} & \mathsf{A}_{\mathcal{F}\mathcal{S}} & \mathsf{A}_{\mathcal{F}\mathcal{N}} & \mathsf{A}_{\mathcal{F}\mathcal{F}} \end{array}\right].$$

Because of the explicit linear dependence between far-field blocks of the matrix, the redundant DOFs $\mathcal{R}$ can now be decoupled from the far-field DOFs $\mathcal{F}$ using elementary block row and column operations. Defining the elimination matrices

$$\mathsf{U}_\mathsf{T} \equiv \left[\begin{array}{cc|c|c} \mathsf{I} & -\mathsf{T}^* & & \\ & \mathsf{I} & & \\ \hline & & \mathsf{I} & \\ \hline & & & \mathsf{I} \end{array}\right], \quad \mathsf{L}_\mathsf{T} \equiv \left[\begin{array}{cc|c|c} \mathsf{I} & & & \\ -\mathsf{T} & \mathsf{I} & & \\ \hline & & \mathsf{I} & \\ \hline & & & \mathsf{I} \end{array}\right],$$

we see that application of these operators on the left and right gives

$$\mathsf{U_T}\left[\begin{array}{cc|c|c} \mathsf{A}_{\mathcal{RR}} & \mathsf{A}_{\mathcal{RS}} & \mathsf{A}_{\mathcal{RN}} & \mathsf{T^*A}_{\mathcal{SF}} \\ \mathsf{A}_{\mathcal{SR}} & \mathsf{A}_{\mathcal{SS}} & \mathsf{A}_{\mathcal{SN}} & \mathsf{A}_{\mathcal{SF}} \\ \hline \mathsf{A}_{\mathcal{NR}} & \mathsf{A}_{\mathcal{NS}} & \mathsf{A}_{\mathcal{NN}} & \mathsf{A}_{\mathcal{NF}} \\ \hline \mathsf{A}_{\mathcal{FS}}\mathsf{T} & \mathsf{A}_{\mathcal{FS}} & \mathsf{A}_{\mathcal{FN}} & \mathsf{A}_{\mathcal{FF}} \end{array}\right]\mathsf{L_T} = \left[\begin{array}{cc|c|c} \mathsf{X}_{\mathcal{RR}} & \mathsf{X}_{\mathcal{RS}} & \mathsf{X}_{\mathcal{RN}} & \\ \mathsf{X}_{\mathcal{SR}} & \mathsf{A}_{\mathcal{SS}} & \mathsf{A}_{\mathcal{SN}} & \mathsf{A}_{\mathcal{SF}} \\ \hline \mathsf{X}_{\mathcal{NR}} & \mathsf{A}_{\mathcal{NS}} & \mathsf{A}_{\mathcal{NN}} & \mathsf{A}_{\mathcal{NF}} \\ \hline & \mathsf{A}_{\mathcal{FS}} & \mathsf{A}_{\mathcal{FN}} & \mathsf{A}_{\mathcal{FF}} \end{array}\right],$$

where the modified nonzero blocks marked with X correspond to some mixing of the second row and column, respectively, with the first row and column as a consequence of the elimination.

Using $\mathsf{X}_{\mathcal{RR}}$ as a pivot block to eliminate the other blocks in the first row and column (i.e., performing block elimination as in subsection 2.1 with $\mathcal{I} = \mathcal{R}$, $\mathcal{J} = \mathcal{S}\cup\mathcal{N}$, and $\mathcal{K} = \mathcal{F}$) we define the corresponding matrices L and U as in (4) to obtain

$$\mathsf{LU_TP^*APL_TU} \approx \left[\begin{array}{cc|c|c} \mathsf{X}_{\mathcal{RR}} & & & \\ & \mathsf{X}_{\mathcal{SS}} & \mathsf{X}_{\mathcal{SN}} & \mathsf{A}_{\mathcal{SF}} \\ \hline & \mathsf{X}_{\mathcal{NS}} & \mathsf{X}_{\mathcal{NN}} & \mathsf{A}_{\mathcal{NF}} \\ \hline & \mathsf{A}_{\mathcal{FS}} & \mathsf{A}_{\mathcal{FN}} & \mathsf{A}_{\mathcal{FF}} \end{array}\right] \equiv \mathsf{Z}\left(\mathsf{A};\mathcal{B}\right), \tag{7}$$

whereupon we see that the redundant DOFs $\mathcal{R}$ are now completely decoupled from the rest of the problem.

We refer to this process as *strong skeletonization* of A with respect to the DOFs $\mathcal{B}$, as it is a direct modification of the skeletonization procedure of Martinsson and Rokhlin [25] for the strong admissibility setting using the multiplicative formulation of Ho and Ying [22, section 3]. We note that while ID-based compression of far-field interactions has been used in the context of kernel-independent FMMs [26, 30], its use for the construction of (approximate) direct solvers is novel.

For a purely notational convenience, we will define the left and right skeletonization operators V and W as

$$\mathsf{V} \equiv \mathsf{PU_T^{-1}L^{-1}}, \quad \mathsf{W} \equiv \mathsf{U^{-1}L_T^{-1}P^*}, \tag{8}$$

with the understanding that these matrices will always be stored and used in the factored form given for efficiency. In particular, recall that the block unit-triangular matrices U, $\mathsf{U_T}$, and so on may be inverted by toggling the sign of the nonzero off-diagonal block. With this shorthand we obtain

$$\mathsf{Z}\left(\mathsf{A};\mathcal{B}\right) \approx \mathsf{V^{-1}AW^{-1}}, \tag{9}$$

a more compact representation of $\mathsf{Z}\left(\mathsf{A};\mathcal{B}\right)$ in (7).

3.2. The use of a proxy surface. For optimal complexity it is desirable to avoid computation with blocks indexed by $\mathcal{F}$ in the construction of $\mathsf{Z}\left(\mathsf{A};\mathcal{B}\right)$, since $|\mathcal{F}|$ is in general large. By design, the only part of computing $\mathsf{Z}\left(\mathsf{A};\mathcal{B}\right)$ that involves blocks indexed by $\mathcal{F}$ is constructing the ID, that is, finding the partition $\mathcal{B} = \mathcal{R}\cup\mathcal{S}$ and the interpolation matrix T in (6). For simplicity, we will drop the block $\mathsf{A}^*_{\mathcal{BF}}$ in this section and explain how the subblock $\mathsf{A}_{\mathcal{FB}}$ can be compressed in an indirect way that is more efficient than operating on $\mathsf{A}_{\mathcal{FB}}$ directly. The "transpose" of these ideas can be used for the full stacked matrix including $\mathsf{A}^*_{\mathcal{BF}}$.

For concreteness, consider the case where $\mathsf{A} \equiv \mathsf{K}$ is a discretization such as (3) with $b(x) \equiv c(x) \equiv 1$ and the 2D Laplace kernel $K(z) = -\frac{1}{2\pi}\log(\|z\|)$. In this case, entries of $\mathsf{A}_{\mathcal{FB}}$ are given (up to a factor of $1/N$, which we drop in our discussion) directly by $K(x_i - x_j)$ for $x_i \in \mathcal{F}$ and $x_j \in \mathcal{B}$.

Suppose Γ is a circle (or sphere in three dimensions) enclosing the DOF set $\mathcal{B}$ with radius normalized to a multiple of $5/2$ the sidelength of the corresponding box. As in Figure 2 (right), we partition the far-field DOFs of $\mathcal{B}$ as $\mathcal{F} = \mathcal{O} \cup \mathcal{P}$, where

$$\mathcal{P} \equiv \{i \in \mathcal{F} |\ i \text{ is contained in a box entirely outside of } \Gamma\}$$

and $\mathcal{O} \equiv \mathcal{F} \setminus \mathcal{P}$, i.e., $\mathcal{O}$ is the set of indices corresponding to the square of boxes containing the proxy surface in the figure.

Define $\phi_i(x) \equiv K(x_i - x)$ for $x_i \in \mathcal{P}$ such that $\phi_i(x_j)$ is the "incoming" harmonic field generated at $x_j \in \mathcal{B}$ by a source at x_i. Because the DOFs in $\mathcal{B}$ are contained in the closed region with boundary Γ and the DOFs in $\mathcal{P}$ are contained in the complementary region, we may (under mild assumptions [27]) use a form of Green's identity to write $\phi_i(x_j)$ for any $x_j \in \mathcal{B}$ in terms of a density $\psi_i(y)$ on Γ as

$$\phi_i(x_j) = \int_\Gamma \psi_i(y) K(x_j - y)\, dy, \tag{10}$$

where $\psi_i(y)$ depends on x_i but not on x_j.

Because $\phi_i(x_j) = K(x_i - x_j)$ for $x_i \in \mathcal{P} \subset \mathcal{F}$ and $x_j \in \mathcal{B}$, the left-hand side of (10) is an entry of $\mathsf{A}_{\mathcal{F}\mathcal{B}}$. The right-hand side is a so-called single-layer representation of this entry. Discretizing this representation by replacing the analytic integral over Γ with numerical integration using n_p points $y_1, \dots, y_{n_p}$, we see that, up to discretization error,

$$\mathsf{A}_{\mathcal{F}\mathcal{B}} = \begin{bmatrix} \mathsf{A}_{\mathcal{O}\mathcal{B}} \\ \mathsf{A}_{\mathcal{P}\mathcal{B}} \end{bmatrix} \approx \begin{bmatrix} \mathsf{I} & \\ & \mathsf{M}_{\mathcal{P}\Gamma} \end{bmatrix} \begin{bmatrix} \mathsf{A}_{\mathcal{O}\mathcal{B}} \\ \mathsf{G}_{\Gamma\mathcal{B}} \end{bmatrix}, \tag{11}$$

where $\mathsf{G}_{\Gamma\mathcal{B}}$ has entries $K(x_j - y_i)$ for $x_j \in \mathcal{B}$ and $\mathsf{M}_{\mathcal{P}\Gamma}$ is the matrix that approximately maps $\mathsf{G}_{\Gamma\mathcal{B}}$ to $\mathsf{A}_{\mathcal{P}\mathcal{B}}$ via a discretization of (10) for each $x_i \in \mathcal{P}$. We do not give an explicit form of $\mathsf{M}_{\mathcal{P}\Gamma}$, as we need only be assured of its existence for what follows.

The *proxy trick* for accelerated compression, which is heavily employed in the literature [25, 7, 12, 8, 13, 21, 22, 30, 33, 26], makes use of two key observations regarding (11). First, if $n_p \ll |\mathcal{P}|$ (for example, if we take $n_p = O(1)$ and N to be large), then it is relatively inexpensive to compute the ID

$$\begin{bmatrix} \mathsf{A}_{\mathcal{O}\mathcal{B}} \\ \mathsf{G}_{\Gamma\mathcal{B}} \end{bmatrix} = \begin{bmatrix} \mathsf{A}_{\mathcal{O}\mathcal{R}} & \mathsf{A}_{\mathcal{O}\mathcal{S}} \\ \mathsf{G}_{\Gamma\mathcal{R}} & \mathsf{G}_{\Gamma\mathcal{S}} \end{bmatrix} \approx \begin{bmatrix} \mathsf{A}_{\mathcal{O}\mathcal{S}} \\ \mathsf{G}_{\Gamma\mathcal{S}} \end{bmatrix} \begin{bmatrix} \mathsf{T} & \mathsf{I} \end{bmatrix}. \tag{12}$$

Furthermore, because Γ is in the far-field of $\mathcal{B}$, $|\mathcal{S}|$ should be small by assumption. Second, using the discrete Green's identity represented by $\mathsf{M}_{\mathcal{P}\Gamma}$, combining (11) and (12) yields

$$\mathsf{A}_{\mathcal{F}\mathcal{B}} \approx \begin{bmatrix} \mathsf{I} & \\ & \mathsf{M}_{\mathcal{P}\Gamma} \end{bmatrix} \begin{bmatrix} \mathsf{A}_{\mathcal{O}\mathcal{S}} \\ \mathsf{G}_{\Gamma\mathcal{S}} \end{bmatrix} \begin{bmatrix} \mathsf{T} & \mathsf{I} \end{bmatrix} \approx \mathsf{A}_{\mathcal{F}\mathcal{S}} \begin{bmatrix} \mathsf{T} & \mathsf{I} \end{bmatrix},$$

i.e., the partitioning $\mathcal{B} = \mathcal{R} \cup \mathcal{S}$ and interpolation matrix T in (12) also give an ID of $\mathsf{A}_{\mathcal{F}\mathcal{B}}$. We caution that the amplification of the approximation error in this ID depends on $\|\mathsf{M}_{\mathcal{P}\Gamma}\|$, among other factors, but note that this does not appear to be an issue in practice (see section 4).

By using these ideas or modifications thereof to obtain the ID (6) as opposed to using a rank-revealing QR on the far-field blocks directly, the complexity of compression now depends on the number of proxy points n_p and not on the total number of points in the domain and is thus substantially reduced. To apply this acceleration, we require only a way to evaluate kernel interactions with Γ and that the entries of $\mathsf{A}_{\mathcal{P}\mathcal{B}}$ satisfy a Green's identity. While we have discussed only the Laplace kernel, the same trick holds for, e.g., the Stokes, or elasticity kernel. A more thorough discussion of the use of a proxy surface can be found in Ho and Ying [22, subsection 3.3], which discusses the case of more general $a(x)$ and $b(x)$ in (1).

3.3. Algorithm and complexity. We turn now to a 1D sketch of our factorization approach using strong skeletonization. Suppose we wish to solve an integral equation over the unit line segment $\Omega = [0, 1]$ using the trapezoid rule to construct the matrix K. Partitioning Ω into eight subintervals with corresponding DOF sets $\mathcal{B}_i$ for $i = 1, \ldots, 8$, we consider the first DOF set $\mathcal{B}_1$ and its corresponding near-field DOFs $\mathcal{N}_1$ and far-field DOFs $\mathcal{F}_1$. This gives a block partitioning and labeling of K as in Figure 3 (top left). We use the strong skeletonization algorithm of subsection 3.1 to decouple redundant DOFs in $\mathcal{B}_1$ to approximately obtain the new matrix

$$\mathsf{Z}(\mathsf{K}; \mathcal{B}_1) = \left[\begin{array}{cc|c|c} \mathsf{X}_{\mathcal{R}_1\mathcal{R}_1} & & & \\ & \mathsf{X}_{\mathcal{S}_1\mathcal{S}_1} & \mathsf{X}_{\mathcal{S}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{S}_1\mathcal{F}_1} \\ \cline{2-4} & \mathsf{X}_{\mathcal{N}_1\mathcal{S}_1} & \mathsf{X}_{\mathcal{N}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{N}_1\mathcal{F}_1} \\ \cline{2-4} & \mathsf{K}_{\mathcal{F}_1\mathcal{S}_1} & \mathsf{K}_{\mathcal{F}_1\mathcal{N}_1} & \mathsf{K}_{\mathcal{F}_1\mathcal{F}_1} \end{array}\right] \tag{13}$$

as in (9). We call the redundant DOFs $\mathcal{R}_1$ *inactive*, because they no longer play an active role in our factorization procedure. In contrast, any DOFs that have not yet been decoupled will be referred to as *active*.

Moving on to the next set of DOFs $\mathcal{B}_2$ with near-field DOFs $\mathcal{N}_2$ and far-field DOFs $\mathcal{F}_2$, we observe that most of the nonzero entries of the matrix $\mathsf{Z}(\mathsf{K}; \mathcal{B}_1)$ are unchanged from their original value in K; see Figure 3 (top right). It is therefore reasonable to perform strong skeletonization of this new matrix with respect to $\mathcal{B}_2$ and expect compression of the corresponding far-field interactions. This renders the redundant DOFs $\mathcal{R}_2$ inactive and yields the new matrix

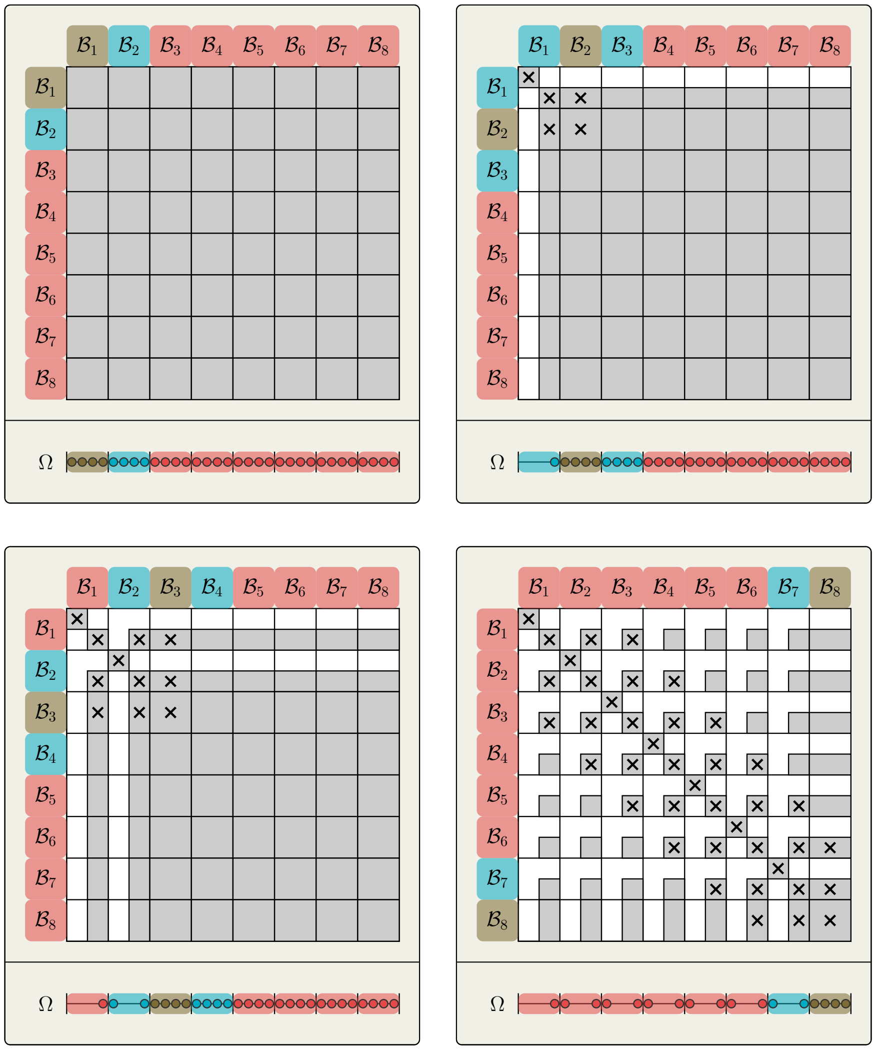
$$\mathsf{Z}(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2) \equiv \mathsf{Z}(\mathsf{Z}(\mathsf{K}; \mathcal{B}_1); \mathcal{B}_2),$$

where we define $\mathsf{Z}(\mathsf{K}; \mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_k)$ to be the result of skeletonizing K with respect to the DOFs $\mathcal{I}_1$, skeletonizing the result with respect to $\mathcal{I}_2$, and so on. In successive steps of strong skeletonization we skeletonize $\mathsf{Z}(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2)$ with respect to the DOFs $\mathcal{B}_3$ through $\mathcal{B}_8$ as in Figure 3 (bottom).

After skeletonization with respect to $\mathcal{B}_8$, the matrix $\mathsf{Z}(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_8)$ has many diagonal blocks corresponding to completely decoupled redundant DOF sets $\mathcal{R}_i$ for $i = 1, \ldots, 8$ as well as many blocks corresponding to interactions between the remaining active skeleton DOF sets $\mathcal{S}_i$. We construct a new partitioning of Ω into four subintervals and define the DOF sets

$$\mathcal{B}_9 \equiv \mathcal{S}_1 \cup \mathcal{S}_2, \quad \mathcal{B}_{10} \equiv \mathcal{S}_3 \cup \mathcal{S}_4, \quad \mathcal{B}_{11} \equiv \mathcal{S}_5 \cup \mathcal{S}_6, \quad \mathcal{B}_{12} \equiv \mathcal{S}_7 \cup \mathcal{S}_8. \tag{14}$$

Permuting $\mathsf{Z}(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_8)$ such that these DOF sets are contiguous with the inactive DOF sets $\mathcal{R}_i$ for $i = 1, \ldots, 8$ permuted to the end for visualization purposes, we obtain a matrix as in Figure 4 (left), at which point we may skeletonize successively with respect to $\mathcal{B}_9$ through $\mathcal{B}_{12}$. It is not possible to again double the size of a subinterval and expose any further compressible blocks, so we stop. The final matrix $\mathsf{Z}(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_{12})$ can be permuted to a block-diagonal matrix with small blocks defined by the DOF sets $\mathcal{R}_i$ for $i = 1, \ldots, 12$ and the DOF set $\mathcal{S}_9 \cup \mathcal{S}_{10} \cup \mathcal{S}_{11} \cup \mathcal{S}_{12}$. This block-diagonal structure can then be exploited to efficiently solve the linear system (2).



$\mathcal{B}_1$ $\mathcal{B}_2$ $\mathcal{B}_3$ $\mathcal{B}_4$ $\mathcal{B}_5$ $\mathcal{B}_6$ $\mathcal{B}_7$ $\mathcal{B}_8$
Ω

FIG. 3. *We partition the* 1*D domain* Ω *into eight subintervals and block the corresponding matrix* K *accordingly. Identifying the DOFs* $\mathcal{B}_1$ *(brown),* $\mathcal{N}_1$ *(blue), and* $\mathcal{F}_1$ *(red) in the top-left figure, we skeletonize with respect to* $\mathcal{B}_1$*. Using the same color scheme for box, near-field, and far-field DOFs in the top-right figure, we see that some redundant DOFs have been completely decoupled from the rest and some blocks of the matrix have been modified through Schur complement updates (blocks marked with "X"). We proceed to skeletonize with respect to the DOFs* $\mathcal{B}_2$*, and then* $\mathcal{B}_3$ *(bottom-left figure) all the way through to* $\mathcal{B}_8$ *(bottom-right figure). Below each matrix, we show only the remaining active DOFs at that step, i.e., we do not show the redundant DOFs corresponding to decoupled diagonal blocks.*

3.3.1. The general case: First level. Having given the flavor of our approach in one dimension, we flesh out the details for the more general case. Suppose the integral equation (1) is discretized over $\Omega \subset \mathbb{R}^d$ for $d = 2$ or 3. Given a tree decomposition of the domain such that each leaf box contains a constant number of unknowns, we number the levels of the tree starting from the finest level ($\ell = 1$) up to the root level ($\ell = L$). We require a fixed but arbitrary bottom-up level-by-level traversal of the

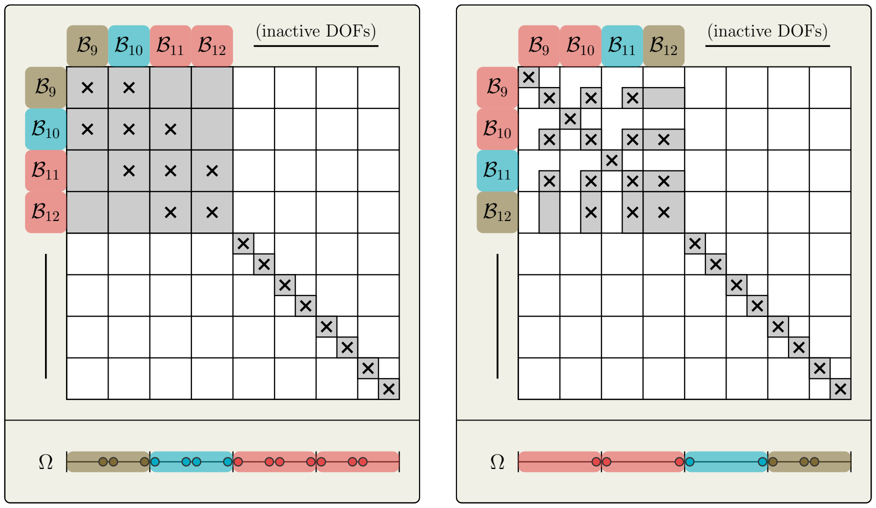


FIG. 4. *After skeletonization with respect to the DOFs $\mathcal{B}_8$ in Figure 3, we permute all decoupled redundant DOF sets $\mathcal{R}_1, \dots, \mathcal{R}_8$ to the end and define the next-level DOF sets $\mathcal{B}_9, \dots, \mathcal{B}_{12}$ as in (14) such that blocks of the matrix in the left figure correspond to aggregating blocks from the previous level. Using the same color scheme as in Figure 3, in skeletonizing with respect to $\mathcal{B}_9$, we see that blocks of interactions between $\mathcal{B}_9$ (brown) and $\mathcal{N}_9$ (blue) have modifications from Schur complement updates (marked with "X") from the previous level. We continue to skeletonize with respect to DOF sets at this level up through to $\mathcal{B}_{12}$ (right figure), decoupling additional redundant DOFs as we go.*

tree and order the boxes accordingly such that a box at level 1 is ordered before any box at level 2 and so on. This ordering on all boxes of the tree induces corresponding orderings on the boxes within each level of the tree, $\mathscr{L}_\ell$ for $\ell = 1, \dots, L$. For example, in the case of a regular grid with $2^{d(L-\ell)}$ boxes at level ℓ we obtain the orderings

$$\mathscr{L}_1 = \left\{1,\, 2,\, \dots,\, 2^{d(L-1)}\right\},$$
$$\mathscr{L}_2 = \left\{2^{d(L-1)} + 1,\, 2^{d(L-1)} + 2,\, \dots,\, 2^{d(L-1)} + 2^{d(L-2)}\right\},$$

and so on. We do not require a regular grid of discretization points but use a regular grid in all figures for illustration. Note in particular that this implies leaf boxes may belong to levels other than $\ell = 1$ in the general case.

Beginning at level $\ell = 1$, we select the first leaf box and label the corresponding DOFs as $\mathcal{B}_1$ with near-field DOFs $\mathcal{N}_1$ and far-field DOFs $\mathcal{F}_1$. We decouple and render inactive redundant DOFs in $\mathcal{B}_1$ through strong skeletonization to obtain

$$\mathsf{V}_1^{-1}\mathsf{K}\mathsf{W}_1^{-1} \approx \mathsf{Z}\left(\mathsf{K}; \mathcal{B}_1\right)$$

with $\mathsf{Z}\left(\mathsf{K}; \mathcal{B}_1\right)$ as in (13) and V_1 and W_1 the left and right skeletonization operators corresponding to $\mathcal{B}_1$ as in (8).

Selecting the next box with corresponding DOFs $\mathcal{B}_2$, we define the DOF sets $\mathcal{N}_2$ and $\mathcal{F}_2$ as

$$\mathcal{N}_2 \equiv \{\textit{active}\ \text{DOFs in the near-field of}\ \mathcal{B}_2\},$$
$$\mathcal{F}_2 \equiv \{\textit{active}\ \text{DOFs in the far-field of}\ \mathcal{B}_2\}$$

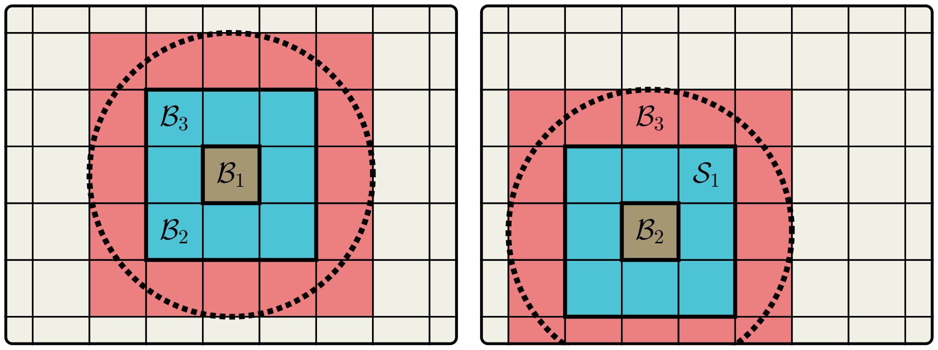


FIG. 5. *Left: Because the DOF sets $\mathcal{B}_2$ and $\mathcal{B}_3$ are both in the near-field of $\mathcal{B}_1$, skeletonization with respect to $\mathcal{B}_1$ leads to updated interactions between $\mathcal{B}_2$ and $\mathcal{B}_3$ as a subblock of $\mathsf{X}_{\mathcal{N}_1\mathcal{N}_1}$ in* (13). *Right: Considering now the next box, we see the DOF set $\mathcal{B}_3$ is in the far-field of $\mathcal{B}_2$, so the previous Schur complement update has led to modified far-field interactions that must be compressed when skeletonizing with respect to $\mathcal{B}_2$. However, interactions between $\mathcal{B}_2$ and DOFs corresponding to boxes outside the proxy surface Γ are still unmodified due to geometric considerations as guaranteed by Theorem* 3.1.

so as to avoid unnecessary further computation with the inactive DOFs indexing the first block row and column of $\mathsf{Z}(\mathsf{K};\mathcal{B}_1)$. To efficiently perform strong skeletonization of $\mathsf{A} = \mathsf{Z}(\mathsf{K};\mathcal{B}_1)$ with respect to the DOFs $\mathcal{B}_2$, it is necessary to assume that the blocks $\mathsf{A}_{\mathcal{B}_2\mathcal{F}_2}$ and $\mathsf{A}_{\mathcal{F}_2\mathcal{B}_2}$ are still compressible. Note that this is not immediate, as *these blocks need not be original blocks of* K. In particular, if $\mathcal{B}_2 \subset \mathcal{N}_1$ and $\mathcal{F}_2 \cap \mathcal{N}_1 \neq \emptyset$, then the block $\mathsf{X}_{\mathcal{N}_1\mathcal{N}_1}$ in (13) includes updated interactions between $\mathcal{B}_2$ and $\mathcal{F}_2 \cap \mathcal{N}_1$, which has the potential to increase the numerical rank of said interactions. This is illustrated in Figure 5.

Note that, drawing the proxy surface Γ around $\mathcal{B}_2$, it is still true due to geometric considerations that the interactions $\mathsf{A}_{\mathcal{P}_2\mathcal{B}_2}$ between $\mathcal{B}_2$ and far-field points $\mathcal{P}_2$ contained in boxes outside of Γ are unchanged and, therefore, the accelerated compression scheme in subsection 3.2 using a proxy surface is still justified.

THEOREM 3.1. *Skeletonization with respect to the DOFs $\mathcal{B}_i$ does not modify interactions between $\mathcal{B}_j$ and $\mathcal{P}_j$ for $j \geq i$. That is, if $\mathsf{A} = \mathsf{Z}(\mathsf{K};\mathcal{B}_1,\mathcal{B}_2,\ldots,\mathcal{B}_i)$, then*

$$\begin{bmatrix} \mathsf{A}_{\mathcal{P}_j\mathcal{B}_j} \\ \mathsf{A}^*_{\mathcal{B}_j\mathcal{P}_j} \end{bmatrix} = \begin{bmatrix} \mathsf{K}_{\mathcal{P}_j\mathcal{B}_j} \\ \mathsf{K}^*_{\mathcal{B}_j\mathcal{P}_j} \end{bmatrix}.$$

Proof. Suppose D is the sidelength of the box with corresponding DOFs $\mathcal{B}_i$ and consider skeletonizing A with respect to $\mathcal{B}_i$. As in (13), we see the only updated interactions between active DOFs are between $\mathcal{S}_i$ and $\mathcal{N}_i$. However, by definition of $\mathcal{P}_j$ we know that $\mathcal{B}_j$ and $\mathcal{P}_j$ correspond to DOF sets separated by at least two boxes of sidelength $D' \geq D$ due to the fact that $j \geq i$ and our ordering corresponds to a bottom-up traversal of the tree. Therefore, since the near-field DOFs $\mathcal{N}_i$ span a distance of no more than $3D$ in each axial direction, either $\mathcal{P}_j \cap (\mathcal{S}_i \cup \mathcal{N}_i) = \emptyset$ or $\mathcal{B}_j \cap (\mathcal{S}_i \cup \mathcal{N}_i) = \emptyset$, which implies skeletonization with respect to $\mathcal{B}_i$ did not modify interactions between $\mathcal{B}_j$ and $\mathcal{P}_j$. ∎

With the above in mind, we skeletonize A with respect to $\mathcal{B}_2$ and obtain

$$\mathsf{V}_2^{-1}\mathsf{A}\mathsf{W}_2^{-1} \approx \mathsf{Z}(\mathsf{A};\mathcal{B}_2) = \mathsf{Z}(\mathsf{K};\mathcal{B}_1,\mathcal{B}_2)$$

with

$$\mathsf{Z}\left(\mathsf{A};\mathcal{B}_2\right) = \left[\begin{array}{c|cc|c|c} \mathsf{X}_{\mathcal{R}_1\mathcal{R}_1} & & & & \\ \hline & \mathsf{X}_{\mathcal{R}_2\mathcal{R}_2} & & & \\ & & \mathsf{X}_{\mathcal{S}_2\mathcal{S}_2} & \mathsf{X}_{\mathcal{S}_2\mathcal{N}_2} & \mathsf{A}_{\mathcal{S}_2\mathcal{F}_2} \\ \hline & & \mathsf{X}_{\mathcal{N}_2\mathcal{S}_2} & \mathsf{X}_{\mathcal{N}_2\mathcal{N}_2} & \mathsf{A}_{\mathcal{N}_2\mathcal{F}_2} \\ \hline & & \mathsf{A}_{\mathcal{F}_2\mathcal{S}_2} & \mathsf{A}_{\mathcal{F}_2\mathcal{N}_2} & \mathsf{A}_{\mathcal{F}_2\mathcal{F}_2} \end{array}\right],$$

where the DOFs $\mathcal{R}_2$ have been made inactive as well. We note that while the nonzero entries in the last block row and block column are unmodified from what they were in $\mathsf{A} = \mathsf{Z}\left(\mathsf{K};\mathcal{B}_1\right)$, this does not necessarily mean they are unmodified from what they were in K. For example, consider that in Figure 5 we have $\mathcal{S}_1 \subset \mathcal{N}_2$ and $\mathcal{B}_3 \subset \mathcal{F}_2$, but skeletonization with respect to $\mathcal{B}_1$ led to modified interactions between $\mathcal{S}_1$ and $\mathcal{B}_3$.

Proceeding as in the 1D case, we loop over each box at level $\ell = 1$, identify its corresponding DOFs $\mathcal{B}_i$ and active near- and far-field DOFs $\mathcal{N}_i$ and $\mathcal{F}_i$, and perform strong skeletonization using the proxy trick to capture interactions with $\mathcal{P}_i$ implicitly. This process can be seen in Figure 6, where the subfigures show all active DOFs

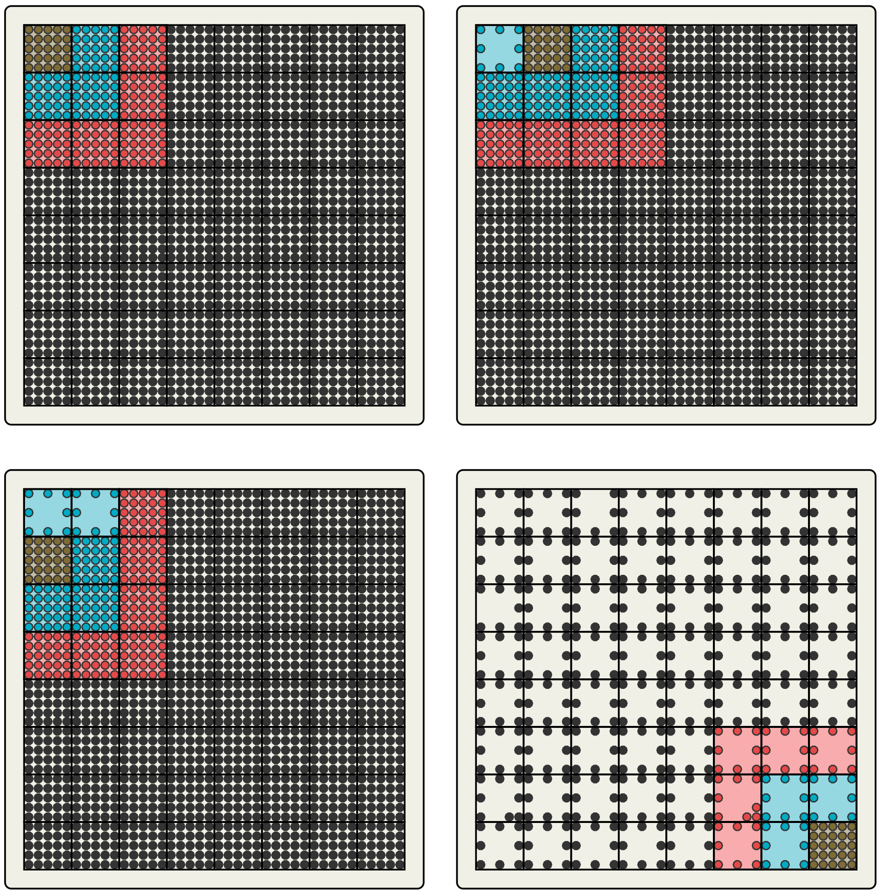


FIG. 6. *Considering an integral equation discretized uniformly over the 2D domain $\Omega = [0,1]^2$, we display the active DOFs at the point of skeletonization with respect to $\mathcal{B}_1$ (top left), $\mathcal{B}_2$ (top right), $\mathcal{B}_3$ (bottom left), and $\mathcal{B}_{64}$ (bottom right). When skeletonizing with respect to $\mathcal{B}_i$ (brown DOFs) the near-field DOFs $\mathcal{N}_i$ are colored blue and the far-field DOFs $\mathcal{O}_i$ corresponding to boxes inside the proxy surface are colored red. Note that the full set of far-field DOFs $\mathcal{F}_i$ includes not just the red DOFs but also all gray DOFs.*

at various times during the skeletonization at level $\ell = 1$ of a regular discretization over the unit square. Supposing that there are r boxes at level $\ell = 1$ (i.e., $\mathscr{L}_1 = \{1, 2, \dots, r\} = [r]$), and defining $[r]' \equiv \{r, r-1, \dots, 1\}$ as the reversal of $[r]$, the resulting matrix at this point is

$$
\begin{aligned}
\mathsf{Z}\left(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_r\right) &\approx \left(\mathsf{V}_r^{-1} \dots \mathsf{V}_2^{-1} \mathsf{V}_1^{-1}\right) \mathsf{K} \left(\mathsf{W}_1^{-1} \mathsf{W}_2^{-1} \dots \mathsf{W}_r^{-1}\right) \\
&\equiv \left(\prod_{i \in [r]'} \mathsf{V}_i^{-1}\right) \mathsf{K} \left(\prod_{i \in [r]} \mathsf{W}_i^{-1}\right).
\end{aligned} \tag{15}
$$

Note that in $\mathsf{Z}\left(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_r\right)$ each redundant DOF set $\mathcal{R}_i$ is completely decoupled from the rest of the problem, but each skeleton DOF set $\mathcal{S}_i$ remains coupled to all other skeleton DOFs.

3.3.2. The general case: Subsequent levels. Having finished level $\ell = 1$ of the tree, we step up to the next level of the spatial hierarchy, wherein boxes are twice as large in each axial direction. Similar to our definitions of $\mathcal{F}_i$ and $\mathcal{N}_i$, for a level $\ell > 1$ we define the DOFs $\mathcal{B}_i$ of a box to be any *active* DOFs geometrically contained in that box, that is, if

$$
C_i \equiv \{j \mid \text{box } j \text{ is a child box of box } i\},
$$

then $\mathcal{B}_i \equiv \bigcup_{j \in C_i} \mathcal{S}_j$ (i.e., it is the union of the *skeleton* DOFs of its child boxes). With this definition in tow we state the following corollary of Theorem 3.1.

COROLLARY 3.2. *Suppose r is the number of the last DOF set at level $\ell \geq 1$. Then, at the beginning of level $\ell + 1$, all far-field interactions between active DOFs at level $\ell + 1$ are unmodified from their initial values in K. That is, if* $\mathsf{A} = \mathsf{Z}\left(\mathsf{K}; \mathcal{B}\right)_1, \mathcal{B}_2, \dots, \mathcal{B}_r$*, then*

$$
\begin{bmatrix} \mathsf{A}_{\mathcal{F}_j \mathcal{B}_j} \\ \mathsf{A}^*_{\mathcal{B}_j \mathcal{F}_j} \end{bmatrix} = \begin{bmatrix} \mathsf{K}_{\mathcal{F}_j \mathcal{B}_j} \\ \mathsf{K}^*_{\mathcal{F}_j \mathcal{B}_j} \end{bmatrix}
$$

for all $j > r$.

Corollary 3.2 tells us that while one might fear that Schur complement updates would propagate beyond interactions between $\mathcal{B}_j$ and $\mathcal{O}_j$ and thus require increasing the size of the proxy surface, the opposite is in fact true: at the beginning of a level, interactions with all of $\mathcal{F}_j$ are unmodified. Therefore, our argument in subsection 3.2 for the use of a proxy surface still holds and it is straightforward to loop over each box on level $\ell = 2$ and perform strong skeletonization with respect to the corresponding DOFs $\mathcal{B}_i$, as is visualized in Figure 7. We repeat level-by-level for $\ell = 3, 4, \dots, L-2$, noting that at level $\ell = L-1$ all boxes are adjacent and thus all sets of active far-field DOFs are empty so compression of this form is not possible.

3.3.3. The final factorization. Supposing that the last set of DOFs at level $\ell = L-2$ is $\mathcal{B}_n$, the matrix $\mathsf{A} = \mathsf{Z}\left(\mathsf{K}; \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_n\right)$ has the same form as (15), albeit with more factors. Defining $\mathcal{B}_t$ to be the set of all active DOFs remaining at this level of the tree, a last permutation to order the DOFs $\mathcal{B}_t$ contiguously yields the block-diagonal matrix

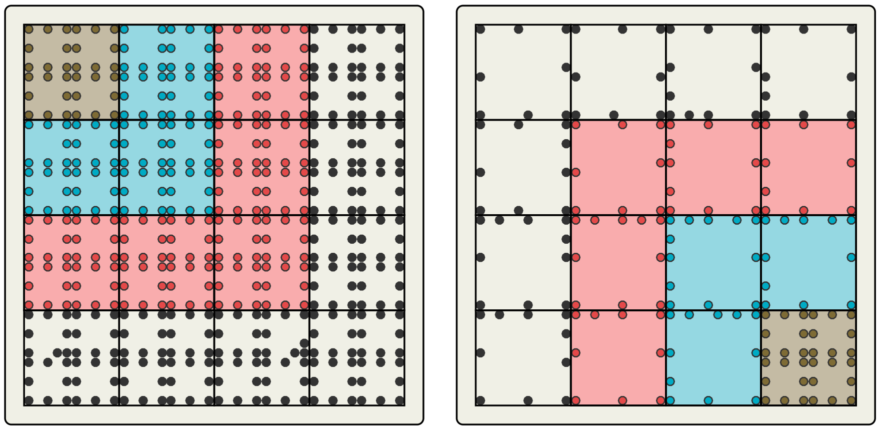


FIG. 7. *After skeletonizing with respect to $\mathcal{B}_{64}$ at the end of Figure 6, we skeletonize with respect to $\mathcal{B}_{65}$, the set of DOFs corresponding to the first larger box at the next level (left). Proceeding to skeletonize with respect to each box through to $\mathcal{B}_{80}$, the last box at this level, we see that further compression has been attained as the number of remaining active DOFs has been reduced substantially (right). All DOFs are colored as in Figure 6.*

$$\begin{aligned} \mathsf{D} \equiv & \begin{bmatrix} \mathsf{X}_{\mathcal{R}_1\mathcal{R}_1} & & & \\ & \ddots & & \\ & & \mathsf{X}_{\mathcal{R}_n\mathcal{R}_n} & \\ & & & \mathsf{A}_{\mathcal{B}_t\mathcal{B}_t} \end{bmatrix} \\ & \approx \mathsf{P}_t^* \left(\prod_{i\in[n]'} \mathsf{V}_i^{-1} \right) \mathsf{K} \left(\prod_{i\in[n]} \mathsf{W}_i^{-1} \right) \mathsf{P}_t. \end{aligned}$$

Rearranging, we obtain an approximate factorization as

$$\mathsf{F} \equiv \left(\prod_{i\in[n]} \mathsf{V}_i \right) \mathsf{P}_t \mathsf{D} \mathsf{P}_t^* \left(\prod_{i\in[n]'} \mathsf{W}_i \right) \approx \mathsf{K}, \tag{16}$$

which is the RS-S of K. The process of computing F is summarized in Algorithm 1.

Note that an approximate factorization of K^{-1} may be obtained directly as

$$\mathsf{F}^{-1} = \left(\prod_{i\in[n]} \mathsf{W}_i^{-1} \right) \mathsf{P}_t \mathsf{D}^{-1} \mathsf{P}_t^* \left(\prod_{i\in[n]'} \mathsf{V}_i^{-1} \right) \approx \mathsf{K}^{-1}, \tag{17}$$

though the approximation error will in general increase by a factor of the condition number of K. Further, in the case where K is positive definite, we have for all i that $\mathsf{V}_i = \mathsf{W}_i^*$. This means that, assuming our approximation is accurate enough that D still admits a square-root $\mathsf{D}^{1/2}$, we may obtain a generalized square-root $\mathsf{F}^{1/2}$ with $\mathsf{F} = \mathsf{F}^{1/2}(\mathsf{F}^{1/2})^*$ as

$$\mathsf{F}^{1/2} \equiv \left(\prod_{i\in[n]} \mathsf{V}_i \right) \mathsf{P}_t \mathsf{D}^{1/2},$$

Algorithm 1. The strong recursive skeletonization factorization.

```
1: // Initialize
2: $\mathsf{A} := \mathsf{K}$
3: for $\ell := 1$ to $L - 2$ do
4:     for each box $i \in \mathscr{L}_\ell$ do
5:         // Identify relevant DOFs for strong skeletonization
6:         $[\mathcal{B}_i, \mathcal{N}_i, \mathcal{F}_i] :=$ {active DOFs in box/near-field/far-field}
7:         // Perform strong skeletonization with respect to DOFs
8:         $\mathsf{A} := \mathsf{Z}(\mathsf{A}; \mathcal{B}_i) \approx \mathsf{V}_i^{-1} \mathsf{A} \mathsf{W}_i^{-1}$
9:     end for
10: end for
11: // Store middle block diagonal matrix and permutation
12: $\mathsf{D} := \mathsf{P}_t^* \mathsf{A} \mathsf{P}_t$
13: Output: $\mathsf{F}$ as in (16)
```

which differs from any generalized square-root $\mathsf{K}^{1/2}$ of K (e.g., the Cholesky factor) by a unitary matrix in the ideal case ($\epsilon = 0$ for all IDs).

In this setting, we may also compute an approximate log-determinant of K using the fact that $\log |\mathsf{D}| \approx \log |\mathsf{K}|$, which is useful for applications in statistics [28].

3.3.4. Complexity. As written, Algorithm 1 applies to an arbitrary tree decomposition of space, i.e., some regions of space may be more refined than others in an adaptive fashion. To compute meaningful complexity bounds, however, it is necessary to impose some structure on the tree. As is standard, we assume a tree with $L = O(\log N)$ levels in d dimensions is given such that each leaf box contains at most a constant number of DOFs independent of N. Letting k_ℓ denote the maximum of $|\mathcal{S}_i|$ over all DOF sets corresponding to boxes on level ℓ and assuming $k_\ell \leq k_{\ell+1}$ for all ℓ, we obtain the following complexity result.

THEOREM 3.3. *Under the above assumptions and assuming further that a constant number of points is used to discretize the proxy surface* Γ*, we have that the cost* t_f *of constructing the RS-S factorization* F *according to Algorithm* 1 *and the cost* t_s *of applying* F *or* F^{-1} *are given, respectively, as*

$$t_f = O(N) + \sum_{\ell=1}^{L-2} O(2^{d(L-\ell)} k_\ell^3), \quad t_s = O(N) + \sum_{\ell=1}^{L-2} O(2^{d(L-\ell)} k_\ell^2).$$

The memory requirement is trivially $m_f = O(t_s)$.

Proof. Let $k_0 = 1$ for convenience. Note that, for a DOF set $\mathcal{B}_i$ corresponding to a box at level ℓ, we have $|\mathcal{B}_i| = O(k_{\ell-1})$, $|\mathcal{N}_i| = O(k_{\ell-1})$, and $|\mathcal{O}_i| = O(k_{\ell-1})$, since for leaf boxes the number of DOFs is bounded by a constant and for nonleaf boxes the DOFs are given by aggregating skeleton DOFs of child boxes at the previous level.

Because the proxy surface Γ is discretized with a constant number of points, the first matrix in (12) used to compute an ID for the skeletonization with respect to $\mathcal{B}_i$ is of size $O(|\mathcal{O}_i|) \times O(|\mathcal{B}_i|)$. This implies that the cost of skeletonizing with respect to the DOFs $\mathcal{B}_i$ corresponding to a box at level ℓ is $O(k_\ell^3)$ using the complexity result in subsection 2.1.

Finally, at each level ℓ, there are at most $2^{d(L-\ell)}$ boxes, which gives the stated complexity for t_f using the fact that $2^{dL} = O(N)$. The complexity for t_s follows a similar argument, noting that all the block unit-triangular matrices can be trivially inverted. ∎

For kernel functions $K(z)$ such as we consider here, i.e., relatively nonoscillatory Green's functions arising from elliptic PDEs, standard multipole-type estimates [14, 15] can typically be used to show far-field interaction blocks $\mathsf{K}_{\mathcal{B}_i\mathcal{F}_i}$ have ranks depending only weakly on N. As previously mentioned in subsection 3.3.1, however, Algorithm 1 involves compressing also far-field interaction blocks that have received Schur complement updates to some of their entries from earlier steps of skeletonization. For such entries, multipole estimates no longer directly apply, but ample numerical experimentation seems to indicate similar rank behavior (see section 4) and thus it is common to assume that updated blocks of this nature still exhibit multipole-like rank behavior [22, 8, 9]. Proceeding under this assumption, we obtain a more explicit complexity estimate.

COROLLARY 3.4. *Suppose that for any fixed tolerance ϵ we have $k_\ell = O(\ell^q)$ in Theorem 3.3 for some $q > 0$, i.e., the skeleton sets grow only as some power of the level index ℓ and $k_{L-2} = O(\log^q N)$. Then the RS-S factorization cost t_f, apply/solve cost t_s, and memory requirement m_f scale as*

$$t_f = O(N), \quad t_s = O(N), \quad m_f = O(N),$$

with constants depending on the tolerance ϵ and dimension d.

Note that the construction of the initial tree decomposition of space requires an additional upfront cost of $O(N \log N)$, but in practice this cost is negligible compared to the factorization itself.

3.4. Extension: Hybrid skeletonization. Algorithmically, the RS-S factorization has much in common with the RS factorization [25, 22]. The key distinction between the two is exactly what is meant by "skeletonization." In subsection 3.1, the strong skeletonization process we describe is used to compress far-field interactions (e.g., $\mathsf{K}_{\mathcal{BF}}$), leading to ranks essentially independent of N under our assumptions. In contrast, in the traditional skeletonization procedure both the far-field *and* the near-field are compressed, i.e., blocks such as $\mathsf{K}_{\mathcal{B}\mathcal{B}^\mathsf{c}}$ with $\mathcal{B}^\mathsf{c} = \mathcal{N} \cup \mathcal{F}$. As a consequence, the skeleton set grows with the rank of the near-field interactions, which typically goes as $O(N^{\frac{d-1}{d}})$ at the top levels as has been illustrated in previous work [21]. After sparse elimination analogous to the strong case, we are left with

$$\widetilde{\mathsf{U}}_\mathsf{T}\widetilde{\mathsf{P}}^*\mathsf{A}\widetilde{\mathsf{P}}\widetilde{\mathsf{L}}_\mathsf{T} = \left[\begin{array}{cc|c} \mathsf{X}_{\mathcal{RR}} & & \\ & \mathsf{X}_{\mathcal{SS}} & \mathsf{A}_{\mathcal{S}\mathcal{B}^\mathsf{c}} \\ \hline & \mathsf{A}_{\mathcal{B}^\mathsf{c}\mathcal{S}} & \mathsf{A}_{\mathcal{B}^\mathsf{c}\mathcal{B}^\mathsf{c}} \end{array}\right] \equiv \widetilde{\mathsf{Z}}(\mathsf{A};\mathcal{B}).$$

Defining the notation

$$\widetilde{\mathsf{V}} \equiv \widetilde{\mathsf{P}}\widetilde{\mathsf{U}}_\mathsf{T}^{-1}, \quad \widetilde{\mathsf{W}} \equiv \widetilde{\mathsf{L}}_\mathsf{T}^{-1}\widetilde{\mathsf{P}}^* \tag{18}$$

analogously to (8), we obtain $\widetilde{\mathsf{Z}}(\mathsf{A};\mathcal{B}) \approx \widetilde{\mathsf{V}}^{-1}\mathsf{A}\widetilde{\mathsf{W}}^{-1}$. We refer to this near-field compression and subsequent decoupling as *weak* skeletonization to distinguish it from its strong counterpart.

While strong skeletonization typically leads to asymptotically more efficient factorizations than weak skeletonization due to the higher rank of near-field interactions compared to far-field interactions, it suffers from a higher storage cost. This is because, in contrast to the weak case, strong skeletonization requires an additional step to explicitly decouple redundant DOFs from their near-field, and the corresponding block elimination operators must be stored. To decrease the constant factor in the

Algorithm 2. The hybrid recursive skeletonization factorization.

```
1: // Initialize
2: $\mathsf{A} := \mathsf{K}$
3: for $\ell := 1$ to $L-2$ do
4:   for each box $i \in \mathscr{L}_\ell$ do
5:     // Identify relevant DOFs for weak skeletonization
6:     $\left[\widetilde{\mathcal{B}}_i, \widetilde{\mathcal{B}}_i^{\mathsf{c}}\right] := \{$active DOFs in box/complement$\}$
7:     // Perform weak skeletonization with respect to DOFs
8:     $\mathsf{A} := \widetilde{\mathsf{Z}}\left(\mathsf{A}; \widetilde{\mathcal{B}}_i\right) \approx \widetilde{\mathsf{V}}_i^{-1}\mathsf{A}\widetilde{\mathsf{W}}_i^{-1}$
9:   end for
10:  for each box $i \in \mathscr{L}_\ell$ do
11:    // Identify relevant DOFs for strong skeletonization
12:    $[\mathcal{B}_i, \mathcal{N}_i, \mathcal{F}_i] := \{$active DOFs in box/near-field/far-field$\}$
13:    // Perform strong skeletonization with respect to DOFs
14:    $\mathsf{A} := \mathsf{Z}(\mathsf{A}; \mathcal{B}_i) \approx \mathsf{V}_i^{-1}\mathsf{A}\mathsf{W}_i^{-1}$
15:  end for
16: end for
17: // Store middle block diagonal matrix and permutation
18: $\mathsf{D} := \mathsf{P}_t^*\mathsf{A}\mathsf{P}_t$
19: Output: $\mathsf{F}$ as in (19)
```

asymptotic storage cost of strong skeletonization, we can combine both weak and strong skeletonization in alternating fashion to obtain the RS-WS in Algorithm 2.

Using exactly the same tree decomposition as before, we begin by looping over each box at the bottom level $\ell = 1$ and performing weak skeletonization with respect to the corresponding DOF sets $\widetilde{\mathcal{B}}_i$ for $i \in \mathscr{L}_1$, where we use a tilde to explicitly mark that we are performing *weak* skeletonization as in (18). Assuming $|\mathscr{L}_1| = r$, this yields

$$\widetilde{\mathsf{Z}}\left(\mathsf{K}; \widetilde{\mathcal{B}}_1, \widetilde{\mathcal{B}}_2, \dots, \widetilde{\mathcal{B}}_r\right) \approx \left(\prod_{i \in \mathscr{L}_1'} \widetilde{\mathsf{V}}_i^{-1}\right) \mathsf{K} \left(\prod_{i \in \mathscr{L}_1} \widetilde{\mathsf{W}}_i^{-1}\right).$$

Having now decoupled some number of DOFs via weak skeletonization without modifying any nonzero off-diagonal blocks, it is now possible to loop *again* over $\mathscr{L}_1$, this time performing *strong* skeletonization with respect to each set of active DOFs $\mathcal{B}_i$ on the level. With $\mathsf{A} = \widetilde{\mathsf{Z}}(\mathsf{K}\widetilde{\mathcal{B}}_1, \widetilde{\mathcal{B}}_2, \dots, \widetilde{\mathcal{B}}_r)$, this gives

$$\mathsf{Z}(\mathsf{A}; \mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_r) \approx \left(\prod_{i \in \mathscr{L}_1'} \mathsf{V}_i^{-1}\right)\left(\prod_{i \in \mathscr{L}_1'} \widetilde{\mathsf{V}}_i^{-1}\right) \mathsf{K} \left(\prod_{i \in \mathscr{L}_1} \widetilde{\mathsf{W}}_i^{-1}\right)\left(\prod_{i \in \mathscr{L}_1} \mathsf{W}_i^{-1}\right).$$

We repeat this process of weak skeletonization followed by strong skeletonization at each level. A final step of permutation leads to the RS-WS factorization $\mathsf{F} \approx \mathsf{K}$ with

(19)
$$\mathsf{F} \equiv \left[\prod_{\ell \in [L-2]} \left(\prod_{i \in \mathscr{L}_\ell} \widetilde{\mathsf{V}}_i\right)\left(\prod_{i \in \mathscr{L}_\ell} \mathsf{V}_i\right)\right] \mathsf{P}_t\mathsf{D}\mathsf{P}_t^* \left[\prod_{\ell \in [L-2]'} \left(\prod_{i \in \mathscr{L}_\ell'} \mathsf{W}_i\right)\left(\prod_{i \in \mathscr{L}_\ell'} \widetilde{\mathsf{W}}_i\right)\right],$$

which is analogous to (16) but more cumbersome notationally due to the need to loop over the boxes on each level twice. We remark that, as a modification to the above, it is possible to perform a final step of weak skeletonization at level $\ell = L-1$ even though subsequent strong skeletonization is not possible, and this is what we do in practice.

4. Numerical results.

To evaluate the performance of the RS-S and RS-WS factorizations, we implemented a number of examples in MATLAB on top of the FLAM library (https://github.com/klho/FLAM/). Our research code is available at https://github.com/victorminden/strong-skel/. We use adaptive quadtrees and octrees as appropriate, refining until the number of DOFs per leaf box is bounded by $n_{\text{occ}} = O(1)$. Diagonal blocks of D were factored using the Cholesky decomposition for positive definite K and the LU decomposition otherwise.

The primary quantities of interest for our examples (where applicable) are given in the following legend:

- ϵ: tolerance parameter for all IDs;
- N: total number of DOFs;
- t_f: wall clock time to construct the factorization F, in seconds;
- t_s: wall clock time to solve $\mathsf{F}x = b$ for x, in seconds;
- m_f: memory required to store F, in GB;
- e_a: estimate of $\|\mathsf{K}-\mathsf{F}\|/\|\mathsf{K}\|$;
- e_s: estimate of $\|\mathsf{I}-\mathsf{K}\mathsf{F}^{-1}\| \geq \|\mathsf{K}^{-1}-\mathsf{F}^{-1}\|/\|\mathsf{K}^{-1}\|$;
- n_i: number of iterations to solve (2) using CG to a tolerance of 10^{-12} on the relative residual norm, where the right-hand side b is given up to scaling by $b = \mathsf{K}x$ for x a vector with normally distributed entries.

We estimate the operator errors using the power method [10, 23] to a tolerance of 10^{-2} in the relative error. For this and for CG, the matrices K and K^* were applied via the fast Fourier transform or a kernel-independent FMM as appropriate.

All computations were performed in MATLAB R2015b on a 64-bit Linux server with Intel Xeon E7-8890 v3 CPUs at 2.50 GHz using up to 72 cores through BLAS multithreading, where the number of cores at any time was chosen adaptively by MATLAB.

4.1. Example 1: Unit square in two dimensions.

We begin with a simple 2D example on the unit square $\Omega = [0,1]^2$. Taking $a(x) \equiv 0$, $b(x) \equiv c(y) \equiv 1$, and $K(z) \equiv -\frac{1}{2\pi}\log(\|z\|)$ in (1), we discretize the resulting first-kind volume integral equation using piecewise-constant collocation on a uniform $\sqrt{N}$ by $\sqrt{N}$ grid such that $\mathsf{K}_{ij} = \frac{1}{N}K(x_i - x_j)$ for $i \neq j$. The diagonal entries K_{ii} are approximated adaptively using the `dblquad` function in MATLAB for simplicity such that

$$\mathsf{K}_{ii} \approx \int_{-h/2}^{h/2}\int_{-h/2}^{h/2} K(x-y)\,dx\,dy,$$

where $h \equiv 1/\sqrt{N}$. Note that this is essentially a Nyström method, but viewing it as piecewise-constant collocation makes sense of the modified diagonal. The order of accuracy of this quadrature is not high compared to other more accurate quadratures based on the idea of local corrections near the singularity, but its simplicity makes it a good candidate for illustrating our approach.

We compare four different skeletonization-based approaches to factorizing K: RS [22, 25], HIF [22], and the strong and hybrid recursive skeletonization factorizations introduced in section 3 (RS-S and RS-WS). Since it is based strictly on near-field compression, we expect RS to exhibit fundamentally different asymptotic scaling, whereas the other three methods should all exhibit essentially linear scaling under our

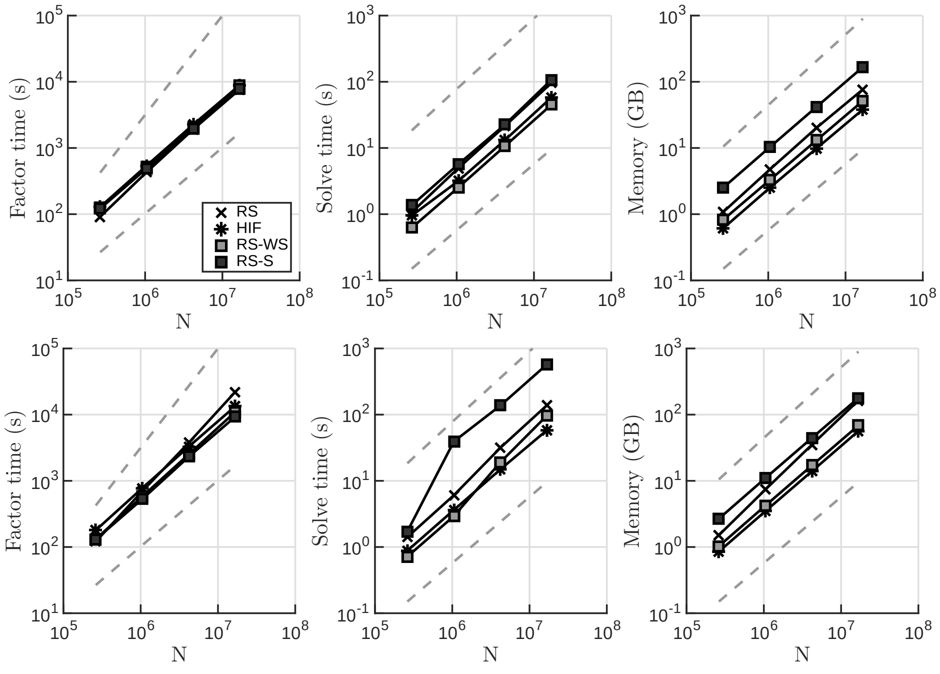


FIG. 8. *Wall clock factor times t_f and solve times t_s and memory usage m_f from Example 1 are shown for $\epsilon = 10^{-6}$ (top row) and $\epsilon = 10^{-9}$ (bottom row). Each plot follows the top-left legend, with additional reference scaling curves $O(N^{3/2})$ and $O(N)$ (left subplots) and $O(N \log N)$ and $O(N)$ (center and right subplots). Corresponding data are given in Table* 1.

rank assumptions. All cases were run across a range of N for tolerances $\epsilon = 10^{-6}$ and $\epsilon = 10^{-9}$, with results visualized in Figure 8 and corresponding data given in Tables 1 and 2. For HIF and RS, we used an occupancy parameter $n_{\rm occ} = 64$, whereas for RS-S and RS-WS, we used $n_{\rm occ} = 256$, hand-tuned in each case for optimal performance. For all methods we used $n_p = 64$ proxy points to discretize the proxy surface.

In this 2D example, we see that all methods remain relatively competitive in terms of factorization time t_f for both tolerances. Looking at the plots of solve time t_s, we see that while all methods seem to scale as $O(N \log N)$ or $O(N)$, for $\epsilon = 10^{-9}$, RS-S is significantly ($\sim$ 4X to 10X) slower than the other methods. Since t_s decreases drastically when using RS-WS (where we add additional levels of weak skeletonization) we hypothesize that the jump in solve time for RS-S is due to caching and the increased size of the subblocks comprising factors of F for RS-S as compared to the other methods. This belief is reinforced by the scaling plots for the memory m_f, in which we see that, in two dimensions, memory usage for RS-S tends to be the highest, followed by RS, RS-WS, and HIF. We note as well the sizable difference in m_f between RS-S and RS-WS, which shows that hybrid skeletonization is effective at reducing memory usage in this setting.

Looking at Table 2, we see that the forward error e_a of the approximate operator is roughly $O(\epsilon)$. This seems to indicate that the relative operator-norm error of the factorization is well-controlled by the relative error in the IDs of off-diagonal blocks, which is difficult to prove for factorizations based on skeletonization. The bound e_s on the error for the inverse operator exhibits similar behavior, though we lose accuracy

TABLE 1
Timing and memory results for Example 1.

		RS			HIF			RS-S			RS-WS		
ϵ	N	t_f	t_s	m_f	t_f	t_s	m_f	t_f	t_s	m_f	t_f	t_s	m_f
10^{-6}	512^2	9.1e+1	1.1e+0	1.1e+0	1.3e+2	9.5e−1	6.1e−1	1.2e+2	1.4e+0	2.5e+0	1.3e+2	6.2e−1	8.3e−1
	1024^2	4.3e+2	4.9e+0	4.7e+0	5.4e+2	3.2e+0	2.5e+0	4.9e+2	5.7e+0	1.0e+1	5.1e+2	2.5e+0	3.3e+0
	2048^2	1.9e+3	2.1e+1	2.0e+1	2.3e+3	1.3e+1	9.8e+0	1.9e+3	2.2e+1	4.2e+1	2.1e+3	1.1e+1	1.3e+1
	4096^2	7.7e+3	9.6e+1	7.7e+1	8.9e+3	5.8e+1	3.8e+1	7.8e+3	1.1e+2	1.7e+2	8.8e+3	4.5e+1	5.2e+1
10^{-9}	512^2	1.2e+2	1.4e+0	1.5e+0	1.8e+2	8.8e−1	8.5e−1	1.3e+2	1.7e+0	2.7e+0	1.3e+2	7.2e−1	1.0e+0
	1024^2	7.0e+2	6.1e+0	7.4e+0	7.8e+2	3.6e+0	3.5e+0	5.3e+2	3.9e+1	1.1e+1	5.8e+2	2.9e+0	4.2e+0
	2048^2	3.8e+3	3.2e+1	3.5e+1	3.3e+3	1.5e+1	1.4e+1	2.3e+3	1.4e+2	4.4e+1	2.5e+3	1.9e+1	1.7e+1
	4096^2	2.2e+4	1.4e+2	1.6e+2	1.3e+4	5.8e+1	5.7e+1	9.3e+3	5.7e+2	1.8e+2	1.1e+4	9.7e+1	6.9e+1

TABLE 2
Accuracy results for Example 1.

		RS			HIF			RS-S			RS-WS		
ϵ	N	e_a	e_s	n_i	e_a	e_s	n_i	e_a	e_s	n_i	e_a	e_s	n_i
10^{-6}	512^2	2.5e−07	3.2e−04	3	2.9e−07	5.8e−04	3	4.0e−08	4.0e−04	3	3.0e−07	7.0e−04	3
	1024^2	7.3e−07	3.6e−04	3	3.8e−07	5.0e−04	3	4.2e−08	6.0e−04	3	3.0e−07	1.6e−03	3
	2048^2	1.2e−06	5.2e−04	3	5.1e−07	9.7e−04	3	4.4e−08	7.9e−04	3	3.0e−07	1.9e−03	3
	4096^2	1.3e−06	9.2e−04	3	6.0e−07	1.0e−03	3	5.0e−08	1.6e−03	3	3.1e−07	2.0e−03	3
10^{-9}	512^2	1.7e−10	3.1e−07	2	3.6e−10	5.3e−07	2	2.7e−11	3.3e−07	2	1.5e−10	1.1e−06	2
	1024^2	6.8e−10	4.5e−07	2	1.9e−10	5.6e−07	2	2.7e−11	5.5e−07	2	1.9e−10	1.7e−06	2
	2048^2	7.6e−10	5.6e−07	2	8.5e−10	9.7e−07	2	2.9e−11	7.8e−07	2	3.0e−10	2.5e−06	2
	4096^2	7.9e−10	1.5e−06	2	9.5e−10	9.9e−07	2	3.0e−11	1.5e−06	2	2.2e−10	3.5e−06	2

due to ill-conditioning of K. Finally, while at these accuracy levels we may use F as a moderate-accuracy direct solver, these factorizations perform exceedingly well when used as preconditioners for CG, as exhibited by the small number of iterations n_i required to attain a relative residual norm of 10^{-12}. Note that the unpreconditioned method fails to converge within 100 iterations.

4.2. Example 2: Unit cube in three dimensions. We turn now to the 3D analogue of Example 1, a first-kind volume integral equation on the unit cube $\Omega = [0,1]^3$ with $a(x) \equiv 0$, $b(x) \equiv c(y) \equiv 1$, and $K(z) \equiv \frac{1}{4\pi\|z\|}$ in (1). As before, we use piecewise-constant collocation on a regular grid with adaptive quadrature for the diagonal entries. In the interest of constructing efficient preconditioners or low-precision direct solvers, we consider the tolerance levels $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$ with results visualized in Figure 9 and corresponding data given in Tables 3 and 4. For $\epsilon = 10^{-3}$ we used the occupancy parameter $n_{\mathrm{occ}} = 64$ for all octrees, whereas for $\epsilon = 10^{-6}$ we used $n_{\mathrm{occ}} = 512$ for RS-S and RS-WS and $n_{\mathrm{occ}} = 64$ for RS and HIF. To discretize the proxy surface, we choose $n_p = 512$ points randomly distributed on the sphere.

Contrary to the 2D case, we immediately observe the difference in scaling between RS and the other methods considered for each of t_f, t_s, and m_f. Further, for $\epsilon = 10^{-3}$ we see that HIF, RS-S, and RS-WS all scale approximately like $O(N)$ with a clear trade-off between factorization time and memory usage—RS-S gives the smallest t_f

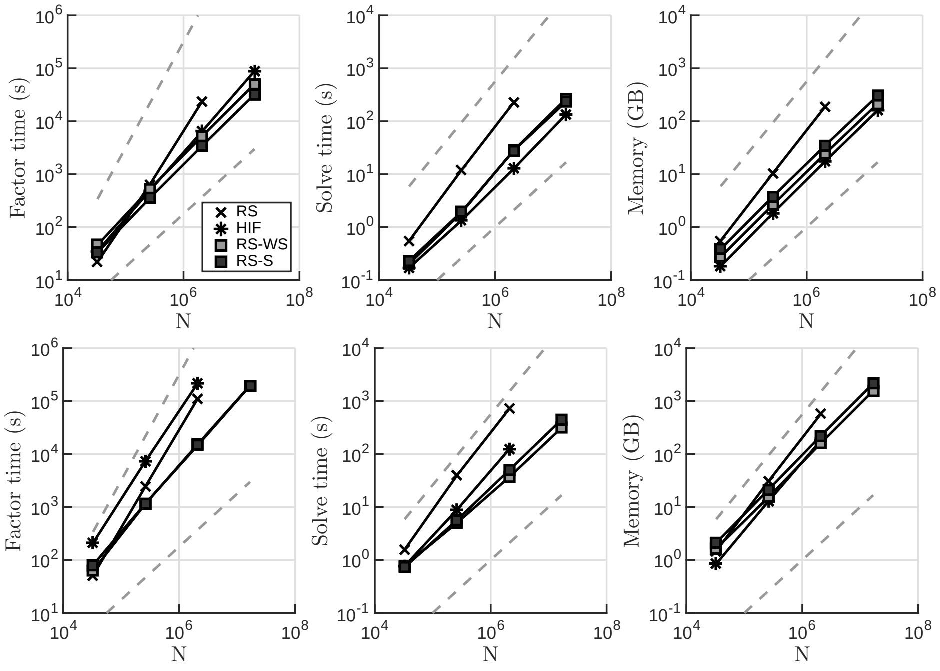


FIG. 9. *Wall clock factor times t_f and solve times t_s and memory usage m_f from Example 2 are shown for $\epsilon = 10^{-3}$ (top row) and $\epsilon = 10^{-6}$ (bottom row). Each plot follows the top-left legend, with additional reference scaling curves $O(N^2)$ and $O(N)$ (left subplots) and $O(N^{4/3})$ and $O(N)$ (center and right subplots). Corresponding data are given in Table 3. Note that in several cases the curves for RS-S and RS-WS lie nearly on top of each other.*

TABLE 3
Timing and memory results for Example 2.

		RS			HIF			RS-S			RS-WS		
ϵ	N	t_f	t_s	m_f	t_f	t_s	m_f	t_f	t_s	m_f	t_f	t_s	m_f
10^{-3}	32^3	2.2e+1	5.5e−1	5.5e−1	3.5e+1	1.7e−1	1.8e−1	3.3e+1	2.3e−1	3.8e−1	4.6e+1	2.0e−1	2.7e−1
	64^3	6.4e+2	1.2e+1	1.0e+1	4.9e+2	1.3e+0	1.8e+0	3.5e+2	1.9e+0	3.8e+0	5.2e+2	1.9e+0	2.6e+0
	128^3	2.4e+4	2.3e+2	1.9e+2	6.4e+3	1.3e+1	1.8e+1	3.4e+3	2.8e+1	3.5e+1	5.2e+3	2.8e+1	2.5e+1
	256^3	–	–	–	8.9e+4	1.3e+2	1.6e+2	3.2e+4	2.4e+2	3.0e+2	5.0e+4	2.7e+2	2.1e+2
10^{-6}	32^3	5.1e+1	1.6e+0	1.5e+0	2.1e+2	7.7e−1	8.5e−1	8.1e+1	7.3e−1	2.1e+0	6.3e+1	7.7e−1	1.6e+0
	64^3	2.4e+3	4.0e+1	3.1e+1	7.3e+3	8.9e+0	1.3e+1	1.2e+3	5.7e+0	2.1e+1	1.2e+3	5.0e+0	1.5e+1
	128^3	1.1e+5	7.2e+2	5.8e+2	2.2e+5	1.2e+2	1.8e+2	1.5e+4	5.0e+1	2.2e+2	1.5e+4	3.7e+1	1.6e+2
	256^3	–	–	–	–	–	–	2.0e+5	4.5e+2	2.2e+3	2.0e+5	3.2e+2	1.6e+3

TABLE 4
Accuracy results for Example 2.

		RS			HIF			RS-S			RS-WS		
ϵ	N	e_a	e_s	n_i	e_a	e_s	n_i	e_a	e_s	n_i	e_a	e_s	n_i
10^{-3}	32^3	2.3e−04	2.3e−02	6	3.1e−04	2.3e−02	6	9.1e−05	1.2e−02	5	1.9e−04	8.8e−02	7
	64^3	3.3e−04	4.5e−02	7	3.6e−04	5.4e−02	6	1.1e−04	2.5e−02	6	2.3e−04	1.0e−01	8
	128^3	1.1e−03	1.2e−01	8	1.2e−03	8.3e−02	8	1.2e−04	4.3e−02	6	3.0e−04	9.7e−02	9
	256^3	–	–	–	3.2e−03	2.0e−01	11	1.3e−04	9.0e−02	7	7.9e−04	3.4e−01	11
10^{-6}	32^3	1.8e−07	4.3e−05	3	1.2e−07	2.8e−05	2	2.0e−08	1.8e−05	2	2.2e−07	1.6e−04	3
	64^3	3.2e−07	7.1e−05	3	2.4e−07	9.2e−05	3	3.1e−08	2.7e−05	2	2.7e−07	1.9e−04	3
	128^3	6.2e−07	1.7e−04	3	3.5e−07	1.7e−04	3	4.0e−08	4.0e−05	3	4.7e−07	3.9e−04	3
	256^3	–	–	–	–	–	–	4.5e−08	8.2e−05	3	8.0e−07	5.9e−04	3

but the largest m_f of the three, the opposite is true for HIF, and RS-WS is somewhere between RS and HIF. In particular, note that the memory usage for RS-WS is less than for RS-S, as desired.

For $\epsilon = 10^{-6}$, we see that t_f for RS-S and RS-WS is markedly less than for HIF. We believe that some of this overhead is due to auxiliary data structures and functions associated with the more complicated geometry exploited in HIF and could therefore potentially be reduced via more sophisticated implementations. For all three, however, we see scaling of t_f that appears better than $O(N^2)$ but not quite $O(N)$ as would be predicted under our assumptions. One possibility is *boundary effects*: essentially, we are perhaps not yet in the asymptotic regime because

(a) boxes at the boundary of Ω have smaller sets of near- and far-field DOFs than do interior boxes, and are therefore cheaper to skeletonize, and,

(b) the number of interior boxes is relatively small in three dimensions for small N.

Another possibility is simply that our assumption on rank behavior is incorrect at $\epsilon = 10^{-6}$ for this example. Unfortunately, distinguishing between these cases requires larger tests than are currently feasible due to memory constraints.

As in the 2D case, Table 4 shows that the approximate relative operator-norm error e_a seems relatively well-controlled by ϵ, though we observe small growth in N. Similarly, the bound e_s on the relative error for the inverse operator again loses a few digits compared to e_a due to conditioning. For $\epsilon = 10^{-3}$ the number of CG iterations to convergence grows with N, albeit slowly. For $\epsilon = 10^{-6}$, however, n_i remains stable.

4.3. Example 3: Unit sphere in three dimensions. As a final example, we move to a more complicated 3D geometry. Letting $G(z) = \frac{1}{4\pi\|z\|}$, we take $a(x) \equiv -1/2$, and $b(x) \equiv c(x) \equiv 1$ on the unit sphere $\Omega = S^2$ to obtain the second-kind boundary integral equation

$$-\frac{1}{2}u(x) + \int_{S^2} \frac{\partial G}{\partial n_y}(x-y)u(y)\,dy = f(x), \quad x \in S^2, \tag{20}$$

where our kernel is the normal derivative of the Green's function for the Laplace equation in three dimensions. This corresponds to a double-layer potential solution representation for the interior Dirichlet Laplace problem on the unit sphere, that is, taking

$$w(x) \equiv \int_{S^2} \frac{\partial G}{\partial n_y}(x-y)u(y)\,dy \tag{21}$$

we have $\Delta w(x) = 0$ for x inside the unit ball and $w(x) = f(x)$ on S^2.

While it is possible to build a periodic quadtree on a 2D parameterization of S^2, we treat the discretization of the sphere as points in $\mathbb{R}^3$ and use an octree. We use a centroid collocation scheme to discretize (20), wherein we represent S^2 as a collection of flat triangles and treat all near-field interactions using fourth-order tensor-product Gauss–Legendre quadrature, where we define near-field interactions as interactions between triangles separated by a distance less than the average triangle diameter. Note that this leads to an unsymmetric matrix K. We choose the occupancy parameter $n_{\text{occ}} = 256$ and $n_p = 512$ random proxy points as in Example 2.

Timing and memory results for $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$ can be seen in Figure 10 with corresponding data for RS-S and RS-WS in Table 5. For $\epsilon = 10^{-3}$, all quantities behave definitively linearly as expected, with RS-WS again offering a trade-off between runtime and memory usage. For $\epsilon = 10^{-6}$, however, we actually observe *sublinear*

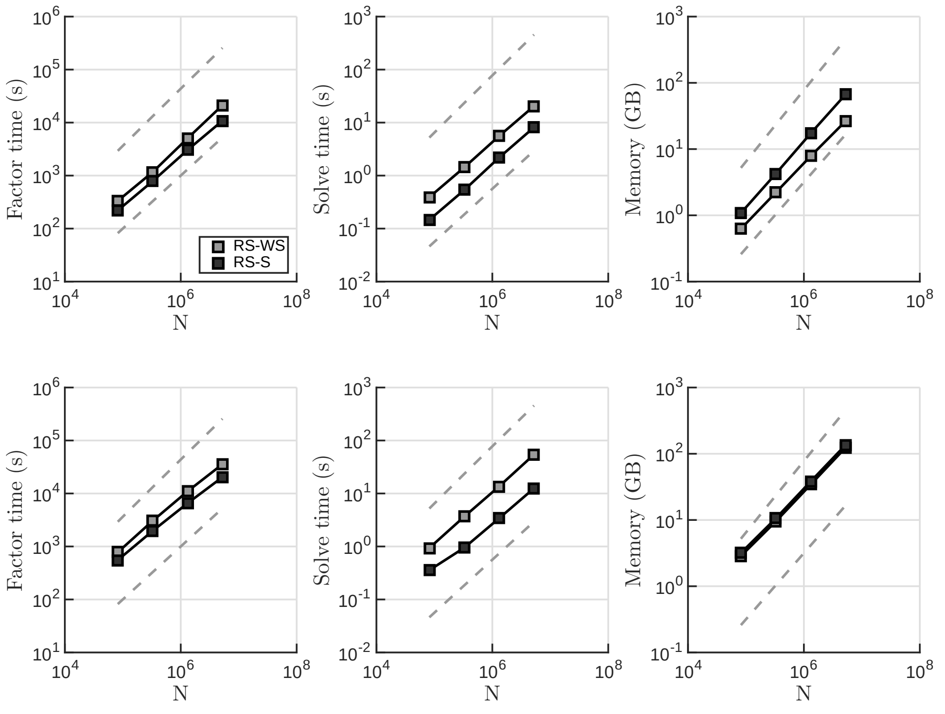


FIG. 10. *Wall clock factor times t_f and solve times t_s and memory usage m_f from Example* 3 *are shown for $\epsilon = 10^{-3}$ (top row) and $\epsilon = 10^{-6}$ (bottom row). Each plot follows the top-left legend, with additional reference scaling curves $O(N \log N)$ and $O(N)$. Corresponding data are given in Table* 5*. Note that in the last subplot the curves for RS-S and RS-WS lie nearly on top of each other.*

TABLE 5
Timing and memory results for Example 3.

		RS-S			RS-WS		
ϵ	N	t_f	t_s	m_f	t_f	t_s	m_f
10^{-3}	81920	2.2e+2	1.5e−1	1.1e+0	3.3e+2	3.9e−1	6.3e−1
	327680	7.8e+2	5.3e−1	4.2e+0	1.2e+3	1.4e+0	2.2e+0
	1310720	3.0e+3	2.2e+0	1.7e+1	5.0e+3	5.6e+0	7.8e+0
	5242880	1.1e+4	8.1e+0	6.7e+1	2.1e+4	2.0e+1	2.6e+1
10^{-6}	81920	5.5e+2	3.6e−1	3.2e+0	7.8e+2	9.4e−1	2.9e+0
	327680	2.0e+3	9.5e−1	1.1e+1	3.0e+3	3.7e+0	9.5e+0
	1310720	6.5e+3	3.4e+0	3.8e+1	1.1e+4	1.4e+1	3.4e+1
	5242880	2.0e+4	1.3e+1	1.4e+2	3.6e+4	5.4e+1	1.2e+2

scaling of t_f with N, which clearly indicates nonasymptotic behavior. Further, looking at m_f we see that memory usage is not significantly lessened with RS-WS for this example due to the fact that near-field compression is only mildly effective at this precision level.

In Table 6 we provide e_a and e_s for this example as well as a new quantity, e_p. The explanation of this is as follows. First, we choose 16 random sources $\{y_j\}$ with $\|y_j\| = 2$ and construct the harmonic field

$$v(x) \equiv \sum_j G(x - y_j) q_j,$$

Table 6
Accuracy results for Example 3.

		RS-S			RS-WS		
ϵ	N	e_a	e_s	e_p	e_a	e_s	e_p
10^{-3}	81920	2.2e−04	2.2e−04	4.0e−04	6.2e−04	6.8e−04	4.8e−04
	327680	4.3e−04	4.3e−04	2.2e−04	9.7e−04	9.8e−04	2.6e−04
	1310720	7.5e−04	7.5e−04	1.6e−04	1.4e−03	1.4e−03	2.3e−04
	5242880	1.1e−03	1.1e−03	1.6e−04	1.9e−03	1.9e−03	2.3e−04
10^{-6}	81920	6.9e−07	6.9e−07	3.8e−04	1.0e−06	1.0e−06	3.7e−04
	327680	1.3e−06	1.3e−06	1.8e−04	1.9e−06	1.9e−06	1.8e−04
	1310720	1.7e−06	1.7e−06	9.0e−05	2.8e−06	2.8e−06	8.9e−05
	5242880	2.2e−06	2.2e−06	4.4e−05	3.9e−06	3.9e−06	4.4e−05

where each q_j is a standard normal random variable. Taking the boundary data $f(x)$ in (20) to be $v(x)$ on S^2, the analytic solution to the interior Laplace boundary value problem is exactly $v(x)$ from uniqueness. Numerically, we may use (21) to reconstruct $\hat{v}(x) \approx v(x)$. Taking 16 random targets $\{z_j\}$ with $\|z_j\| = 1/2$, we compute the relative ℓ^2-norm error e_p between $\{v(z_j)\}$ and $\{\hat{v}(z_j)\}$. These results show that RS-S and RS-WS both solve the integral equation (20) up to discretization error. We do not provide preconditioning results for this example, as the linear operator is well-conditioned even without preconditioning.

5. Conclusions.

By modifying the recursive skeletonization process of Martinsson and Rokhlin [25] to operate directly on strongly admissible structure (i.e., perform only far-field compression), we obtain a new factorization, RS-S, that is useful for solving integral equations in $\mathbb{R}^2$ and $\mathbb{R}^3$ both as a medium-accuracy direct solver and as an excellent preconditioner for iterative methods. As a high-accuracy direct solver, the performance of RS-S in three dimensions is less practical due to memory requirements, though 2D performance remains competitive. We further offer a modification of our approach, RS-WS, which gives a trade-off between runtime complexity and storage complexity through additional levels of compression.

We apply both factorizations to a number of examples to evaluate performance according to a number of metrics. While we focus in this paper on solving integral equations, the linear algebraic machinery developed can be applied much more broadly for general structured matrices (e.g., kernelized covariance matrices [28]).

Compared to other skeletonization-based methods for obtaining linear or nearly linear complexity factorizations such as HIF [22] or the method of Corona, Martinsson, and Zorin [8] (for 2D problems), our approach is competitive but more importantly is simpler to implement. In particular, these previous methods based on near-field compression have obtained better runtime complexity at the cost of increased algorithmic complexity by introducing additional geometric information beyond the tree decomposition of space. By working directly with strong admissibility, this becomes unnecessary. For 3D problems, we also observe better runtime performance than HIF; see Figure 9.

In contrast to the IFMM of Coulier, Pouransari, and Darve [9] and Ambikasaran and Darve [2], which provides a fast method for solving integral equations based on exploiting strong admissibility through a telescoping additive decomposition, our method takes the form of a multiplicative factorization. This gives greater flexibility in that we may compute approximate generalized square-roots or log-determinants—essentially, we get the benefit of having a true (albeit approximate) triangular

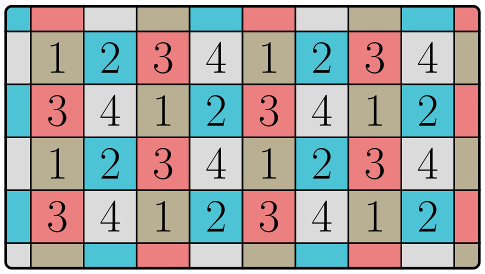


FIG. 11. *To parallelize RS-S and RS-WS efficiently, a four-coloring of the domain (shown) can be used such that no box is colored the same as its neighbors. All boxes of a given color may be skeletonized independently. The* 3*D case is similar, albeit requiring more colors.*

factorization. Further, building on the skeletonization framework allows accelerated compression throughout our algorithm due to the use of a proxy surface as described in subsection 3.2.

The most pressing direction for future research is understanding the rank behavior of far-field blocks that have been subject to Schur complement updates through the skeletonization process as discussed in subsection 3.3.1. The efficiency of our factorizations hinges on these blocks remaining low-rank as the algorithm progresses, which seems to be more or less true in our numerical experiments. Another direction of future research is understanding the additional approximation error introduced through the use of a proxy surface in subsection 3.2. In particular, while it follows from the discussion of subsection 3.2 that an exact ID using proxy points leads to an exact compression of the far-field interactions, it is not immediately evident how tight a bound on the relative error in an approximate ID might be attainable when the IDs are no longer exact and when the discrete approximation to Green's identity is used.

Finally, due to the simple tree structure, RS-S and RS-WS are both easily parallelizable. For example, on a regular 2D grid we may use a four-coloring of the boxes on each level as in Figure 11. In this case, we may perform strong skeletonization with respect to the DOF sets of each brown leaf box in parallel, then similarly for each blue leaf box, and so on. This ordering should also, in principle, allow for adaptation of fast factorization-updating algorithms such as we described in earlier work [29].

Acknowledgments. The authors thank the referees for valuable feedback, particularly on strengthening the statement of Corollary 3.4. The first author V.M. would also like to thank A. Benson, R. Estrin, Y. Li, B. Nelson, N. Skochdopole, and X. Suo for useful comments on early drafts of this manuscript, as well as Stanford University and the Stanford Research Computing Center for providing computational resources and support that have contributed to these research results.

REFERENCES

[1] S. AMBIKASARAN AND E. DARVE, *An $O(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices*, SIAM J. Sci. Comput., 57 (2013), pp. 477–501, https://doi.org/10.1007/s10915-013-9714-z.

[2] S. AMBIKASARAN AND E. DARVE, *The Inverse Fast Multipole Method*, arXiv.org:1407.1572, 2014.

[3] J. BREMER, *A fast direct solver for the integral equations of scattering theory on planar curves with corners*, J. Comput. Phys., 231 (2012), pp. 1879–1899, https://doi.org/10.1016/j.jcp.2011.11.015.

[4] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, *A fast solver for HSS representations via sparse matrices*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 67–81, https://doi.org/10.1137/050639028.

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

[6] Y. Chen, *A fast, direct algorithm for the Lippmann-Schwinger integral equation in two dimensions*, Adv. Comput. Math., 16 (2002), pp. 175–190, https://doi.org/10.1023/A:1014450116300.

[7] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, *On the compression of low rank matrices*, SIAM J. Sci. Comput., 26 (2005), pp. 1389–1404, https://doi.org/10.1137/030602678.

[8] E. Corona, P.-G. Martinsson, and D. Zorin, *An $O(N)$ direct solver for integral equations on the plane*, Appl. Comput. Harmon. Anal., 38 (2015), pp. 284–317, https://doi.org/10.1016/j.acha.2014.04.002.

[9] P. Coulier, H. Pouransari, and E. Darve, *The Inverse Fast Multipole Method: Using a Fast Approximate Direct Solver as a Preconditioner for Dense Linear Systems*, arXiv:1508.01835, 2015.

[10] J. D. Dixon, *Estimating extremal eigenvalues and condition numbers of matrices*, SIAM J. Numer. Anal., 20 (1983), pp. 812–814, https://doi.org/10.1137/0720053.

[11] W. Fong and E. Darve, *The black-box fast multipole method*, J. Comput. Phys., 228 (2009), pp. 8712–8725, https://doi.org/10.1016/j.jcp.2009.08.031.

[12] A. Gillman, P. Young, and P.-G. Martinsson, *A direct solver with $O(N)$ complexity for integral equations on one-dimensional domains*, Front. Math. China, 7 (2012), pp. 217–247, https://doi.org/10.1007/s11464-012-0188-3.

[13] L. Greengard, D. Gueyffier, P.-G. Martinsson, and V. Rokhlin, *Fast direct solvers for integral equations in complex three-dimensional domains*, Acta Numer., 18 (2009), pp. 243–275, https://doi.org/10.1017/S0962492906410011.

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

[15] L. Greengard and V. Rokhlin, *A new version of the fast multipole method for the Laplace equation in three dimensions*, Acta Numer., 6 (1997), pp. 229–269, https://doi.org/10.1017/S0962492900002725.

[16] M. Gu and S. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869, https://doi.org/10.1137/0917055.

[17] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* I: *Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108, https://doi.org/10.1007/s006070050015.

[18] W. Hackbusch and S. Börm, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), pp. 1–35, https://doi.org/10.1007/s00607-002-1450-4.

[19] W. Hackbusch and B. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic. Part* II: *Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47, http://dl.acm.org/citation.cfm?id=333825.333827.

[20] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. National Bureau of Standards, 49 (1952), pp. 409–436.

[21] K. Ho and L. Greengard, *A fast direct solver for structured linear systems by recursive skeletonization*, SIAM J. Sci. Comput., 34 (2012), pp. A2507–A2532, https://doi.org/10.1137/120866683.

[22] K. Ho and L. Ying, *Hierarchical interpolative factorization for elliptic operators: Integral equations*, Commun. Pure Appl. Math., (2015), https://doi.org/10.1002/cpa.21577.

[23] J. Kuczyski and H. Woniakowski, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1094–1122, http://dx.doi.org/10.1137/0613066.

[24] P.-G. Martinsson, *A fast direct solver for a class of elliptic partial differential equations*, J. Sci. Comput., 38 (2009), pp. 316–330, https://doi.org/10.1007/s10915-008-9240-6.

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

[26] P.-G. Martinsson and V. Rokhlin, *An accelerated kernel-independent fast multipole method in one dimension*, SIAM J. Sci. Comput., 29 (2007), pp. 1160–1178, https://doi.org/10.1137/060662253.

[27] W. McLean, *Strongly Elliptic Systems and Boundary Integral Equations*, Cambridge University Press, Cambridge, UK, 2000.

[28] V. Minden, A. Damle, K. L. Ho, and L. Ying, *Fast Spatial Gaussian Process Maximum Likelihood Estimation via Skeletonization Factorizations*, arXiv:1603.08057, 2016.

[29] V. Minden, A. Damle, K. L. Ho, and L. Ying, *A technique for updating hierarchical skeletonization-based factorizations of integral operators*, Multiscale Model. Simul., 14 (2016), pp. 42–64, https://doi.org/10.1137/15M1024500.

[30] X. Pan, J. Wei, Z. Peng, and X. Sheng, *A fast algorithm for multiscale electromagnetic problems using interpolative decomposition and multilevel fast multipole algorithm*, Radio Sci., 47 (2012), https://doi.org/10.1029/2011RS004891.

[31] D. A. Sushnikova and I. V. Oseledets, *"Compress and Eliminate" Solver for Symmetric Positive Definite Sparse Matrices*, arXiv:1603.09133, 2016.

[32] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1382–1411, https://doi.org/10.1137/09074543X.

[33] L. Ying, G. Biros, and D. Zorin, *A kernel-independent adaptive fast multipole algorithm in two and three dimensions*, J. Comput. Phys., 196 (2004), pp. 591–626, https://doi.org/10.1016/j.jcp.2003.11.021.