LOW-RANK KERNEL MATRIX APPROXIMATION
USING SKELETONIZED INTERPOLATION
WITH ENDO- OR EXO-VERTICES

ZIXI XU*, LÉOPOLD CAMBIER†, FRANCOIS-HENRY ROUET‡, PIERRE
L’EPLATTENIER†, YUN HUANG‡, CLEVE ASHCRAFT‡, AND ERIC DARVE*

Abstract. The efficient compression of kernel matrices, for instance the off-diagonal blocks of
discretized integral equations, is a crucial step in many algorithms. In this paper, we study the
application of Skeletonized Interpolation to construct such factorizations. In particular, we study
four different strategies for selecting the initial candidate pivots of the algorithm: Chebyshev grids,
points on a sphere, maximally-dispersed and random vertices. Among them, the first two introduce
new interpolation points (exo-vertices) while the last two are subsets of the given clusters (endo-
vertices). We perform experiments using three real-world problems coming from the multiphysics
code LS-DYNA. The pivot selection strategies are compared in term of quality (final rank) and
efficiency (size of the initial grid). These benchmarks demonstrate that overall, maximally-dispersed
vertices provide an accurate and efficient sets of pivots for most applications. It allows to reach
near-optimal ranks while starting with relatively small sets of vertices, compared to other strategies.

Key word. Low-rank, Kernel, Skeletonization, Interpolation, Rank-revealing QR, Maximally
Dispersed Vertices, Chebyshev

1. Introduction. Our goal is to compute low-rank approximations of matrices
that come from certain kernel functions

\( \mathcal{K} : X \times Y \)

where \( X \) and \( Y \) are 2-D or 3-D regions of space. From \( X \) and \( Y \) we choose sets of
points

\[ X = \{ x_1, \ldots, x_m \} \quad \text{and} \quad Y = \{ y_1, \ldots, y_n \} \]

on which to evaluate the kernel. This yields a \( m \times n \) matrix, \( K_{X,Y} \), where \( K_{i,j} = \mathcal{K}(x_i,y_j) \). When \( \mathcal{K} \) is smooth over the domain surrounding \( X \) and \( Y \), \( K_{X,Y} \) typically
has rapidly decaying singular values and can be well approximated by a low-rank
matrix.

Matrices of this sort arise naturally in many applications. A typical electromagnetic
application may have at a boundary integral equation

\( \tilde{\Phi}(x) = \frac{\mu_0}{4\pi} \int_{\Gamma_X} \mathcal{K}(x,y) \tilde{k}(y) \, dy = \frac{\mu_0}{4\pi} \int_{\Gamma_X} \frac{1}{||x-y||^2} \tilde{k}(y) \, dy \quad \text{for} \quad x \in \Gamma_X \)

where \( \Gamma_X \) and \( \Gamma_Y \) are the boundaries of \( X \) and \( Y \). Here \( \tilde{\Phi} \) is the magnetic vector
potential, \( \mu_0 \) is the electrical permeability and \( \tilde{k} \) is the surface current introduced as
an intermediate variable [33].

In acoustics, a boundary integral equation based on Helmholtz equations is com-
plex

\( \frac{1}{\gamma} p(x) = -\int_{\Gamma_Y} \left( i\rho\omega v_n(y)\mathcal{G} + p(y) \frac{\partial \mathcal{G}}{\partial n} \right) \, dy. \)
where $p(x), p(y)$ are acoustic pressures, $v_n(y)$ is the normal velocity, $\rho$ is the fluid density, $\omega$ is the round frequency [46].

In equations (1.2) and (1.3), the kernel matrices are given by

$$\mathcal{K}(x, y) = \frac{1}{r} \quad \text{and} \quad \mathcal{G}(x, y) = \frac{1}{r} e^{-ikr}$$

where $r = \|x - y\|_2$. Other BEM kernel functions of interest include $\ln(r)$ and $1/r^2$.

The discretized forms of Eq. (1.2) and Eq. (1.3) are linear systems

$$K_{N,N} u_N = b_N$$

where the $|N| \times |N|$ matrix $K_{N,N}$ is dense, with solution $u_N$ and right hand side $b_N$. A submatrix $K_{X,Y}$, where $X \subset N$ and $Y \subset N$, will be dense, but if the subsets $X$ and $Y$ are chosen well, $K_{X,Y}$ will have small numerical rank.

In this case, we have $K_{X,Y} \approx U_{X,\alpha} V_{Y,\alpha}^\top$ where typically $|X| = |Y| = n$ and $|\alpha| = r$, this takes $n^2$ kernel evaluations to form $K_{X,Y}$ and $O(n^2r)$ operations for the RRQR. Our goal is to reduce these two costs, to $O(nr)$ kernel evaluations and $O(nr^2)$ or even $O(nr)$ linear algebra operations.

One approach is to form the dense $K_{X,Y}$ and then compute a rank-revealing QR factorization. If $|X| = |Y| = n$ and $|\alpha| = r$, this takes $n^2$ kernel evaluations to form $K_{X,Y}$ and $O(n^2r)$ operations for the RRQR. Our goal is to reduce these two costs, to $O(nr)$ kernel evaluations and $O(nr^2)$ or even $O(nr)$ linear algebra operations.

A new low-rank approximation approach for the kernel matrix is proposed in this paper. For illustration purpose, this paper takes the kernel functions $1/r$ and $1/r^2$ ($r = \|x - y\|_2$) as examples to show how the approach works. Note that although our approach only focuses on each single sub-matrix extracted from the fully dense matrix, it can be integrated as the building block of a more complex multi-level method using hierarchical matrices for instance.

For convenience, Table 1.1 summarizes the notations we use throughout the paper. Table 1.2 summarizes the terminology (acronyms and algorithms).

**Table 1.1: Notation**

| Symbol | Description |
|--------|-------------|
| $\mathcal{K}$ | The kernel function |
| $X, Y$ | Two regions in space |
| $X^\circ, Y^\circ$ | Sets of discretization points in subdomains of a geometry |
| $\hat{X}, \hat{Y}$ | Subsets of $X$ and $Y$ chosen by Skeletonized Interpolation used to build the low-rank approximation |
| $K_{X,Y}$ | The kernel matrix defined as $(K_{X,Y})_{ij} = \mathcal{K}(x_i, y_j)$ |
| $W_{X^\circ, Y^\circ}$, $W_{\hat{X}, \hat{Y}}$ | Corresponding weight matrices for $X^\circ$ and $Y^\circ$ |
| $r_0$ | Size of $X^\circ$ or $Y^\circ$ |
| $r_1$ | Size of $\hat{X}$ or $\hat{Y}$ |
| $dr(i,j)$ | The distance ratio between two pairs $i$ and $j$ of clusters |
| $\varepsilon$ | The tolerance used in the RRQR algorithms |
| $\varepsilon^*$ | Relative Frobenius-norm error used in the experiments |

**1.1. Previous work.** The efficient solution of boundary integral equations [e.g., Eqns. (1.2) and (1.3)] has been extensively studied. Multiple algorithms have been
### Table 1.2: Terminology

| Abbreviation | Description |
|--------------|-------------|
| SI           | Skeletonized Interpolation |
| MDV          | Maximally-dispersed vertices |
| RRQR         | Rank-revealing QR |
| SI-Chebyshev | SI algorithm using Chebyshev nodes for $X^o$ and $Y^o$ |
| SI-MDV       | SI algorithm using maximally-dispersed vertices for $X^o$, $Y^o$ |
| SI-sphere    | SI algorithm using points on a bounding sphere for $X^o$, $Y^o$ |
| SI-random    | SI algorithm using random vertices for $X^o$ and $Y^o$ |
| GenInitSet   | An algorithm that returns, for a set of vertices $X$, a set $X^o$ of a given maximum size and the corresponding weights $W_{X^o,X}$ |

proposed to efficiently compute low-rank approximations for the off-diagonal blocks $K_{X,Y}$ of the full kernel matrix.

A variety of analytical expansion methods are based on the expansion properties of the kernel. The Fast Multipole Method (FMM) [39, 15, 16] is based on a series expansion of the fundamental solution. It was first proposed by [40] for the Laplacian, and then a diagonal version for the Helmholtz operator [24]. FMM can accelerate matrix-vector products which are coupled with an iterative method. More recently, [18] proposed a kernel-independent FMM based on the interpolation of the kernel.

In [12], Chew et al. presented their benchmark results and scaling studies for the multilevel fast multipole algorithm (MLFMA) and the fast inhomogeneous plane wave algorithm (FIPWA). A hybrid version of MLFMA only requires a fraction of CPU time and memory of the traditional MLFMA. FIPWA largely reduces the CPU time thus making simulations feasible with over one million particles.

In addition to the FMM, there exists other techniques that approximate the kernel function to explicitly compute the low-rank factorization of the kernel submatrices. The Panel Clustering method [27] provides a kernel function approximation using Taylor Series. [48] (and similarly [8] and [47] in Fourier space) takes advantage of the interpolation of $\mathcal{K}(x,y)$ over $X \times Y$ to build a low-rank expansion $K_{X,Y} = S_{X,Y} K_{X,Y} T_{X,Y}^T$ and uses the SVD to recompress that expansion further. Also, Barnes and Hut [2] compute mutual forces in particle systems using a center of mass approximation with a special far-field separation or “admissibility” criterion. This is similar to an order-1 multipole expansion (where one matches multipole terms of order up to 1).

However, these analytical methods are often limited to some specific types of kernel functions, or have complexities larger than $O(nr)$. Adaptive Cross Approximation (ACA) [4, 3] computes individual rows and columns of the matrix. However, it provides few accuracy guarantees and its termination criterion is inaccurate in some cases. Furthermore, the method is not very efficient because it proceeds column by column and row by row instead of using matrix-matrix operations (level 3 BLAS subprograms) that have a more efficient memory access pattern.

Finally, Bebendorf [3] proposes the form

\[
K_{X,Y} = K_{X,Y}^{-1} K_{X,Y}^\dagger \quad (1.5)
\]

where $\tilde{X}$ and $\tilde{Y}$ are interpolation points built iteratively from $\mathcal{K}(x,y)$. Our method explores this interpolation idea; however, we choose the interpolation nodes in a very different way. While [3] builds the nodes one by one in an adaptive fashion, we start
from a predefined grid of nodes and then sample this grid using a RRQR factorization. This leads to a fast and very easy algorithm.

1.2. Skeletonized Interpolation. We now introduce our approach. Consider the kernel matrix \( K_{X,Y} \) having an exact rank \( r \). Let \( \tilde{X} \subseteq X \) and \( \tilde{Y} \subseteq Y \) be sampling points such that \( |\tilde{X}| = |\tilde{Y}| = r \) and \( K_{\tilde{X},\tilde{Y}} \) is nonsingular. Write the kernel matrix in block form and factor as

\[
K_{X,Y} = \begin{bmatrix} K_{\tilde{X},\tilde{Y}} & K_{\tilde{X},Y\setminus\tilde{Y}} \\ K_{X\setminus\tilde{X},\tilde{Y}} & K_{X\setminus\tilde{X},Y\setminus\tilde{Y}} \end{bmatrix} = \begin{bmatrix} I_{\tilde{X},\tilde{Y}} & L_{X\setminus\tilde{X},\tilde{Y}} \end{bmatrix} K_{\tilde{X},\tilde{Y}} \begin{bmatrix} I_{\tilde{Y},\tilde{Y}} & U_{\tilde{Y},Y\setminus\tilde{Y}} \end{bmatrix} + \begin{bmatrix} 0_{\tilde{X},\tilde{Y}} & 0_{X\setminus\tilde{X},Y\setminus\tilde{Y}} \\ 0_{X\setminus\tilde{X},\tilde{Y}} & E_{X\setminus\tilde{X},Y\setminus\tilde{Y}} \end{bmatrix}
\]

(1.6)

where \( L_{X\setminus\tilde{X},\tilde{Y}} = K_{X\setminus\tilde{X},\tilde{Y}} K_{\tilde{X},\tilde{Y}}^{-1} \) and \( U_{\tilde{Y},Y\setminus\tilde{Y}} = K_{\tilde{X},Y\setminus\tilde{Y}} K_{\tilde{X},\tilde{Y}}^{-1} \). If \( K_{X,Y} \) has numerical rank \( r \), as does \( K_{\tilde{X},\tilde{Y}} \), then error matrix \( E_{X\setminus\tilde{X},Y\setminus\tilde{Y}} \) is zero, and we have this low rank factorization of \( K_{X,Y} \).

\[
K_{X,Y} = \begin{bmatrix} I_{\tilde{X},\tilde{X}} & L_{X\setminus\tilde{X},\tilde{X}} \end{bmatrix} K_{\tilde{X},\tilde{Y}} \begin{bmatrix} I_{\tilde{Y},\tilde{Y}} & U_{\tilde{Y},Y\setminus\tilde{Y}} \end{bmatrix} = K_{X\setminus\tilde{X},Y\setminus\tilde{Y}} K_{\tilde{X},\tilde{Y}} K_{X\setminus\tilde{X},\tilde{Y}}^{-1}
\]

(1.7)

We call sampled degrees of freedom \( \tilde{X} \subseteq X \) and \( \tilde{Y} \subseteq Y \) endo-vertices, for they are internal to the domains \( X \) and \( Y \).

In contrast, consider exo-vertices, where \( \tilde{X} \) and \( \tilde{Y} \) are chosen from outside the sets \( X \) and \( Y \), \( K_{\tilde{X},\tilde{Y}} \) is square and nonsingular. Form the large kernel matrix and factor.

\[
K_{X\cup\tilde{X},Y\cup\tilde{Y}} = \begin{bmatrix} K_{\tilde{X},\tilde{Y}} & K_{\tilde{X},Y} \\ K_{X\setminus\tilde{X},\tilde{Y}} & K_{X\setminus\tilde{X},Y\setminus\tilde{Y}} \end{bmatrix} = \begin{bmatrix} I_{\tilde{X},\tilde{X}} & L_{X\setminus\tilde{X},\tilde{Y}} \end{bmatrix} K_{\tilde{X},\tilde{Y}} \begin{bmatrix} I_{\tilde{Y},\tilde{Y}} & U_{\tilde{Y},Y\setminus\tilde{Y}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 \end{bmatrix}
\]

(1.8)

If the error matrix \( E_{X,Y} \) is small in norm, then we have this low rank representation for \( K_{X,Y} \).

\[
K_{X,Y} = L_{X\setminus\tilde{X}} K_{\tilde{X},Y} U_{Y\setminus\tilde{Y}} = K_{\tilde{X},\tilde{Y}} K_{X\setminus\tilde{X},Y\setminus\tilde{Y}}^{-1} K_{X\setminus\tilde{X},\tilde{Y}}
\]

(1.9)

where \( L_{X\setminus\tilde{X}} = K_{X\setminus\tilde{X},\tilde{Y}} K_{\tilde{X},\tilde{Y}}^{-1} \) and \( U_{Y\setminus\tilde{Y}} = K_{\tilde{X},Y\setminus\tilde{Y}} K_{\tilde{X},\tilde{Y}}^{-1} \).

In both equations Eq. (1.7) and Eq. (1.9), the large kernel matrix \( K_{X,Y} \) as well as the smaller \( K_{\tilde{X},\tilde{Y}} \) matrices have log-linear singular values, and so these matrices are ill-conditioned. However, in [11], it was shown that if backward-stable algorithms are used to factor \( K_{\tilde{X},\tilde{Y}} \), the product of the three matrices on the right-hand-sides can be computed accurately.

There is a key difference between choosing endo-vertices and exo-vertices. The choice of endo-vertices \( \tilde{X} \) and \( \tilde{Y} \) means that for endo-, \( K_{\tilde{X},\tilde{Y}} \) must be able to capture the log-linear singular values of \( K_{X,Y} \). For exo-, \( K_{\tilde{X},\tilde{Y}} \) must be able to capture the log-linear singular values of the larger kernel matrix \( K_{\tilde{X},\tilde{Y},X\setminus\tilde{X},Y\setminus\tilde{Y}} \). The numerical rank of \( K_{\tilde{X},\tilde{Y},X\setminus\tilde{X},Y\setminus\tilde{Y}} \) will generally be larger than the numerical rank of \( K_{\tilde{X},\tilde{Y}} \). We will see this to be the case in the experiments to follow.

The problem now becomes how to choose sets \( \tilde{X} \) and \( \tilde{Y} \) in an easy, fast and accurate way. To select \( \tilde{X} \) and \( \tilde{Y} \), we start from larger sets of initial points \( X^0 \subseteq X \) and \( Y^0 \subseteq Y \). We introduce different ways of choosing those points in the next sections; we keep their exact definition unspecified at the moment.
Using the method proposed in [11], we perform two RRQR factorizations over
\[
\tilde{K}_{X^o,Y^o} = W_{X^o,X^o}^{1/2}K_{X^o,Y^o}W_{Y^o,Y^o}^{1/2}
\]
and its transpose in order to choose an optimal set of rows and columns, \(\tilde{X} \subset X^o\) and \(\tilde{Y} \subset Y^o\), for a given tolerance \(\varepsilon\). The diagonal matrices \(W_{X^o,X^o}\) and \(W_{Y^o,Y^o}\) contain integration weights related to the choice of \(X^o\) and \(Y^o\) (more details about this in Section 2.5). Once \(\tilde{X}\) and \(\tilde{Y}\) are selected, the resulting approximation is given by
\[
K_{X,Y} \approx K_{\tilde{X},\tilde{Y}} K_{\tilde{X},\tilde{Y}}^{-1} K_{\tilde{X},Y}
\]
as indicated above.

Let us consider the expense to create this low rank representation. For simplicity, let \(|X| = |Y| = n, |X^o| = |Y^o| = r_0\), and \(|\tilde{X}| = |\tilde{Y}| = r_1\).

- There are \(r_0^2, r_1^2, r_1 n\) and \(r_1 n\) kernel function evaluations to compute \(K_{X^o,Y^o}, K_{\tilde{X},\tilde{Y}}, K_{\tilde{X}\setminus\tilde{X},\tilde{Y}}\) and \(K_{\tilde{X},\tilde{Y}\setminus\tilde{Y}}\), respectively, for a total of \(2r_1 n + r_1^2 + r_1^2\).

- Two RRQR factorizations on \(K_{X^o,Y^o}\) for \(8r_0^2 r_1\) linear algebra operations. To invert \(K_{\tilde{X},\tilde{Y}}\) requires \(\frac{2}{3} r_1^3\) operations, for a total of \(\frac{8}{3} r_0^2 r_1 + \frac{2}{3} r_1^3\) operations.

Ideally, \(X^o\) and \(Y^o\) are as small as possible, and \(\tilde{X}\) and \(\tilde{Y}\) have (almost) the same sizes as the \(\varepsilon\)-rank of \(K_{X,Y}\). The smaller those sets, the less expensive the factorization is to compute, and the better the sets. In addition, as we discuss in Section 2.5, one can always further recompress a given low-rank approximation. However, in this paper, we are interested in the cost of the construction of the initial low-rank basis based on the initial \(X^o\) and \(Y^o\) and reduced \(\tilde{X}\) and \(\tilde{Y}\).

We call this method Skeletonized Interpolation. A high-level description of the Skeletonized Interpolation algorithm is given in Algorithm 1.1; \(\text{GenInitSet}\) denotes a given function (algorithm) used to produce integration weights related to the choice of \(X^o\) and \(Y^o\) (more details about this in Section 2.5). Once \(\tilde{X}\) and \(\tilde{Y}\) are selected, the resulting approximation is given by

\[
K_{X,Y} \approx K_{\tilde{X},\tilde{Y}} K_{\tilde{X},\tilde{Y}}^{-1} K_{\tilde{X},Y}
\]
as indicated above.

Let us consider the expense to create this low rank representation. For simplicity, let \(|X| = |Y| = n, |X^o| = |Y^o| = r_0\), and \(|\tilde{X}| = |\tilde{Y}| = r_1\).

- There are \(r_0^2, r_1^2, r_1 n\) and \(r_1 n\) kernel function evaluations to compute \(K_{X^o,Y^o}, K_{\tilde{X},\tilde{Y}}, K_{\tilde{X}\setminus\tilde{X},\tilde{Y}}\) and \(K_{\tilde{X},\tilde{Y}\setminus\tilde{Y}}\), respectively, for a total of \(2r_1 n + r_1^2 + r_1^2\).

- Two RRQR factorizations on \(K_{X^o,Y^o}\) for \(8r_0^2 r_1\) linear algebra operations. To invert \(K_{\tilde{X},\tilde{Y}}\) requires \(\frac{2}{3} r_1^3\) operations, for a total of \(\frac{8}{3} r_0^2 r_1 + \frac{2}{3} r_1^3\) operations.

In this algorithm, as a rule of thumb, the tolerance \(\varepsilon\) should be slightly smaller than the desired final accuracy. This comes from the fact that the RRQR’s are sub-optimal compared to the SVD (in the sense that they typically lead to slightly larger ranks for a given accuracy).

**Algorithm 1.1** Skeletonized Interpolation: \([\tilde{X}, \tilde{Y}] = SI(\mathcal{X}, X, Y, \text{GenInitSet}, r_0, \varepsilon)\)

**Require:** Kernel \(\mathcal{X}\), discretization points \(X\) and \(Y\), algorithm \(\text{GenInitSet}\), rank \(r_0\), tolerance \(\varepsilon\)

1. Calculate \((X^o, W_{X^o,X^o}) = \text{GenInitSet}(X, r_0)\)
2. Calculate \((Y^o, W_{Y^o,Y^o}) = \text{GenInitSet}(Y, r_0)\)
3. Build temporary matrix \(T_{X^o,Y^o} = W_{X^o,X^o} K_{X^o,Y^o} W_{Y^o,Y^o}\)
4. Perform truncated RRQR \(T_{X^o,Y^o} P_{Y^o,Y^o} = Q_{X^o,\alpha} R_{\alpha,Y^o} + E_{X^o,Y^o}\)
   where \(\|E_{X^o,Y^o}\|_F \leq \varepsilon\)
5. Perform truncated RRQR \(T_{X^o,Y^o}^T P_{X^o,X^o} = Q_{Y^o,\beta} R_{\beta,X^o} + E_{Y^o,X^o}\)
   where \(\|E_{Y^o,X^o}\|_F \leq \varepsilon\)
6. Define \(\hat{Y}\) as the leading \(\min(|\alpha|, |\beta|)\) columns selected by \(P_{Y^o,Y^o}\).
7. Define \(\hat{X}\) as the leading \(\min(|\alpha|, |\beta|)\) columns selected by \(P_{X^o,X^o}\).
8. **return** Interpolation points \(\hat{X}\) and \(\hat{Y}\)

In this paper, we explore four different heuristics to define those sets, in which
may be a subset of \((X, Y)\) (endo-skeleton) or not (exo-skeleton). Those four ways are:

1. Chebyshev grid (exo)
2. Random subset of the vertices (endo)
3. Maximally-Dispersed Vertices or MDV (endo)
4. Points on an enclosing surface, e.g., a sphere or ellipsoid (exo).

1.3. Optimality of the point set. The question of optimality of the rows and columns used for the pivot block \(K_{\hat{X},\hat{Y}}\) has been studied in the past in several papers in a range of fields. See for example \([25, 38, 17, 42, 41, 21, 13, 23, 22, 44]\). We summarize some of the key results and their relevance for the current work. In \([25]\), the authors prove that a strong rank-revealing QR (RRQR) factorization can identify \(r\) columns such that

\[
\sigma_i(A\Pi) \geq \frac{\sigma_i(A)}{q(r, n)}
\]

for matrix \(A\), where \(\Pi\) is a matrix with \(r\) columns that selects the optimal \(r\) columns in \(A\), \(\sigma_i\) is the \(i\)th singular value, and \(q\) is a “low-degree” polynomial. Similarly the error \(E\) from a strong RRQR factorization when we keep only the first \(r\) columns (rank-\(r\) approximation) has an upper bound involving \(\sigma_{r+i}\):

\[
\sigma_i(E) \leq \sigma_{r+i}(A)q(r, n)
\]

From \(A\Pi\), we can similarly calculate a strong RRLQ (the transpose of RRQR) to select \(r\) important rows of \(A\Pi\). This leads to a square sub-block \(A_{SI}\) of \(A\). Note that in this two-step process, the first step with RRQR is the one that determines the accuracy of the low-rank factorization. The second step with RRLQ merely guarantees the stability of the SI procedure (with respect to roundoff errors and small perturbations in the input); specifically, it minimizes the conditioning of the operation with respect to perturbations in the input data. In turn, this guarantees the numerical stability of computing \(K_{\hat{X},\hat{Y}}K_{\hat{X},\hat{Y}}^{-1}\) and \(K_{\hat{X},\hat{Y}}^{-1}K_{\hat{X},\hat{Y}}\), in spite of the fact that the singular values of \(K_{\hat{X},\hat{Y}}\) are rapidly decaying.

From the interlacing singular values theorem \([19]\), the singular values \(\{\sigma_{SI}^1, \ldots, \sigma_{SI}^r\}\) of \(A_{SI}\) keep increasing as \(r\) increases. As a result, we have both lower and upper bounds on \(\sigma_{SI}^r\). This leads to a definition of the best choice of SI block. It should satisfy two equivalent properties:

1. Its singular values \(\{\sigma_{SI}^i\}_{i=1, \ldots, r}\) are close to the corresponding singular values of \(A\) (i.e., with minimal error).
2. The volume of \(A_{SI}\), as defined by the absolute value of its determinant, is maximum.

This motivates some heuristic criteria for choosing exo-points:

1. They should be easy to calculate, such that computing the exo-points is computationally advantageous compared to selecting endo-points from \(K_{\hat{X},\hat{Y}}\).
2. When increasing the size of the matrix, by adding the exo-points \(X^o\) and \(Y^o\), the increase in rank should be minimal. This happens when the set \((X^o, Y^o)\) is “close” to \((X, Y)\); this is important when, for example, the \((X, Y)\) points lie on a sub-manifold.
3. The volume of \(K_{\hat{X},\hat{Y}}\) should be maximum. This implies that the points \((X^o, Y^o)\) are widely spread. In terms of the singular functions of the kernel

\[
(X^o, Y^o)
\]
\( K \), this corresponds heuristically to adding points in regions when the singular functions are rapidly varying (e.g., near the boundary of the domain). In practice, computing these optimal sets is very expensive since it requires solving an NP-hard combinatorics problem (e.g., finding a permutation that maximizes the volume). But we will see that the heuristics we propose can lead to near-optimal solutions at a fraction of the computational cost.

1.4. Contribution. In this paper, we adopted four strategies of computing \( X^o \) and \( Y^o \) as input for Algorithm 1.1. Numerical experiments are conducted to compare their performance (in terms of sizes of the initial sets \( X^o, Y^o \) and the resulting sets \( \hat{X}, \hat{Y} \)) of each strategy. The goal is to obtain an algorithm such that

1. the method has a complexity of \( O(r(m + n)) \) where \( r \) is the target rank;
2. the method is efficient, accurate, and stable;
3. the method is applicable to complicated geometries;
4. the method is simple to implement.

We motivate and explain the four methods of selecting \( X^o \) and \( Y^o \) in Section 2 and Section 3 presents numerical experiments to compare the four.

2. Selecting the initial interpolation points. Consider Eq. (1.10) and notice that is can be rewritten

\[
K_{X,Y} \approx K_{\hat{X},\hat{Y}} K^{-1}_{\hat{X},\hat{Y}} K_{\hat{X},Y} = (K_{X,\hat{Y}} K^{-1}_{\hat{X},\hat{Y}}) K_{\hat{X},\hat{Y}} (K^{-1}_{\hat{X},\hat{Y}} K_{\hat{X},Y})
\]

\[
= S_{X,\hat{X}} K_{\hat{X},\hat{Y}} T_{Y,\hat{Y}}
\]

where \( S_{X,\hat{X}} \) and \( T_{Y,\hat{Y}} \) are Lagrange basis functions (not polynomials). Each column can be seen as one of \(|\hat{X}| \) (resp., \(|\hat{Y}| \)) basis functions evaluated at \( X \) (resp., \( Y \)). The multiplication by the node matrix \( K_{\hat{X},\hat{Y}} \) produces an interpolator evaluated at \( X \times Y \). Because of this representation, the points \( \hat{X} \) and \( \hat{Y} \) can be considered as interpolation points. Formally, the same statement holds for \( X^o \) and \( Y^o \), from which \( \hat{X} \) and \( \hat{Y} \) are constructed.

The selection of the initial sets \( X^o \) and \( Y^o \) is crucial. There are three choices.

- **Endo-vertices**: Choose initial points from the inside the set of discretization points, \( X^o \subseteq X \).
- **Exo-vertices**: Alternatively, choose initial points from outside the set of discretization points, \( X^o \cap X = \emptyset \).
- **Mixed-vertices**: If neither \( X^o \subseteq X \) nor \( X^o \cap X = \emptyset \) hold, the \( X^o \) are mixed, some in \( X \), others not in \( X \).

We have two examples of exo and endo vertices.

- The first exo-method is to choose \( X^o \) to be a tensor product of Chebyshev nodes to enclose the domain \( X \). The tensor grid may be skew, and is chosen to closely enclose the domain.
- The second exo-method is to enclose the domain \( X \) within a ball centered at the centroid of \( X \), and choose \( X^o \) to be well-dispersed points on the surface of the ball.

Exo methods have an advantage that once a tensor grid of points or points on a sphere have been selected, their interpolation matrices can be used for any domain enclosed by the tensor grid. This property is not shared by the endo-methods, where the choice of interpolation points are a function of the domain.

- The first endo-method is to choose \( X^o \subseteq X \) to be a set of “maximally dispersed” vertices, i.e., we try and maximize the minimum distance between
any two vertices in $X^\circ$.

- The second endo-method is to choose $X^\circ \subset X$ to be a random set of vertices. Endo methods have the advantage of being nested, when we need more accuracy, we can add the next few vertices in the sequence and save on computation.

We now describe these four methods in detail.

2.1. Tensor grids formed of Chebyshev nodes. Polynomial interpolation at Chebyshev nodes is a popular approach to approximate low-rank kernel matrices, used for time-domain cosine expansion of large stationary covariance matrices [45], and general least-square analysis [14].

The idea of Chebyshev interpolation [18, 37], is simple — the more differentiable the kernel function $K$, the faster the coefficients of its Chebyshev expansion decay. This means that the kernel matrix has singular values that decay, and a low rank representation of the matrix is a good approximation. This is independent of the choice of discretization points $X$ and $Y$ inside domains $X$ and $Y$.

Consider a kernel $K$ defined over the domains $X \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}^d$. The kernel can be approximated using a polynomial interpolation rule as

$$K(x, y) \approx \sum_{x^\circ \in X^\circ} \sum_{y^\circ \in Y^\circ} S_{x^\circ}(x) \cdot K(x^\circ, y^\circ) T_{y^\circ}(y)$$

where $X^\circ$ and $Y^\circ$ are Chebyshev interpolation points and $S_{x^\circ}(x)$ and $T_{y^\circ}(y)$ are Lagrange polynomials. In 1D over $[-1, 1]$, the $m$ Chebyshev nodes (and their associated integration weights) are defined as

$$x_k^\circ = \cos \left( \frac{2k-1}{2m} \pi \right), \quad w_k^\circ = \frac{\pi}{m} \sin \left( \frac{2k-1}{2m} \pi \right)$$

for $k = 1, \ldots, m$.

See [6] for further details about practical Chebyshev interpolation.

In order to accommodate arbitrary geometries, we build a tensor grid where the numbers of points along each dimension are $n_1$, $n_2$ and $n_3$ for a total number of points $N = n_1 n_2 n_3$. The $n_i$ are chosen in rough proportion to the lengths of the sides of the enclosing box. We use a simple principle component analysis (PCA) to find the orientation of the bounding box. The lengths of the sides of the box are chosen to enclose the domain. This defines $X^\circ$ and $Y^\circ$. The associated weights $W_{X^\circ, X^\circ}, W_{Y^\circ, Y^\circ}$ are the products of the associated one-dimensional integration weights.

Note that while this approach is justified by the existence of the interpolation (cf. Eq. (2.1)), we merely rely on the nodes and the weights. There is no need to ever build or evaluate the associated Lagrange basis functions.

Benchmark tests in [11] show that Si-Chebyshev works well when the sizes of $X$ and $Y$ are large, $O(10,000)$. However, when $X$ and $Y$ are small, $O(100)$, the construction of the interpolation grids from Chebyshev expansion may be inefficient (the initial rank $r_0$ has to be too large for a given tolerance).

Since the 1-D Chebyshev points are clustered toward the endpoints of the interval, the tensor grid points are clustered in the corners. Unless the domains fill out the corners of the enclosing boxes, a corner may not be the best place to place interpolation points. The second exo-method distributes the points in a more even fashion.

2.2. Points on a sphere. The second method of constructing exo-vertices $X^\circ$ and $Y^\circ$ is to evenly distribute points on a sphere that encloses the domain. For kernels that satisfy Green’s theorem, points on an enclosing surface are sufficient. Since for a
geometry of dimension \(d\), we only need to build \(X^\circ\) and \(Y^\circ\) on a manifold of dimension \(d - 1\), this approach can significantly reduce the size of \(X^\circ\) and \(Y^\circ\).

Green’s third identity represents the far-field using an integration over the boundary instead of the whole domain. Consider points \(x\) in \(X\) and \(y\) in \(Y\). Define a surface \(\Gamma\) enclosing \(x\). The exterior domain \(\Omega \subset \mathbb{R}^3\) with boundary \(\Gamma\) contains \(y\) but not \(x\).

The function \(\mathcal{K}(x, y)\) satisfies the boundary value problem

\[
\Delta_y u(y) = 0 \quad y \in \Omega \\
u(y) = \mathcal{K}(x, y) \quad \text{for all } y \in \Gamma
\]

Generally, the function \(u\) satisfies the following representation formula [1, 5, 10, 29]

(2.2) \[
\int_{\Gamma} \left\{ [u(y)]_{\Gamma} \frac{\partial \mathcal{K}}{\partial n_1}(z, y) - \left[ \frac{\partial u}{\partial n_2}(x, z) \right]_{\Gamma} \mathcal{K}(z, y) \right\} dz
\]

for all \(y \in \mathbb{R}^3 \setminus \Gamma\), where \([ \ ]_{\Gamma}\) denotes the jump across \(\Gamma\). Assume the field \(u\) is continuous across the boundary \(\Gamma\). Note that it is then equal to \(\mathcal{K}(x, y)\) for \(y \in \Omega\), but, being smooth for all \(y \in \mathbb{R}^3 \setminus \Gamma\), \(u\) is not equal to \(\mathcal{K}(x, y)\) for \(y \notin \Omega\) (for example near \(x\) where \(\mathcal{K}(x, y)\) is singular). With this choice \(\|u(y)\|_{\Gamma} = 0\), we simplify Eq. (2.2).

(2.3) \[
\mathcal{K}(x, y) = -\int_{\Gamma} \left[ \frac{\partial u}{\partial n_2}(x, z) \right]_{\Gamma} \mathcal{K}(z, y) \, dz
\]

This implies that it is possible to find “pseudo-sources” on the surface of \(\Gamma\) that will reproduce the field \(\mathcal{K}(x, y)\) on \(\Omega\) [34]. This is the motivation behind using points on a sphere, since for instance \(\mathcal{K}(x, y) = \|x - y\|^{-1}\) satisfies the potential equation. However, not all kernels satisfy this equation. In particular \(\Delta_x \|x - y\|^{-2} \neq 0\). As a result, one can’t apply Green’s theorem, and points on a sphere are not enough to interpolate this kernel.

This is illustrated in Section 3.2. A representation similar to Eq. (2.2) has been used before. One idea, the Green hybrid method (GrH), is explored in [9]. This method takes advantage of a two-step strategy. It first analytically approximates the kernel using an integral representation formula, then further compresses its rank by a factor of two using a nested cross approximation. Another approach is to use Green’s formula to place a bounding circle around the domain, and then spread interpolation points equally spaced around the circle.

In [7] and [9], both \(\mathcal{K}\) and its normal derivative appear, while our approach only uses \(\mathcal{K}\) on the uniformly distributed points on the bounding sphere. Similar to our method, “pseudo-points” [34] anchor the multipole expansion to approximate the potential field.

In this paper, to deal with 3D geometries, we use a Fibonacci lattice [20, 31] to spread uniformly distributed points \(x \in X^\circ\) on a 3D bounding sphere.

\[
x_i = c_M + r_M \begin{bmatrix} r_i \cos \theta_i & r_i \sin \theta_i & z_i \end{bmatrix}^T
\]

\[
\Delta \theta = (3 - \sqrt{5})\pi \\
\Delta z = \frac{2}{n - 1} \\
\theta_1 = \Delta \theta \\
z_1 = -1 \\
\theta_i+1 = \theta_i + \Delta \theta \\
z_i+1 = z_i + \Delta z \\
r_i = \sqrt{1 - z_i^2}
\]
An obvious extension is to include points interior to the sphere, i.e., place interpolation points on nested spheres.

2.3. Random vertices. The first endo-method is very simple, simply select a set of interpolation vertices $X^o$ from $X$ at random [28, 26, 35, 36]. For large $X$, this is reasonable, but in general, this approach is not as good as our second endo-method.

The scaling matrix $W_{X^o,X^o}$ is diagonal, with $W_{x_i,x_i}$ proportional to the “area” of a patch surrounding vertex $x_i$. We can define area this patch as the weight of the vertices that are close to $x_i$.

Consider a vertex $v \in X$. If there is one interpolation vertex $x_i \in X^o$ that it is closest to, then give $x_i$ the weight of vertex $v$. If there are two or more interpolation points that are closest to $v$, then give an equal portion of the weight of $v$ to each of the closest interpolation points. With this definition of area, we define the scaling matrices $W_{X^o,X^o}$ and $W_{Y^o,Y^o}$.

2.4. Maximally dispersed vertices. If domains $X$ and $Y$ are well-separated, the range of $K_{X,Y}$ is well represented by interpolation vertices $X^o$ and $Y^o$ that are roughly uniformly distributed throughout $X$ and $Y$.

We construct a sequence of vertices $x_1, x_2, \ldots, x_m$ such that each leading subset is “maximally dispersed”, they are as far away from each other as possible.

We want to choose interpolation vertices $X^o$ from discretization vertices $X$. Choose a random vertex $u$ and find a vertex $v$ the furthest from $u$. The first interpolation vertex in $X^o$ is vertex $v$. To add another interpolation vertex, we look for a vertex that is the furthest distance from any interpolation vertex, choose $v \in X \setminus X^o$ with a maximum minimum distance.

\[
\min_{x^o \in X^o} \text{dist}(v, x^o) = \max_{w \in X \setminus X^o} \left( \min_{x^o \in X^o} \text{dist}(w, x^o) \right)
\]

We use either the Euclidean distance metric or the graph distance metric. When we build the interpolation set up one vertex at a time, and at each step the equality holds, we say that these vertices are maximally dispersed. Figure 2.1 shows the process for 1, 10, 25 and 50 vertices in $X^o$ for a triangulated disk $X$, triangulated with 351 discretization points $X$.

![Fig. 2.1: MDV initial interpolation points $X^o$ for a circular disk $X$, triangulated with 351 discretization points $X$.](image)

2.5. Discussion. We compare four methods: exo-methods SI-Chebyshev and SI-sphere, and endo-methods SI-MDV and SI-random.
Fig. 2.2: (Relative) Frobenius error $\|K_{X,Y} - K_{X,\hat{Y}}K_{X,\hat{Y}}^{-1}K_{X,Y}\|_F$ when $\hat{X}$ and $\hat{Y}$ are built using Algorithm 1.1 (with the rank fixed a priori) with and without the weight matrices. $X$ and $Y$ are 20$^3$ points on two facing unit-cubes separated by a distance of 1, and $X^\circ$, $Y^\circ$ are tensor grids of $7^3$ Chebyshev nodes. $\mathcal{K}(x,y) = 1/r$.

- **Scaling matrices** The weight matrices $W_{X^\circ,X^\circ}$ and $W_{Y^\circ,Y^\circ}$ are needed so that the 2-norm of the rows (resp. columns) of $\hat{K}_{X^\circ,Y^\circ}$ properly approximates the $L_2$ norm of $\mathcal{K}$ over $X$ (resp. $Y$). Otherwise, a higher concentration of points in a given area may be given a higher than necessary importance during the columns (resp. rows) selection process in Algorithm 1.1. Figure 2.2 illustrates the impact on the Frobenius error (a proxy for the $L_2$ error over $X \times Y$) when using weights or not with SI-Chebyshev. We observe that the absence of weights leads to a higher Frobenius error (with a higher variance) in the result. This justifies the choice of weights matrices for the four methods:
  - **SI-sphere** does a very good job of distributing the points evenly on the sphere, so the areas of the patches are nearly constant. The scaling matrix for the unit sphere is close to $(4\pi/|X^\circ|)$ times the identity.
  - **SI-MDV** also does a good job of evenly distributing vertices, see Figure 2.1d. Its scaling matrix is nearly a constant diagonal.
  - For moderate numbers of initial vertices, **SI-random** can have some variation in areas, and so a non-unit scaling matrix should be used.
  - The nodes in a Chebyshev tensor grid are found mostly near the boundaries, the edges, and the corners. The scaling matrix for **SI-Chebyshev** is necessary for the selection of interpolation points $\hat{X}$.

- **Directional bias**
  All methods that evenly distribute points can suffer from loss of accuracy for near-field matrices. Consider below where we show four steps of choosing a good set of dispersed vertices for a rectangular domain $X$. The $Y$ domain is two diameters to the northeast of $X$. We compute the RRQR factorization of the kernel matrix $K_{X,Y}$ and show the first 5, 10, 15 and 20 vertices chosen as pivots. There is a definite bias to the northeast, in the direction of the $Y$ domain. When the two domains are closer, this bias is more pronounced.
SI-MDV, SI-sphere and so some extent SI-random will need a more dense uniformly dispersed initial set $X^\circ$ to be able to provide enough vertices in the necessary area of concentration. Conversely, SI-Chebyshev may not suffer too badly, since its nodes are largely found near the boundary.

- **Further compression of the low-rank approximation**

  Given an approximation

  \[
  \|K_{X,Y} - K_{X,\hat{Y}}K_{\hat{X},\hat{Y}}\|_F \leq \varepsilon \|K_{X,Y}\|_F
  \]

  one can always, no matter the sets $\hat{X}$ and $\hat{Y}$, obtain a re-compressed approximation with a rank equal to or near the optimal $\varepsilon$-rank of $K_{X,Y}$.

  To see this, consider a matrix $A$ and a matrix $B$ of rank $r_1$ (in our problem, $A = K_{X,Y}$ and $B = K_{X,\hat{Y}}K_{\hat{X},\hat{Y}}K_{\hat{X},\hat{Y}}$). The key is that

  \[
  \|A - B\|_2 \leq \varepsilon \Rightarrow |\sigma_k(A) - \sigma_k(B)| \leq \varepsilon.
  \]

  This is a direct consequence of corollary 8.6.2 in [19]. Intuitively, this means the singular values are continuous, and so up to $\varepsilon$, $A$ and $B$ have roughly the same singular values. One can compress $B$ to another low-rank matrix with an even smaller rank, $r_2$, closer to the $\varepsilon$-rank of $A$.

  \[
  K_{X,\hat{Y}}K_{\hat{X},\hat{Y}}^{-1}K_{\hat{X},\hat{Y}} = (Q_{X,\alpha}R_{\alpha,\hat{Y}})K_{\hat{X},\hat{Y}}^{-1}(Q_{Y,\beta}R_{\beta,\hat{X}})^\top \\
  = Q_{X,\alpha}(R_{\alpha,\hat{Y}}K_{\hat{X},\hat{Y}}^{-1}R_{\beta,\hat{X}}^\top)Q_{Y,\beta}^\top \\
  \approx Q_{X,\alpha}(U_{\alpha,\gamma}\Sigma_{\gamma,\gamma}V_{\beta,\gamma}^\top)Q_{Y,\beta}^\top \\
  = (Q_{X,\alpha}U_{\alpha,\gamma})(\Sigma_{\gamma,\gamma}Q_{Y,\beta}V_{\beta,\gamma}) = U_{X,\gamma}W_{Y,\gamma}
  \]

  The error in the truncation of the singular values of $R_{\alpha,\hat{Y}}K_{\hat{X},\hat{Y}}^{-1}R_{\beta,\hat{X}}$ (from the second to third line) has not been further amplified since $Q_{X,\alpha}$ and $Q_{Y,\beta}$ are orthogonal.

  The singular values of $K_{X,Y}$ and $R_{\alpha,\hat{Y}}K_{\hat{X},\hat{Y}}^{-1}R_{\beta,\hat{X}}^\top$ are close up to $\varepsilon$. Hence, by truncating up to $\varepsilon$, we can expect to recover the $\varepsilon$ rank of $K_{X,Y}$. We illustrate this result on a concrete example in Section 3.5.

- **Computational complexities**

  We summarize the computational complexities of the Skeletonized Interpolation and the recompression process, where we assume

  \[
  |X| = |Y| = n, \ |X^\circ| = |Y^\circ| = r_0 \leq n, \text{ and } |\hat{X}| = |\hat{Y}| = r_1 \leq r_0.
  \]

  Furthermore, we assume there that the cost of evaluating the kernel on $X \times Y$ is $O(|X||Y|)$.

  For Skeletonized Interpolation, the leading cost is the GenInitSet algorithm (if using MDV), the RRQR’s and the construction of the left and right factors. Many algorithms exist to compute RRQR’s; in our experiments, we use a simple column-pivoted QR algorithm with cost $O(r_0^2r_1)$ ([19], page 278).
Skeletonized Interpolation
(Algorithm 1.1)

| GenInitSet | kernel evaluation | linear algebra |
|------------|-------------------|----------------|
| build $K_{X^*,Y^*}$ | $O(r_0^2)$ | $O(nr_0)$ for MDV $O(r_0)$ otherwise |
| RRQR of $K_{X^*,Y^*} K_{X^*,Y^*}^\top$ | $O(nr_1)$ | $O(r_0^2 r_1)$ |
| build $K_{\hat{X},\hat{Y}}$ | $O(r_1^2)$ | $O(r_1^2) O(nr_1 + r_0^2)$ for MDV $O(r_0^2 r_1)$ otherwise |
| LU factorization of $K_{\hat{X},\hat{Y}}$ | | |
| total | $O(nr_1 + r_0^2)$ | $O(nr_1 + r_0^2 r_1)$ for MDV $O(r_0^2 r_1)$ otherwise |

The goal is to have $r_1$ be close to the numerical rank of $K_{X,Y}$, but if it is larger, as we will see is the case for SI-Chebyshev, then it will pay to recompress the matrix.

Recompression (Eq. (2.7))

| linear algebra |
|----------------|
| QR of $K_{X,Y}$ and $K_{\hat{X},\hat{Y}}^\top$ | $O(nr_1^2)$ |
| compute $R_{\alpha,\hat{Y}} K_{\hat{X},\hat{Y}} R_{\beta,\hat{X}}$ | $O(r_1^3)$ |
| SVD factorization of $R_{\alpha,\hat{Y}} K_{\hat{X},\hat{Y}} R_{\beta,\hat{X}}^\top$ | $O(r_1^3)$ |
| compute $U_{X,\gamma}$ and $V_{Y,\gamma}$ | $O(nr_1 r_2)$ |
| total | $O(nr_1^2)$ |

3. Numerical experiments. We describe numerical experiments on a set of three meshes, a torus, a plate and coil modeling electromagnetic forming, and an engine block from acoustic analysis. We use two kernel functions, $1/r$ and $1/r^2$.

$$\mathcal{K}(x,y) = \frac{1}{\|x-y\|_2} \quad \text{or} \quad \mathcal{K}(x,y) = \frac{1}{\|x-y\|_2^2}$$

The three meshes were generated by LS-DYNA [30] from Livermore Software Technology Corporation. The torus and plate-coil cases are from the electromagnetic solver (see [33]) and the engine case is from the acoustics solver (see [32]).

![Torus geometry. Inner and outer radii are 3 and 8, respectively](image1.png)

![Plate-coil geometry](image2.png)

![Engine geometry](image3.png)

Fig. 3.1: The three benchmarks geometries.

The three meshes are moderate in size, $O(10,000)$ discretization points, domain
size $|X|$, where $250 \leq |X| \leq 650$. The standard deviation is a measure of the load imbalance due to subdomains of different sizes. Note, the torus mesh has 128 domains of equal size.

| Geometry  | # Vertices | # domains | Min. | Max. | Mean | Std Dev. |
|-----------|------------|-----------|------|------|------|----------|
| torus     | 32,768     | 128       | 256  | 256  | 256  | 0.0      |
| plate-coil| 20,794     | 64        | 309  | 364  | 325  | 15.4     |
| engine    | 78,080     | 128       | 573  | 647  | 610  | 4.6      |

We study each of the four algorithms to construct the interpolation points. To compute the $X^\circ$, $Y^\circ$, $\hat{X}$ and $\hat{Y}$ sets, we increase the sizes of $X^\circ$ and $Y^\circ$ by a factor $\omega = 1.1$ and perform SI until the relative error

$$\left\| K_{X,Y} - K_{X,\hat{Y}} K_{\hat{X},\hat{Y}}^{-1} K_{\hat{X},Y} \right\|_F \leq \varepsilon^*$$

is smaller than a desired tolerance $\varepsilon^*$. We set the initial $r_0 = 1$ for all strategies but SI-Chebychev, for which we use $r_0 = 8$ (2 nodes in each 3 dimensions).

This error tolerance $\varepsilon^*$ is different from the $\varepsilon$ in Algorithm 1.1, which is used to extract $\hat{X}$ and $\hat{Y}$ based on the $\varepsilon$-rank. Here, we use $\varepsilon = 0.1\varepsilon^*$. We vary $\varepsilon^* = 10^{-3}, \ldots, 10^{-10}$ and collect the resulting ranks $r_0 = |X^\circ| = |Y^\circ|$ and $r_1 = |\hat{X}| = |\hat{Y}|$. We measure the distance between two domains $X_i$ and $X_j$ by the distance ratio $dr(i,j)$.

$$dr(i,j) = \frac{\text{dist}(c_i, c_j)}{\min(r_i, r_j)} = \frac{\text{dist}(\text{centroid}(X_i), \text{centroid}(X_j))}{\min(\text{radius}(X_i), \text{radius}(X_j))}$$

The four strategies are applied to compute low rank factorizations for all subdomain pairs $(i, j)$ with $dr(i, j) \geq 1$.

The torus mesh has perfect load balance in subdomain size, while plate-coil and engine have irregularities, in size and in shape. The bounding boxes of SI-Chebychev and the spheres of SI-sphere fit closely around the irregular domains.

For the sake of simplicity, the experiments used the identity matrix as weights matrix for SI-MDV and SI-random, since they generate roughly uniformly distributed points in the volume. SI-Chebychev and SI-sphere used the weights described in Section 2.

For each mesh and the $1/r$ kernel, we used SI-Chebychev, SI-sphere, SI-MDV and SI-random to compute $r_0 = |X^\circ| = |Y^\circ|$ and $r_1 = |\hat{X}| = |\hat{Y}|$. We split the submatrix pairs into three sets using the distance ratio — near-field, mid-field and far-field submatrices. For each accuracy, for each set we compute mean values of $r_0$ and $r_1$, which we compare to the mean SVD rank at that accuracy.

The results for the $1/r^2$ kernel are very close to that of the $1/r$ kernel, except for one case. In Section 3.2, we see that SI-sphere does not work for this kernel, as anticipated. An enclosing surface does not suffice, interior vertices must be included in $X^\circ$.

3.1. torus : 32,768 vertices, 128 domains.

There are 8128 off-diagonal submatrices in the lower block triangle. Of these, 7,040 pairs had distance ratio $dr > 1$, which we split into three equally sized sets to represent near-field, medium-field and far-field pairs. Figure 3.2 plots mean values of $r_0$ and $r_1$ for the three different sets of submatrices. In the torus case, all the subdomains
are of the same shape and all the clusters have a same distribution. Due to the large distance ratio, SVD ranks are lower than 20 for all given tolerances. We observe that to achieve a tolerance of $\varepsilon^* = 10^{-10}$, we only require $r_0$ less than 20% of the subdomain size $|X|$. Since the ranks are overall small, $r_0$ grows fairly slowly.

While SI-Chebyshev and SI-sphere have similar $r_0$ for all tolerances, “$r_1$ Chebyshev” is less than “$r_1$ sphere” (see Figures 3.2b, 3.2d and 3.2f). This means that using SI-sphere is less efficient and SI-Chebyshev can compress the rank further.

As shown in Figures 3.2a, 3.2c and 3.2e, “$r_0$ MDV” is usually smaller than “$r_0$ random”. Therefore, using SI-MDV can substantially reduce the sizes of $X^o$ and $Y^o$ thus it is more efficient than SI-random. However, Figures 3.2d and 3.2f suggest that sometimes applying RRQR on “$r_0$ MDV” is less effective than on “$r_0$ random”. When the tolerance is high, “$r_1$ MDV” is slightly larger than “$r_1$ random”. When the tolerance is low, “$r_0$ MDV” $\approx$ “$r_1$ MDV”. This is because of the MDV heuristic works very well in uniformly distributed clusters thus we already found $\hat{X}$ and $\hat{Y}$ as the MDV sets and do not need to do RRQR to further compress the rank.

3.2. plate-coil : 20,794 vertices, 64 domains.

There are 2,016 off-diagonal submatrices, of these 1,212 pairs have distance ratio greater than one. Figure 3.3 shows plots of mean values of $r_0$ and $r_1$ for near-field, medium-field and far-field submatrices.

Unlike torus, the subdomains in plate-coil are irregularly shaped, and the domains do not have uniformly distributed points. Because plate-coil mesh is smaller in size and in the number of domains, the distances are shorter. The closer two subdomains, the higher the SVD rank. This is also reflected in the size of the initial sets $r_0 = |X^o|$, up to half the size of the domain for high accuracy near-field submatrices.

As the distance ratio increases, both $r_0$ and $r_1$ decrease for all four strategies. The relative performance of the four strategies is similar to that of torus case. Although in Figures 3.3c and 3.3e “$r_0$ Chebyshev” is higher than “$r_0$ Sphere”, “$r_1$ Chebyshev” is always less than “$r_1$ Sphere” (see Figures 3.3b, 3.3d and 3.3f). Thus SI-Chebyshev can compress the rank further than SI-sphere.

As for SI-MDV and SI-random (Figures 3.3a, 3.3c and 3.3e), “$r_0$ MDV” is much smaller than “$r_0$ random” thus SI-MDV is more efficient. The final ranks of SI-MDV and SI-random are nearly equal to the svd ranks, and further compression is not needed. However, the final ranks of SI-Chebyshev and SI-sphere are greater than the svd ranks, and further compression is needed.

Figure 3.4 shows the computational results with the kernel $\mathcal{K}(x,y) = ||x-y||^{-2}_2$. The curve of SI-sphere is only partially shown. This is because when the given tolerance is less than $10^{-4}$, $r_0$ of SI-sphere blows up and exceeds 500 in our experiments. This means that building $X^o$ and $Y^o$ using vertices on a sphere does not work with this kernel, as explained in Section 2.2.

3.3. engine : 78,080 vertices, 128 domains. The results are depicted on Figure 3.5. As for plate-coil, the domains are irregular in size and shape. However, $r_0$ here is less than that of the plate-coil case (about 20% of the subdomain size when the tolerance is low), due to the larger distance ratios. The relative performance of the four strategies is very similar to plate-coil.

3.4. Cases where MDV is inefficient. The drawback of MDV is that it a priori selects points uniformly in the volume (or using a graph-distance). In most cases, this ends up working very well. However, for kernels that are also Green’s
Fig. 3.2: The mean of the computed ranks in torus, $r_0$ on the left, $r_1$ on the right, for 2,347 pairs $dr \in [1, 4.41]$ (top), for 2,347 pairs $dr \in [4.41, 6.59]$ (middle) and for 2,346 pairs $dr \in [6.59, 7.87]$ (bottom).
Fig. 3.3: The mean of the computed ranks for 404 pairs in the plate-coil case, $dr \in [1, 1.58]$ (top), $dr \in [1.58, 2.21]$ (middle) and $dr \in [2.21, 3.28]$ (top). In Figure 3.3b and Figure 3.3f, SI-random is behind SI-MDV.
functions (e.g., when solving some integral equations), it may be sufficient to place interpolation points on an enclosing surface. Consider for example \( \log(\|x-y\|) \) in 2D. We know that this kernel can be approximated using multipole functions. To obtain a multipole expansion of order \( p \), we need only \( O(p) \) coefficients \[34\]; this is in contrast to general expansions such as Taylor or Chebyshev that require \( O(p^2) \) terms. Moreover, one can always find a set of equivalent \( p \) “charges” on an enclosing circle that lead to the same first \( p \) terms in the multipole expansion \[34\]. This implies that to reach order \( p \), \text{SI-sphere} requires \( O(p) \) points only. 

\text{SI-MDV} requires about the same density of points. The problem is that if the points in \( X \) and \( Y \) are uniformly distributed in the volume, then in order to get the correct density of points near the boundary one needs \( r_0 = O(p^2) \) points in \text{SI-MDV}. This is reflected by a much larger value of \( r_0 \) in the case of \text{SI-MDV} compared to \text{SI-sphere}. See Figure 3.6 for an illustrative benchmark. In particular, see the distribution of points for MDV after RRQR (\( \hat{X} \) and \( \hat{Y} \)), and how it tries to approximate the distribution from Sphere.

3.5. Example of recompression. To illustrate the effect of the recompression step discussed in Section 2.5 on one specific pair of the coil geometry. Figure 3.7 shows, as expected, that the newly obtained low-rank approximations all have ranks \( r_2 \leq r_1 \) very close to the SVD rank of \( K_{X,Y} \). Experiments on other pairs of clusters produce similar results.

4. Conclusion. We introduced four heuristics for selecting an initial set of points for the Skeletonized Interpolation method: Chebyshev grids, Maximally Dispersed Vertices, points on a sphere, and random sampling. Some of these methods use endo-points, i.e., a subset of the given points (MDV and random), while others use exo-points, i.e., they introduce new points for the interpolation (Chebyshev and sphere).

These methods should be considered as a way to build an initial low-rank approximation at the smallest possible cost. Once a low-rank factorization exists, it can always be further compressed to have a near-optimal rank. 

\text{SI-Chebyshev} is robust with guaranteed accuracy, even for complicated geometries and very large clusters. But it can be inefficient when points nearly lie on a
Fig. 3.5: The mean of the computed ranks for 2150 near-field pairs in the engine case, \( d r \in [1, 1.87] \) (top), \( d r \in [1.87, 2.84] \) (middle) and \( d r \in [2.84, 7.46] \) (bottom). In Figure 3.5b, SI-random is hidden behind SI-MDV and SVD.
Fig. 3.6: Example where MDV does worse than the points on a sphere. In this geometry, we arrange 833 points uniformly in two 2D disks of the same shape, separated by less than one diameter. The kernel is the 2D Laplacian kernel, $\mathcal{K}(x, y) = \log(\|x - y\|_2)$. We observe than arranging points on a circle leads to smaller $X^\circ$ and $Y^\circ$ for a given accuracy. Chebyshev and Random give similar results (in terms of ranks $r_0$) than MDV points and vertices on a sphere, respectively, and are omitted for clarity.

SI-MDV, as a heuristic, is efficient and accurate, and very simple to implement. It may become inefficient for specific kernels, like Green’s functions of Laplace equation. In that case, a large set of initial points ($r_0$) may be required.

SI-Sphere constructs initial points on a 2D surface instead of a 3D geome-
try. This largely reduces the number of initial points, asymptotically. However, this method only works for kernels that are Green’s functions of Laplace equation, and tends to overestimate the final rank \( (r_1) \). When points are distributed uniformly in the volume (rather than on a submanifold), \textbf{SI-sphere} does very well for those Green’s functions.

\textbf{SI-Random} is robust and general and is the easiest to implement. Nevertheless, randomly sampling the points can lead to redundancy thus making the sizes of initial points too large.

Three benchmark tests are performed, in which the four SI methods are applied to torus, plate-coil and engine geometries. The main conclusions are summarized below:

1. A comparison between \textbf{SI-Sphere} and \textbf{SI-Chebyshev} shows that the final rank approximated by \textbf{SI-Chebyshev} is lower than that of \textbf{SI-sphere} in all cases, thus numerically \textbf{SI-Chebyshev} is more accurate than \textbf{SI-sphere}.

2. Given a pair of clusters, \textbf{SI-MDV} builds an MDV set whose size is much smaller than the randomly sampled set constructed by \textbf{SI-random}. Therefore, \textbf{SI-MDV} is more efficient than \textbf{SI-random}.

3. Compared with \textbf{SI-MDV}, \textbf{SI-Chebyshev} always constructs a larger size for both \( X^\circ \) and \( \bar{X} \), thus is less efficient and accurate than \textbf{SI-MDV}. This result suggests that in dealing with small clusters or complicated geometries, using \textbf{SI-Chebyshev} is not advantageous, despite its theoretically guaranteed accuracy and robustness. As another special case of SI method, \textbf{SI-MDV} method is a good complement for \textbf{SI-Chebyshev}.

4. When points are distributed uniformly inside the volume and the kernel is a \textbf{Green’s function}, \textbf{SI-sphere} does very well and is superior to \textbf{SI-MDV}. The reason is that the optimal choice of points consists in choosing points near the boundary of the domain, while \textbf{SI-MDV} samples points uniformly in the volume. This leads to overestimating \( r_0 \) with MDV.

Overall, for small clusters, we recommend using \textbf{SI-MDV}, for specific Green’s functions, \textbf{SI-Sphere}, for large clusters, \textbf{SI-Chebyshev}, and \textbf{SI-Random} as the
simplest algorithm to implement.

REFERENCES

[1] Banerjee, P. K., and Butterfield, R. Boundary element methods in engineering science, vol. 17. McGraw-Hill London, 1981.
[2] Barnes, J., and Hut, P. A hierarchical $O(N \log N)$ force-calculation algorithm. Nature 324 (Dec 1986), 446–449.
[3] Bebendorf, M. Approximation of boundary element matrices. Numerische Mathematik 86, 4 (Oct 2000), 565–589.
[4] Bebendorf, M., and Rjasanow, S. Adaptive low-rank approximation of collocation matrices. Computing 70, 1 (Mar. 2003), 1–24.
[5] Becker, A. A. The boundary element method in engineering: a complete course. McGraw-Hill Companies, 1992.
[6] Berrut, J.-P., and Trefethen, L. N. Barycentric Lagrange interpolation. SIAM Review 46 (2004), 501–517.
[7] Börn, S., and Gördes, J. Low-rank approximation of integral operators by using the green formula and quadrature. Numerical Algorithms 64, 3 (2013), 567–592.
[8] Börn, S., and Grasedyck, L. Low-rank approximation of integral operators by interpolation. Computing 72, 3 (Jun 2004), 325–332.
[9] Börn and Sven Christophersen, S. Approximation of integral operators by green quadrature and nested cross approximation. Numerische Mathematik 133 (2016), 409–442.
[10] Brebbia, C. A. The boundary element method for engineers. Pentech press, 1980.
[11] Cambier, L., and Darve, E. Fast Low-Rank Kernel Matrix Factorization through Skeletonized Interpolation. ArXiv e-prints 1706.02812 (Jun 2017).
[12] Chew, W., Song, J., Cui, T., Velamparambil, S., Hastriter, M., and Hu, B. Review of large scale computing in electromagnetics with fast integral equation solvers. CMES - Computer Modeling in Engineering and Sciences 5, 4 (2004), 361–372.
[13] Ciivil, A., and Magdon-Ismail, M. On selecting a maximum volume sub-matrix of a matrix and related problems. Theoretical Computer Science 410, 47 (2009), 4801 – 4811.
[14] Cole, W., Hohs, G., Champion, D. J., Manchester, R. N., and Verbiest, J. P. W. Pulsar timing analysis in the presence of correlated noise. Monthly Notices of the Royal Astronomical Society 418, 1 (2011), 561–570.
[15] Darve, E. The fast multipole method I: error analysis and asymptotic complexity. SIAM Journal on Numerical Analysis 38, 1 (2000), 98–128.
[16] Darve, E. The fast multipole method: numerical implementation. Journal of Computational Physics 160, 1 (2000), 195–240.
[17] Fonarev, A., Mikhailov, A., Serdyukov, P., Gusev, G., and Oseledets, I. Efficient rectangular maximal-volume algorithm for rating elicitation in collaborative filtering. In Data Mining (ICDM), 2016 IEEE 16th International Conference on (2016), IEEE, pp. 141–150.
[18] Gong, W., and Darve, E. The black-box fast multipole method. J. Comput. Physics 228 (2009), 872–8725.
[19] Golub, G. H., and Van Loan, C. F. Matrix Computations, fourth ed. JHU Press, 2013.
[20] Gonzalez, Á. Measurement of areas on a sphere using fibonacci and latitude–longitude lattices. Mathematical Geosciences 42, 1 (2010), 49.
[21] Goreinov, S., Tyrtyshnikov, E., and Zamarashkin, N. A theory of pseudoskeleton approximations. Linear Algebra and its Applications 261, 1 (1997), 1 – 21.
[22] Goreinov, S. A., and Tyrtyshnikov, E. E. Quasioptimality of skeleton approximation of a matrix in the chebyshev norm. Doklady Mathematics 83, 3 (Jun 2011), 374–375.
[23] Goreinov, S. A., Zamarashkin, N. L., and Tyrtyshnikov, E. E. Pseudo-skeleton approximations by matrices of maximal volume. Mathematical Notes 82, 4 (1997), 515–519.
[24] Greengard, L., and Rokhlin, V. A fast algorithm for particle simulations. Journal of Computational Physics 135, 2 (1997), 280 – 292.
[25] Gu, M., and Eisenstat, S. C. Efficient algorithms for computing a strong rank-revealing qr factorization. SIAM Journal on Scientific Computing 17, 4 (1996), 848–869.
[26] Gu, Y., Yu, W., and Li, Y. Efficient randomized algorithms for adaptive low-rank factorizations of large matrices. arXiv preprint arXiv:1606.09402 (2016).
[27] Hackbusch, W., and Nowak, Z. P. On the fast matrix multiplication in the boundary element method by panel clustering. Numerische Mathematik 54, 4 (Jul 1989), 463–491.
[28] Halvorsen, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review 53,
2 (2011), 217–288.

[29] Hall, W. S. Boundary element method. In The Boundary Element Method. Springer, 1994, pp. 61–83.

[30] Hallquist, J. O., et al. LS-DYNA Theory Manual. Livermore Software Technology Corporation, 1992.

[31] Hardy, G. H., and Wright, E. M. An introduction to the theory of numbers. Oxford university press, 1979. Page 209.

[32] Huang, Y., Souli, M., and Liu, R. BEM methods for acoustic and vibroacoustic problems in LS-DYNA. Proceedings of the 8th European LS-DYNA Users Conference (May 2011).

[33] L'Eplattenier, P., Cook, G., Ashcraft, C., Burger, M., Imbert, J., and Worswick, M. Introduction of an electromagnetism module in LS-DYNA for coupled mechanical-thermal-electromagnetic simulations. Steel Research International 80, 5 (2009), 351–358.

[34] Makino, J. Yet another fast multipole method without multipoles—pseudoparticle multipole method. Journal of Computational Physics 151, 2 (1999), 910 – 920.

[35] Martinsson, P.-G., and Voronin, S. A randomized blocked algorithm for efficiently computing rank-revealing factorizations of matrices. SIAM Journal on Scientific Computing 38, 5 (2016), S485–S507.

[36] Mary, T., Yamazaki, I., Kurzak, J., Luszczek, P., Tomov, S., and Dongarra, J. Performance of random sampling for computing low-rank approximations of a dense matrix on GPUs. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (2015), ACM, p. 60.

[37] Messner, M., Schanz, M., and Darve, E. Fast directional multilevel summation for oscillatory kernels based on Chebyshev interpolation. Journal of Computational Physics 231, 4 (2012), 1175 – 1196.

[38] Mikhailov, A., and Oseledets, I. Rectangular maximum-volume submatrices and their applications. Linear Algebra and its Applications 538 (2018), 187 – 211.

[39] Nishimura, N. Fast multipole accelerated boundary integral equation methods. Applied Mechanics Reviews 55 (Jul 2002), 299.

[40] Rokhlin, V. Rapid solution of integral equations of classical potential theory. Journal of Computational Physics 60, 2 (1985), 187 – 207.

[41] Savostyanov, D. V. Quasioptimality of maximum-volume cross interpolation of tensors. Linear Algebra and its Applications 458 (2014), 217 – 244.

[42] Schenk, L., and Gondzio, J. Rank revealing gaussian elimination by the maximum volume concept.

[43] Thurau, C., Kersting, K., and Bauckhage, C. Deterministic CUR for Improved Large-Scale Data Analysis: An Empirical Study. pp. 684–695.

[44] Tyryshnikov, E., Goreinov, S., and Zamarashkin, N. Pseudo skeleton approximations. Doklady Akademii Nauk 343, 2 (1995), 151–152.

[45] Van Haasteren, R., and Vallisneri, M. Low-rank approximations for large stationary covariance matrices, as used in the Bayesian and generalized-least-squares analysis of pulsar-timing data. Monthly Notices of the Royal Astronomical Society 446, 2 (2014), 1170–1174.

[46] Wrobel, L. The Boundary Element Method - Vol.1, Applications in Thermo-Fluids and Acoustics. The Boundary Element Method. John Wiley & Sons, 2002.

[47] Wu, Z., and Alkhalifah, T. The optimized expansion based low-rank method for wavefield extrapolation. GEOPHYSICS 79, 2 (2014), T51–T60.

[48] Yarvin, N., and Rokhlin, V. Generalized gaussian quadratures and singular value decompositions of integral operators. SIAM Journal on Scientific Computing 20, 2 (1998), 699–718.