SIAM J. SCI. COMPUT.
Vol. 41, No. 3, pp. A1652–A1680

FAST LOW-RANK KERNEL MATRIX FACTORIZATION USING SKELETONIZED INTERPOLATION*

LÉOPOLD CAMBIER† AND ERIC DARVE‡

Abstract. Integral equations are commonly encountered when solving complex physical problems. Their discretization leads to a dense kernel matrix that is block or hierarchically low-rank. This paper proposes a new way to build a low-rank factorization of those low-rank blocks at a nearly optimal cost of $\mathcal{O}(nr)$ for an $n \times n$ block submatrix of rank r. This is done by first sampling the kernel function at new interpolation points, then selecting a subset of those using a CUR decomposition and finally using this reduced set of points as pivots for a rank-revealing LU-type factorization. We also explain how this implicitly builds an optimal interpolation basis for the kernel under consideration. We show the asymptotic convergence of the algorithm, explain its stability, and demonstrate on numerical examples that it performs very well in practice, allowing us to obtain rank nearly equal to the optimal rank at a fraction of the cost of the naive algorithm.

Key words. low-rank, kernel, skeletonization, interpolation, rank-revealing QR, Chebyshev

AMS subject classifications. 15-04, 15B99, 45-04, 45A05, 65F30, 65R20

DOI. 10.1137/17M1133749

1. Introduction.

In this paper, we are interested in the low-rank approximation of kernel matrices, i.e., matrices K_{ij} defined as

$$K_{ij} = \mathscr{K}(x_i, y_j)$$

for $x_i \in X = \{x_1, \dots, x_m\} \subseteq \mathcal{X}$ and $y_j \in Y = \{y_1, \dots, y_n\} \subseteq \mathcal{Y}$ and where $\mathscr{K}$ is a smooth function over $\mathcal{X} \times \mathcal{Y}$. A typical example is when

$$\mathscr{K}(x, y) = \frac{1}{\|x - y\|_2}$$

and $X \subset \mathcal{X}$ and $Y \subset \mathcal{Y}$ are two well-separated sets of points.

This kind of matrices arises naturally when considering integral equations like

$$a(x)u(x) + \int_{\tilde{\mathcal{Y}}} \mathscr{K}(x, y)u(y)\mathrm{d}y = f(x) \quad \forall x \in \tilde{\mathcal{X}},$$

where the discretization leads to a linear system of the form

$$a_i u_i + \sum_j K_{ij} u_j = f_i, \tag{1}$$

where K is a *dense* matrix. While this linear system as a whole is usually not low-rank, one can select subsets of points $X \subset \mathcal{X}$ and $Y \subset \mathcal{Y}$ such that $\mathscr{K}$ is smooth over $\mathcal{X} \times \mathcal{Y}$ and hence $\mathscr{K}(X, Y)$ is low-rank. This corresponds to a submatrix of the complete K.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section June 9, 2017; accepted for publication (in revised form) January 11, 2019; published electronically May 21, 2019.

http://www.siam.org/journals/sisc/41-3/M113374.html

†Institute for Computational & Mathematical Engineering, Stanford University, Stanford, CA 94305 (lcambier@stanford.edu, https://stanford.edu/~lcambier).

‡Department of Mechanical Engineering, Stanford University, Stanford, CA 94305 (darve@stanford.edu).

TABLE 1
Notation used in the paper.

$\mathscr{K}$	The smooth kernel function.
$\mathcal{X}, \mathcal{Y}$	The spaces over which $\mathscr{K}$ is defined, i.e., $\mathcal{X} \times \mathcal{Y}$.
x, y	Variables, $x \in \mathcal{X}$, $y \in \mathcal{Y}$.
X, Y	The mesh of points over which to approximate $\mathscr{K}$, i.e., $X \times Y$.
K	The kernel matrix, $K = \mathscr{K}(X, Y)$, $K_{ij} = \mathscr{K}(x_i, y_j)$.
m, n	$m = \lvert X\rvert$, $n = \lvert Y\rvert$.
$\overline{X}$, $\overline{Y}$	The tensor grids of Chebyshev points.
$\widehat{X}$, $\widehat{Y}$	The subsets of $\overline{X}$ and $\overline{Y}$ output by the algorithm used to build the low-rank approximation.
$\overline{m}$, $\overline{n}$	The number of Chebyshev tensor nodes, $\overline{m} = \lvert\overline{X}\rvert$, $\overline{n} = \lvert\overline{Y}\rvert$.
r_0	The "interpolation" rank of $\mathscr{K}$, i.e., $r_0 = \min(\lvert\overline{X}\rvert, \lvert\overline{Y}\rvert)$.
r_1	The skeletonized interpolation rank of $\mathscr{K}$, i.e., $r_1 = \lvert\widehat{X}\rvert = \lvert\widehat{Y}\rvert$.
r	The rank of the continuous SVD of $\mathscr{K}$.
$S(x, \overline{X})$, $T(y, \overline{Y})$	Row vectors of the Lagrange basis functions, based on $\overline{X}$ and $\overline{Y}$ and evaluated at x and y, respectively. Each column is one Lagrange basis function.
$\widehat{S}(x, \widehat{X})$, $\widehat{T}(y, \widehat{Y})$	Row vectors of Lagrange basis functions, based on $\widehat{X}$ and $\widehat{Y}$, built using the skeletonized interpolation and evaluated at x and y, respectively. Each column is one function.
w_k, w_l	Chebyshev integration weights.
$\mathrm{diag}(\overline{W}_X)$, $\mathrm{diag}(\overline{W}_Y)$	Diagonal matrices of integration weights when integration is done at nodes $\overline{X}$ and $\overline{Y}$.

Being able to efficiently compute a low-rank factorization of such a submatrix would lead to significant computational savings. By "smooth" we usually mean a function with infinitely many continuous derivatives over its domain. Such a function can be well approximated by its interpolant at Chebyshev nodes, for instance.

Low-rank factorization means that we seek a factorization of $K = \mathscr{K}(X, Y)$ as

$$K = USV^\top,$$

where $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$, $S \in \mathbb{R}^{r \times r}$, and r is the rank. In that factorization, U and V don't necessarily have to be orthogonal. One way to compute such a factorization is to first compute the matrix K at a cost $\mathcal{O}(mn)$ and then to perform some rank-revealing factorization like SVD, rank-revealing QR, or rank-revealing LU at a cost usually proportional to $\mathcal{O}(mnr)$. But even though the resulting factorization has a storage cost of $\mathcal{O}((m+n)r)$, linear in the size of X and Y, the cost would be proportional to $\mathcal{O}(mn)$, i.e., quadratic.

1.1. Notation. In the following, we will denote by $\mathscr{K}$ a function over $\mathcal{X} \times \mathcal{Y}$. X and Y are finite sequences of vectors such that $X \subset \mathcal{X}$ and $Y \subset \mathcal{Y}$, and $\mathscr{K}(X, Y)$ denotes the matrix $K_{ij} = \mathscr{K}(x_i, y_j)$. Lowercase letters x and y denote arbitrary variables, while uppercase letters $\overline{X}$, $\widehat{X}$, $\check{X}$, $\widetilde{X}$ denote sequences of vectors. We denote matrices like $A(X, Y)$ when the rows refer to the set X and the columns to the set Y. Table 1 summarizes all the symbols used in this paper.

1.2. Previous work. The problem of efficiently solving (1) has been extensively studied in the past. As indicated above, discretization often leads to a dense matrix K_{ij}. Hence, traditional techniques such as the LU factorization cannot be applied because of their $\mathcal{O}(n^3)$ time and even $\mathcal{O}(n^2)$ storage complexity. To deal with such matrices, the now traditional method is to use the fact that they usually present a (hierarchically) low-rank structure, meaning we can represent the matrix as

a hierarchy of low-rank blocks. The fast multipole method (FMM) [28, 12, 2] takes advantage of this fact to accelerate computations of matrix-vector products Kv, and one can then couple this with an iterative method. More recently, [10] proposed a kernel-independent FMM based on interpolation of the kernel function.

Other techniques compute explicit low-rank factorization of blocks of the kernel matrix through approximation of the kernel function. The panel clustering method [18] first computes a low-rank approximation of $\mathscr{K}(x,y)$ as

$$\mathscr{K}(x,y) \approx \sum_i \kappa_i(x;y_0)\phi_i(y)$$

by Taylor series and then uses it to build the low-rank factorization.

Bebendorf and Rjasanow proposed the adaptive cross approximation [4], or ACA, as a technique to efficiently compute low-rank approximations of kernel matrices. ACA has the advantage of only requiring to evaluate rows or columns of the matrix and provides a simple yet very effective solution for smooth kernel matrix approximations. However, it can have convergence issues in some situations (see, for instance, [7]) if it cannot capture all necessary information to properly build the low-rank basis and lacks convergence guarantees.

In the realm of analytic approximations, [31] (and similarly [6, 7, 10, 30] in the Fourier space) interpolate $\mathscr{K}(x,y)$ over $\mathcal{X}\times\mathcal{Y}$ using classical interpolation methods (for instance, polynomial interpolation at Chebyshev nodes in [10]), resulting in expressions like

$$\mathscr{K}(x,y) \approx S(x,\widetilde{X})\mathscr{K}(\widetilde{X},\widetilde{Y})T(y,\widetilde{Y})^\top = \sum_k \sum_l S_k(x)\mathscr{K}(\tilde{x}_k,\tilde{y}_l)T_l(y),$$

where S and T are Lagrange interpolation basis functions. Those expressions can be further recompressed by performing a rank-revealing factorization on the node matrix $\mathscr{K}(\widetilde{X},\widetilde{Y})$, for instance, using SVD [10] or ACA [7]. Furthermore, [31] takes the SVD of a scaled $\mathscr{K}(\widetilde{X},\widetilde{Y})$ matrix to further recompress the approximation and obtain an explicit expression for u_r and v_r such that

$$\mathscr{K}(x,y) \approx \sum_s \sigma_s u_s(x) v_s(y),$$

where $\{u_s\}_s$ and $\{v_s\}_s$ are sequences of orthonormal functions in the usual L_2 scalar product.

Bebendorf [3] builds a low-rank factorization of the form

$$\mathscr{K}(x,y) = \mathscr{K}(x,\widetilde{Y})\mathscr{K}(\widetilde{X},\widetilde{Y})^{-1}\mathscr{K}(\widetilde{X},y), \tag{2}$$

where the nodes $\widetilde{X}$ and $\widetilde{Y}$ are interpolation nodes of an interpolation of $\mathscr{K}(x,y)$ built iteratively. Similarly, in their second version of the hybrid cross approximation algorithm, Börm and Grasedyck [7] propose applying ACA to the kernel matrix evaluated at interpolation nodes to obtain pivots $\widetilde{X}_i$, $\widetilde{Y}_j$ and implicitly build an approximation of the form given in (2). Both of those algorithms resemble our approach in that they compute pivots $\widetilde{X},\widetilde{Y}$ in some way and then use (2) to build the low-rank approximation. In contrast, our algorithm uses weights and has stronger accuracy guarantees. We highlight those differences in section 5.

Our method inserts itself among those low-rank kernel factorization techniques. However, with the notable exception of ACA, those methods often either rely on

analytic expressions for the kernel function (and are then limited to some specific ones) or have suboptimal complexities, i.e., greater than $\mathcal{O}(nr)$. In addition, even though we use interpolation nodes, it is worth noting that our method differs from interpolation-based algorithms as we never explicitly build the $S(x, \widetilde{X})$ and $T(y, \widetilde{Y})$ matrices containing the basis functions. We merely rely on their existence.

$\mathcal{H}$-matrices [16, 17, 15] are one way to deal with kernel matrices arising from boundary integral equations that are hierarchically block low-rank. The compression criterion (i.e., which blocks are compressed as low-rank, and which are not) leads to different methods, usually denoted as strongly admissible (only compress well-separated boxes) or weakly admissible (compress adjacent boxes as well). In the realm of strongly admissible $\mathcal{H}$-matrices, the technique of Ho and Ying [22] as well as Tyrtyshnikov [29] are of particular interest to us. They use skeletonization of the matrix to reduce storage and computation cost. In [22], they combine skeletonization and sparsification to keep compressing blocks of $\mathcal{H}$-matrices. [29] uses a somewhat nontraditional skeletonization technique to also compress hierarchical kernel matrices.

Finally, extending the framework of low-rank compression, [9] uses tensor-train compression to rewrite $\mathscr{K}(X, Y)$ as a tensor with one dimension per coordinate, i.e., $\mathscr{K}(x_1, \ldots, x_d, y_1, \ldots, y_d)$, and then compress it using the tensor-train model.

1.3. Contribution.

1.3.1. Overview of the method. In this paper, we present a new algorithm that performs this low-rank factorization at a cost proportional to $\mathcal{O}(m + n)$. The main advantages of the method are as follows:

1. The complexity of our method is $\mathcal{O}(r(m + n))$ (in terms of kernel function $\mathscr{K}$ evaluations), where r is the target rank.
2. The method is robust and accurate, irrespective of the distribution of points x and y.
3. We can prove both convergence and numerical stability of the resulting algorithm.
4. The method is very simple and relies on well-optimized BLAS3 (general matrix matrix products) and LAPACK (rank-revealing QR, LU) kernels.

Consider the problem of approximating $\mathscr{K}(x, y)$ over the mesh $X \times Y$ with $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$. Given the matrix $\mathscr{K}(X, Y)$, one possibility to build a low-rank factorization is to do a rank-revealing LU. This would lead to the selection of

$$X_{\text{piv}} \subset X, \qquad Y_{\text{piv}} \subset Y,$$

called the "pivots," and the low-rank factorization would then be given by

$$\mathscr{K}(X, Y) \approx \mathscr{K}(X, Y_{\text{piv}})\mathscr{K}(X_{\text{piv}}, Y_{\text{piv}})^{-1}\mathscr{K}(X_{\text{piv}}, Y).$$

In practice, however, this method may become inefficient as it requires assembling the matrix $\mathscr{K}(X, Y)$ first.

In this paper, we propose and analyze a new method to select the "pivots" *outside* of the sets X and Y. The key advantage is that this selection is independent of the sets X and Y, hence the reduced complexity. Let us consider the case where $\mathcal{X}, \mathcal{Y} = [-1, 1]^d$. We will keep this assumption throughout this paper. Then, within $[-1, 1]^d$, one can build tensor grids of Chebyshev points $\overline{X}, \overline{Y}$ and associated integration weights $\overline{W}_X, \overline{W}_Y$ and then consider the matrix

$$K_w = \operatorname{diag}(\overline{W}_X)^{1/2}\mathscr{K}(\overline{X}, \overline{Y}) \operatorname{diag}(\overline{W}_Y)^{1/2}.$$

Denote $r_0 = \min(|\overline{X}|, |\overline{Y}|)$. Based on interpolation properties, we will show that this matrix is closely related to the continuous kernel $\mathscr{K}(x, y)$. In particular, they share a similar spectrum. Then we select the sets $\widehat{X} \subset \overline{X}$, $\widehat{Y} \subset \overline{Y}$ by performing strong rank-revealing QRs [13] over, respectively, $K_w^\top$ and K_w (this is also called a CUR decomposition),

$$K_w P_y = Q_y R_y,$$
$$K_w^\top P_x = Q_x R_x,$$

and build $\widehat{X}$ by selecting the elements of P_x associated to the largest rows of R_x, and similarly for $\widehat{Y}$ (if they differ in size, extend the smallest). We denote the rank of this factorization $r_1 = |\widehat{X}| = |\widehat{Y}|$, and in practice we observe that $r_1 \approx r_{SVD}$, where r_{SVD} is the rank that the truncated SVD of $\mathscr{K}(X, Y)$ would provide. The resulting factorization is

$$\mathscr{K}(X, Y) \approx \mathscr{K}(X, \widehat{Y}) \mathscr{K}(\widehat{X}, \widehat{Y})^{-1} \mathscr{K}(\widehat{X}, Y). \tag{3}$$

Note that, in this process, at no point did we build any Lagrange basis function associated with $\overline{X}$ and $\overline{Y}$. We only evaluate the kernel $\mathscr{K}$ at $\overline{X} \times \overline{Y}$.

This method appears to be very efficient in selecting sets $\widehat{X}$ and $\widehat{Y}$ of minimum sizes. Indeed, instead, one could aim for a simple interpolation of $\mathscr{K}(x, y)$ over both $\mathcal{X}$ and $\mathcal{Y}$ separately. For instance, using the regular polynomial interpolation at Chebyshev nodes $\overline{X}$ and $\overline{Y}$, it would lead to a factorization of the form

$$\mathscr{K}(X, Y) \approx S(X, \overline{X}) \mathscr{K}(\overline{X}, \overline{Y}) T(Y, \overline{Y})^\top.$$

In this expression, we collect the Lagrange basis functions (each one associated to a node of $\overline{X}$) evaluated at X in the columns of $S(X, \overline{X})$, and similarly for $T(Y, \overline{Y})$. This provides a robust way of building a low-rank approximation. The rank $r_0 = \min(|\overline{X}|, |\overline{Y}|)$, however, is usually much larger than the true rank r_{SVD} and than r_1 (given a tolerance). Note that even if those factorizations can always be further recompressed to a rank $\approx r_{SVD}$, they incur a high upfront cost because of the rank $r_0 \gg r_{SVD}$. See subsection 1.3.4 for a discussion about this.

1.3.2. Distinguishing features of the method. Since there are many methods that resemble our approach, we point out its distinguishing features. The singular value decomposition (SVD) offers the optimal low-rank representation in the 2-norm. However, its complexity scales like $\mathcal{O}(n^3)$. In addition, we will show that the new approach is negligibly less accurate than the SVD in most cases.

The rank-revealing QR and LU factorization, and methods of random projections [19], have a reduced computational cost of $\mathcal{O}(n^2 r)$ but still scale quadratically with n.

Methods like ACA [4], the rank-revealing LU factorization with rook pivoting [11], and techniques that randomly sample from columns and rows of the matrix scale like $\mathcal{O}(nr)$, but they provide no accuracy guarantees. In fact, counterexamples can be found where these methods fail. In contrast, our approach relies on Chebyshev nodes, which offer strong stability and accuracy guarantees. The fact that new interpolation points, $\overline{X}$ and $\overline{Y}$, are introduced (the Chebyshev nodes) in addition to the existing points in X and Y is one of the key elements.

Analytical methods are available, like FMM, etc., but they are limited to specific kernels. Other techniques, which are more general, like Taylor expansion and

Chebyshev interpolation [10], have strong accuracy guarantees and are as general as the method presented. However, their cost is much greater; in fact, the difference in efficiency is measured directly by the reduction from r_0 to r_1 in our approach.

1.3.3. Low-rank approximation based on SVD and interpolation. Consider the kernel function $\mathscr{K}$ and its SVD [26, Theorem VI.17].

THEOREM 1 (SVD). *Suppose $\mathscr{K} : [-1,1]^d \times [-1,1]^d$ is square integrable. Then there exist two sequences of orthogonal functions $\{u_i\}_{i=1}^{\infty}$ and $\{v_i\}_{i=1}^{d}$ and a nonincreasing sequence of nonnegative real numbers $\{s_i\}_{i=1}^{\infty}$ such that*

$$\mathscr{K}(x,y) = \sum_{s=1}^{\infty} \sigma_s u_s(x) v_s(y). \tag{4}$$

As one can see, under relatively mild assumptions, any kernel function can be expanded into an SVD. Hence from any kernel function expansion we find a low-rank decomposition for the matrix $\mathscr{K}(X,Y)$ (which is *not* the same as the matrix SVD):

$$\mathscr{K}(X,Y) \approx \sum_{s=1}^{r} u_s(X)\sigma_s v_s(Y) = \begin{bmatrix} u_1(X) & \cdots & u_r(X) \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \begin{bmatrix} v_1^{\top}(Y) \\ \vdots \\ v_r^{\top}(Y) \end{bmatrix}, \tag{5}$$

where the sequence $\{s_i\}_{i=1}^{\infty}$ was truncated at an appropriate index r. As a general rule of thumb, the smoother the function $\mathscr{K}(x,y)$, the faster the decay of the σ_s's and the lower the rank.

If we use a polynomial interpolation method with Chebyshev nodes, we get a similar form:

$$\mathscr{K}(X,Y) \approx S(X,\overline{X})\, \mathscr{K}(\overline{X},\overline{Y})\, T(Y,\overline{Y})^{\top}. \tag{6}$$

The interpolation functions $S(x,\overline{X})$ and $T(y,\overline{Y})$ have strong accuracy guarantees, but the number of terms required in the expansion is $r_0 \gg r \approx r_1$. This is because Chebyshev polynomials are designed for a broad class of functions. In contrast, the SVD uses basis functions u_s and v_s that are optimal for the chosen $\mathscr{K}$.

1.3.4. Optimal interpolation methods. We will now discuss a more general problem, then derive our algorithm as a special case. Let's start with understanding the optimality of the Chebyshev interpolation. With Chebyshev interpolation, $S(x,\overline{X})$ and $T(y,\overline{Y})$ are polynomials. This is often considered one of the best (most stable and accurate) ways to interpolate smooth functions. We know that for general polynomial interpolants we have

$$f(x) - S(x,\overline{X})f(\overline{X}) = \frac{f^{(m)}(\xi)}{m!} \prod_{j=1}^{m} (x - \overline{X}_j). \tag{7}$$

If we assume that the derivative $f^{(m)}(\xi)$ is bounded, we can focus on finding interpolation points such that

$$\prod_{j=1}^{m} (x - \overline{X}_j) = x^m - r_{\overline{X}}(x)$$

is minimal, where $r_{\overline{X}}(x)$ is a degree $m-1$ polynomial. Since we are free to vary the interpolation points $\overline{X}$, then we have m parameters (the location of the interpolation

points) and m coefficients in $r_{\overline{X}}$. By varying the location of the interpolation points, we can recover any polynomial $r_{\overline{X}}$. Chebyshev points are known to solve this problem optimally. That is, they lead to an $r_{\overline{X}}$ such that $\max_x |x^m - r_{\overline{X}}(x)|$ is minimal.

Chebyshev polynomials are a very powerful tool because of their generality and simplicity of use. Despite this, we will see that this can be improved upon with relatively minimal effort. Let's consider the construction of interpolation formulas for a family of functions $\mathscr{K}(x,\lambda)$, where λ is a parameter. We would like to use the SVD, but, because of its high computational cost, we rely on the cheaper rank-revealing QR factorization (RRQR, a QR algorithm with column pivoting). RRQR solves the following optimization problem:

$$\min_{\{\lambda_s, v_s\}} \max_{\lambda} \left\| \mathscr{K}(x,\lambda) - \sum_{s=1}^{m} \mathscr{K}(x,\lambda_s) v_s(\lambda) \right\|_2, \qquad v_s(\lambda_t) = \delta_{st},$$

where the 2-norm is computed over x—in addition, RRQR produces an orthogonal basis for $\{\mathscr{K}(x,\lambda_s)\}_s$, but this is not needed in the current discussion. The vector space $\operatorname{span}\{\mathscr{K}(x,\lambda_s)\}_{s=1,\dots,m}$ is close to $\operatorname{span}\{u_s\}_{s=1,\dots,m}$ [see (4)], and the error can be bounded by σ_{m+1}.

Define $\widehat{\Lambda} = \{\lambda_1, \dots, \lambda_m\}$. From there, we identify a set of m interpolation nodes $\widehat{X}$ such that the square matrix

$$\mathscr{K}(\widehat{X}, \widehat{\Lambda}) := \begin{bmatrix} \mathscr{K}(\widehat{X}, \lambda_1) & \cdots & \mathscr{K}(\widehat{X}, \lambda_m) \end{bmatrix}$$

is as well-conditioned as possible. We now define our interpolation operator as

$$\widehat{S}(x, \widehat{X}) = \mathscr{K}(x, \widehat{\Lambda}) \mathscr{K}(\widehat{X}, \widehat{\Lambda})^{-1}.$$

By design, this operator is exact on $\mathscr{K}(x,\lambda_s)$:

$$\widehat{S}(x, \widehat{X}) \mathscr{K}(\widehat{X}, \lambda_s) = \mathscr{K}(x, \lambda_s).$$

It is also very accurate for $\mathscr{K}(x,\lambda)$ since

$$\widehat{S}(x, \widehat{X}) \mathscr{K}(\widehat{X}, \lambda) \approx \sum_{s=1}^{m} \widehat{S}(x, \widehat{X}) \mathscr{K}(\widehat{X}, \lambda_s) v_s(\lambda) = \sum_{s=1}^{m} \mathscr{K}(x, \lambda_s) v_s(\lambda) \approx \mathscr{K}(x, \lambda).$$

With Chebyshev interpolation, $S(x, \overline{X})$ is instead defined using order $m-1$ polynomial functions.

A special case that illustrates the difference between SI (skeletonized interpolation) and Chebyshev is with rank-1 kernels:

$$\mathscr{K}(x,\lambda) = u(x) v(\lambda).$$

In this case, we can pick any x_1 and λ_1 such that $\mathscr{K}(x_1, \lambda_1) \neq 0$ and define $\widehat{X} = \{x_1\}$ and

$$\widehat{S}(x, \widehat{X}) = \mathscr{K}(x, \lambda_1) \mathscr{K}(x_1, \lambda_1)^{-1},$$

$$\widehat{S}(x, \widehat{X}) \mathscr{K}(\widehat{X}, \lambda) = u(x) v(\lambda_1) \frac{1}{u(x_1) v(\lambda_1)} u(x_1) v(\lambda) = u(x) v(\lambda).$$

SI is exact using a single interpolation point x_1. An interpolation using Chebyshev polynomials would lead to errors, for any expansion order (unless u is fortuitously a polynomial).

So, one of the key differences between SI and Chebyshev interpolation is that SI uses, as the basis for its interpolation, *a set of nearly optimal functions that approximate the left singular functions of* $\mathscr{K}$, rather than generic polynomial functions.

1.3.5. Proposed method. In this paper, we use the framework from subsection 1.3.4 to build an interpolation operator for the class of functions $\mathscr{K}(x,y)$, which we view as a family of functions of x parameterized by y (and vice versa to obtain a symmetric interpolation method). The approximation (eq. (3)) can be rewritten

$$\mathscr{K}(X,Y) \approx \left[\mathscr{K}(X,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\right] \mathscr{K}(\widehat{X},\widehat{Y}) \left[\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},Y)\right],$$

and by comparing with (6), we recognize the interpolation operators:

$$\widehat{S}(x,\widehat{X}) = \mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}, \qquad \widehat{T}(y,\widehat{Y}) = \mathscr{K}(\widehat{X},y)^{\top}\mathscr{K}(\widehat{X},\widehat{Y})^{-T}.$$

These interpolation operators are nearly optimal; because of the way these operators are constructed we call the method "skeletonized interpolation." The sets $\widehat{X}$ and $\widehat{Y}$ are the minimal sets such that if we sample $\mathscr{K}$ at these points we can interpolate $\mathscr{K}$ at any other point with accuracy ϵ. In particular, $\widehat{X}$ and $\widehat{Y}$ are much smaller than their Chebyshev-interpolant counterparts $\overline{X}$ and $\overline{Y}$, and their size, r_1, is very close to r in (5). The approach we are proposing produces nearly optimal interpolation functions for our kernel instead of generic polynomial functions.

Note that none of the previous discussions explains why the proposed scheme is stable; the inverse $\mathscr{K}(\widehat{X},\widehat{\Lambda})^{-1}$ as well as $\mathscr{K}(\widehat{X},\widehat{Y})^{-1}$ in (3) could become troublesome numerically. We will explain in detail in section 3 why this is not an issue numerically, and we explore the connection with interpolation in more detail in section 4.

1.3.6. Organization of the paper. This paper is organized as follows. In section 2, we present the algorithm in detail and present some theoretical results about its convergence. In section 3, we discuss its numerical stability, and in section 4 we revisit the interpolation interpretation on a simple example. Finally, section 5 illustrates the algorithm on more complex geometries, compares its accuracy with other classical algorithms, and presents computational complexity results.

2. Skeletonized interpolation.

2.1. The algorithm. Algorithm 1 provides the high-level version of the algorithm. It consists of 3 steps:

- Build grids $\overline{X}$ and $\overline{Y}$, tensor grids of Chebyshev nodes. Over $[-1,1]$ in 1D, the $\overline{m}$ Chebyshev nodes of the first kind are defined as (see [1])

$$\bar{x}_k = \cos\left(\frac{2k-1}{2\overline{m}}\pi\right), \qquad k = 1,\dots,\overline{m}.$$

  In higher dimensions, they are defined as the tensor product of 1D grids. The number of points in every dimension should be such that

$$\sum_{k=1}^{\overline{m}}\sum_{l=1}^{\overline{n}} S_k(x)\mathscr{K}(\bar{x}_k,\bar{y}_l)T_l(y) = S(x,\overline{X})\mathscr{K}(\overline{X},\overline{Y})T(y,\overline{Y})^{\top}$$

  provides an δ uniform approximation over $[-1,1]^d \times [-1,1]^d$ of $\mathscr{K}(x,y)$. Denote

$$r_0 = \min(|\overline{X}|,|\overline{Y}|).$$

- Recompress the grid by performing a strong rank-revealing QR factorization [13] of

$$\operatorname{diag}(\overline{W}_X)^{1/2}\mathscr{K}(\overline{X},\overline{Y})\operatorname{diag}(\overline{W}_Y)^{1/2} \tag{8}$$

Algorithm 1. Skeletonized interpolation.

procedure SKELETONIZED INTERPOLATION($\mathscr{K} : [-1,1]^d \times [-1,1]^d \to \mathbb{R}$, X, Y, ϵ, δ)

Build $\overline{X}$ and $\overline{Y}$, sets of Chebyshev nodes over $[-1,1]^d$ that interpolate $\mathscr{K}$ with error δ uniformly

Build K_w as

$$K_w = \operatorname{diag}(\overline{W}_X)^{1/2} \mathscr{K}(\overline{X}, \overline{Y}) \operatorname{diag}(\overline{W}_Y)^{1/2}$$

Extract $\widehat{Y} \subseteq \overline{Y}$ by performing a strong RRQR over K_w with tolerance ϵ ;

$$K_w P_y = Q_y R_y$$

Extract $\widehat{X} \subseteq \overline{X}$ by performing a strong RRQR over $K_w^\top$ with tolerance ϵ ;

$$K_w^\top P_x = Q_x R_x$$

If the sets have different size, extends the smallest to the size of the largest.

return

$$\mathscr{K}(X,Y) \approx \mathscr{K}(X,\widehat{Y}) \mathscr{K}(\widehat{X},\widehat{Y})^{-1} \mathscr{K}(\widehat{X},Y)$$

end procedure

and its transpose, up to accuracy ϵ. This factorization is also named CUR decomposition [24, 8]. While our error estimates only hold for strong RRQR factorizations, in practice, a simple column-pivoted QR factorization based on choosing columns with large norms works as well. In the case of Chebyshev nodes of the first kind in 1D over $[-1,1]$ the integration weights are given by

$$w_k = \frac{\pi}{\overline{m}} \sqrt{1 - \bar{x}_k^2} = \frac{\pi}{\overline{m}} \sin\left(\frac{2k-1}{2\overline{m}}\pi\right).$$

The weights in d dimensions are the products of the corresponding weights in 1D, and the $\operatorname{diag}(\overline{W}_X)$ and $\operatorname{diag}(\overline{W}_Y)$ matrices are simply the diagonal matrices of the integration weights. Denote

$$r_1 = |\widehat{X}| = |\widehat{Y}|.$$

In case the sets $\widehat{X}$ and $\widehat{Y}$ output by those RRQRs are of slightly different size (which rarely occurred in our experiments), extend the smallest to have the same size as the largest.

- Given $\widehat{X}$ and $\widehat{Y}$, the low-rank approximation is given by

$$\mathscr{K}(X,\widehat{Y}) \mathscr{K}(\widehat{X},\widehat{Y})^{-1} \mathscr{K}(\widehat{X},Y)$$

of rank $r_1 \approx r_{SVD}$.

2.2. Theoretical convergence.

2.2.1. Overview. In this section, we prove that the error made during the RRQR is not too much amplified when evaluating the interpolant. We first recall the following:

1. From interpolation properties,

$$\mathscr{K}(x,y) = S(x,\overline{X})\mathscr{K}(\overline{X},\overline{Y})T(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y),$$

where T and S are small matrices (i.e., bounded by logarithmic factors in r_0) and $E_{\mathrm{INT}} = \mathcal{O}(\delta)$.
2. From the strong RRQR properties,

$$K_w = \begin{bmatrix} I \\ \widehat{S} \end{bmatrix} \hat{K}_w \begin{bmatrix} I & \widehat{T}^\top \end{bmatrix} + E_{\mathrm{QR}},$$

where $\hat{K}_w$ has a spectrum similar to that of K_w (up to a small polynomial), $\widehat{S}$ and $\widehat{T}$ are bounded by a small polynomial, and $E_{\mathrm{QR}} = \mathcal{O}(\epsilon)$.

Then, by combining those two facts and assuming $\delta < \epsilon$, one can show that

1. first, the interpolation operators are bounded,

$$\|\mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\|_2 = \mathcal{O}(p(r_0,r_1))\,, \tag{9}$$

where p is a small polynomial;
2. second, the error ϵ made in the RRQR is not too much amplified, i.e.,

$$|\mathscr{K}(x,y) - \mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y)| = \mathcal{O}(p'(r_0,r_1)\epsilon)\,, \tag{10}$$

where p' is another small polynomial.

Finally, if one assume that $\sigma_i(K_w)$ decays exponentially fast, so does ϵ, and the resulting approximation in (10) converges.

In the following, we present the main lemmas (some proofs are relocated to the appendix for brevity) leading to the above result.

2.2.2. Interpolation-related results. We first consider the interpolation itself. Consider $\overline{X}$ and $\overline{Y}$, constructed such that

$$\mathscr{K}(x,y) = S(x,\overline{X})\mathscr{K}(\overline{X},\overline{Y})T(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y).$$

LEMMA 1 (interpolation at Chebyshev nodes). *For all $x \in \mathcal{X}$ and $\overline{X}$ tensor grids of Chebyshev nodes of the first kind,*

$$\|S(x,\overline{X})\|_2 = \mathcal{O}\big(\log(|\overline{X}|)^d\big)\,,$$

where $\mathcal{X} \subset \mathbb{R}^d$. In addition, the weights, collected in the weight matrix $\mathrm{diag}(\overline{W}_X)$, *are such that*

$$\|\,\mathrm{diag}(\overline{W}_X)^{1/2}\|_2 \le \frac{\pi^{d/2}}{\sqrt{\overline{m}}} = \mathcal{O}\bigg(\frac{1}{\sqrt{\overline{m}}}\bigg)\,,$$
$$\|\,\mathrm{diag}(\overline{W}_X)^{-1/2}\|_2 \le \frac{\overline{m}}{\pi^{d/2}} = \mathcal{O}(\overline{m})\,,$$

where $\overline{m} = |\overline{X}|$.

2.2.3. Skeletonization results. We now consider the skeletonization step of the algorithm performed through the two successive rank-revealing QR factorizations.

Rank-revealing QR factorizations. Let us first recall what a rank-revealing QR factorization is. Given a matrix $A \in \mathbb{R}^{m\times n}$, one can compute a rank-revealing QR factorization [11] of the form

$$A\Pi = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix},$$

where Π is a permutation matrix, Q an orthogonal matrix, and R a triangular matrix. Both R and Q are partitioned so that $Q_1 \in \mathbb{R}^{m\times k}$ and $R_{11} \in \mathbb{R}^{k\times k}$. If $\|R_{22}\| \approx \varepsilon$, this factorization typically indicates that A has an ε-rank of k. The converse, however, is not necessarily true [11] in general.

From there, one can also write

$$A\Pi = Q_1 R_{11} \begin{bmatrix} I & R_{11}^{-1} R_{12} \end{bmatrix} + E = A_1 \begin{bmatrix} I & T \end{bmatrix} + E,$$

where T is the *interpolation* operator, A_1 a set of k columns of A, and E the approximation error. This approximation can be achieved by a simple column-pivoted QR algorithm [11]. This algorithm, however, is not guaranteed to always work (i.e., even if A has rapidly decaying singular values, this rank-revealing factorization may fail to exhibit it).

A *strong* rank-revealing QR, however, has more properties. It has been proven [13, 8] that one can compute in $\mathcal{O}(mn^2)$ a rank-revealing QR factorization that guarantees

$$\sigma_i(A_1) \geq \frac{\sigma_i(A)}{q_1(n,k)}\ ,\ \ \sigma_j(E) \leq \sigma_{k+j}(A) q_1(n,k),\ \text{and}\ \ \|T\|_F \leq q_2(n,k), \tag{11}$$

where q_1 and q_2 are two small polynomials (with fixed constants and degrees). The existence of this factorization is a crucial part of our argument. Using the interlacing property of singular values [11], this implies that we now have both lower and upper bounds on the singular values of A_1:

$$\frac{\sigma_i(A)}{q_1(n,k)} \leq \sigma_i(A_1) \leq \sigma_i(A). \tag{12}$$

From (11) we can directly relate the error E and σ_{k+1} from

$$\|E\|_2 = \sigma_1(E) \leq \sigma_{k+1}(A) q_1(n,k). \tag{13}$$

Finally, given a matrix A, one can apply the above result to both its rows and columns, leading to the factorization

$$\Pi_r^\top A \Pi_c = \begin{bmatrix} I \\ T_r \end{bmatrix} A_{rc} \begin{bmatrix} I & T_c \end{bmatrix} + E$$

with the same properties as detailed above.

Skeletonized interpolation. We can now apply this result to the K_w and $\hat{K}_w$ matrices.

LEMMA 2 (CUR decomposition of K_w). *The partition $\overline{X} = \widehat{X} \cup \check{X}$, $\overline{Y} = \widehat{Y} \cup \check{Y}$ of Algorithm 1 is such that there exist $\check{S}$, $\check{T}$, $E_{QR}(\overline{X}, \overline{Y})$ matrices and a slowly growing polynomial $p(r_0, r_1)$ such that*

$$K_w = \begin{bmatrix} I \\ \check{S} \end{bmatrix} \hat{K}_w \begin{bmatrix} I & \check{T}^\top \end{bmatrix} + E_{QR}(\overline{X}, \overline{Y})$$

and where

$$\epsilon = \|E_{QR}(\overline{X},\overline{Y})\|_2 \le p(r_0,r_1)\sigma_{r_1+1}(K_w),$$
$$\|\breve{S}\|_2 \le p(r_0,r_1),$$
$$\|\breve{T}\|_2 \le p(r_0,r_1).$$

Finally, we have

$$\|\hat{K}_w^{-1}\|_2 \le \frac{p(r_0,r_1)^2}{\epsilon}.$$

Proof. The first three results are direct applications of [8, Theorem 3 and Remark 5], as explained in the previous paragraph. The last result follows from the properties of the strong rank-revealing QR:

$$\|\hat{K}_w^{-1}\|_2 = \frac{1}{\sigma_{r_1}(\hat{K}_w)} \le \frac{p(r_0,r_1)}{\sigma_{r_1}(K_w)} \le \frac{p(r_0,r_1)}{\sigma_{r_1+1}(K_w)} \le \frac{p(r_0,r_1)^2}{\epsilon}.$$

The first inequality follows from $\sigma_{r_1}(K_w) \le \sigma_{r_1}(\hat{K}_w)p(r_0,r_1)$ (eq. (12)), the second from $\sigma_{r_1}(K_w) \ge \sigma_{r_1+1}(K_w)$ (by definition of singular values), and the last from $\sigma_{r_1+1}(K_w)^{-1} \le p(r_0,r_1)\epsilon^{-1}$ (eq. (13)). ∎

Finally, a less obvious result.

LEMMA 3. *There exists a polynomial $q(r_0,r_1)$ such that for any $x \in \mathcal{X}, y \in \mathcal{Y}$,*

$$\|\mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\|_2 = \mathcal{O}(q(r_0,r_1)),$$
$$\|\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y)\|_2 = \mathcal{O}(q(r_0,r_1)).$$

We provide the proof in the appendix; the key ingredient is simply that $\|\hat{K}_w^{-1}\|_2 \le p(r_0,r_1)^2\epsilon^{-1}$ from the RRQR properties; hence $\hat{K}_w$ is ill-conditioned, but not arbitrarily. Its condition number grows like ϵ^{-1}. Then, when multiplied by quantities like ϵ or $\delta \ll \epsilon$, the factors cancel out and the resulting product can be bounded.

2.2.4. Link between the node matrix and the continuous SVD. In this section, we link the continuous SVD and the spectrum (singular values) of the matrix $\operatorname{diag}(\overline{W}_X)^{1/2}K_w\operatorname{diag}(\overline{W}_Y)^{1/2}$. This justifies the use of the weights.

For the sake of simplicity, consider the case where interpolation is performed at *Gauss–Legendre* nodes $\overline{X},\overline{Y}$ with the corresponding integration weights $\overline{W}_X,\overline{W}_Y$. (A more complete explanation can be found in [31].)

Take the classical discrete SVD of K_w,

$$K_w = \overline{U}\,\overline{\Sigma}\,\overline{V}^\top.$$

We then have

$$\mathscr{K}(x,y) = \underbrace{S_w(x,\overline{X})\overline{U}\,\overline{\Sigma}\,\overline{V}^\top T_w(y,\overline{Y})^\top}_{=\overline{\mathscr{K}}(x,y)} + E_{\mathrm{INT}}(x,y).$$

Then denote the sets of new basis functions

$$\overline{u}(x) = S_w(x,\overline{X})\overline{U}, \qquad \overline{v}(y) = T_w(y,\overline{Y})\overline{V}.$$

The key is to note that those functions are orthonormal. Namely, for $\overline{u}$,

$$\begin{aligned}\int_{\mathcal{X}} \overline{u}_i(x)\overline{u}_j(x)\mathrm{d}x &= \sum_{k=1}^{r_0} \overline{w}_k \overline{u}_i(\overline{x}_k)\overline{u}_j(\overline{x}_k) \\ &= \sum_{k=1}^{r_0} \overline{w}_k \left(\sum_{l=1}^{r_0} \overline{w}_l^{-1/2} S_l(\overline{x}_k)\overline{U}_{li}\right)\left(\sum_{l=1}^{r_0} \overline{w}_l^{-1/2} S_l(\overline{x}_k)\overline{U}_{lj}\right) \\ &= \sum_{k=1}^{r_0} \overline{w}_k \left(\sum_{l=1}^{r_0} \delta_{kl}\overline{w}_l^{-1/2}\overline{U}_{li}\right)\left(\sum_{l=1}^{r_0} \delta_{kl}\overline{w}_l^{-1/2}\overline{U}_{lj}\right) \\ &= \sum_{k=1}^{r_0} \overline{w}_k \overline{w}_k^{-1/2}\overline{U}_{ki}\overline{w}_k^{-1/2}\overline{U}_{kj} = \sum_{k=1}^{r_0} \overline{U}_{ki}\overline{U}_{kj} = \delta_{ij}.\end{aligned}$$

The same result holds for $\overline{v}$. This follows from the fact that a Gauss–Legendre quadrature rule with n points can exactly integrate polynomials up to degree $2n-1$. This shows that we are implicitly building a factorization

$$\mathscr{K}(x,y) = \sum_{s=1}^{\infty} \sigma_s u_s(x) v_s(y) = \underbrace{\sum_{s=1}^{r_0} \sigma_s(K_w)\overline{u}_s(x)\overline{v}_s(y)}_{=\overline{\mathscr{K}}(x,y)} + E_{\mathrm{INT}}(x,y), \tag{14}$$

where the approximation error is bounded by the interpolation error E_{INT} and where the sets of basis functions are orthogonal.

Assume now that the kernel $\mathscr{K}$ is square-integrable over $[-1,1]^d \times [-1,1]^d$. This is called a Hilbert–Schmidt kernel [27, Lemma 8.20]. This implies that the associated linear operator is compact [27, Theorem 8.83]. $\overline{\mathscr{K}}$ is compact as well since it is finite rank [27, Theorem 8.80]. Given the fact that $|E_{\mathrm{INT}}(x,y)| \leq \delta$ for all x, y, $\|\mathscr{K} - \overline{\mathscr{K}}\|_{L_2} \leq C\delta$ for some C, and hence, by compactness of both operators [14, Corollary 2.2.14],

$$|\sigma_i - \sigma_i(\overline{\mathscr{K}})| \leq C\delta$$

for some $C > 0$, then, from the above discussion, we clearly have $\sigma_i(K_w) = \sigma_i(\overline{\mathscr{K}}) + \mathcal{O}(\delta)$ and hence

$$\sigma_i(K_w) = \sigma_i + \mathcal{O}(\delta)\,.$$

This result only formally holds for Gauss–Legendre nodes and weights. However, this motivates the use of integration weights in the case of Chebyshev as well.

2.2.5. Convergence of the skeletonized interpolation. We now present the main result of this paper.

THEOREM 2 (convergence of skeletonized interpolation). *If $\widehat{X}$ and $\widehat{Y}$ are constructed following Algorithm* 1*, then there exists a polynomial $r(r_0, r_1)$ such that for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$,*

$$|\mathscr{K}(x,y) - \mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y)| = \mathcal{O}(\epsilon\, r(r_0,r_1))\,.$$

The key here is that the error incurred during the CUR decomposition, ϵ, is amplified by, at most, a polynomial of r_0 and r_1. Hence, Theorem 2 indicates that if the spectrum decays fast enough (i.e., if $\epsilon \to 0$ when $r_0, r_1 \to \infty$ faster than $r(r_0, r_1)$ grows), the proposed approximation should converge to the true value of $\mathscr{K}(x,y)$.

What is left is then simply linking ϵ, r_0, and r_1. We have, from the CUR properties,

$$\epsilon \leq p(r_0, r_1)\sigma_{r_1+1}(K_w),$$

which implies

$$|\mathscr{K}(x,y) - \mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y)| = \mathcal{O}(\sigma_{r_1+1}(K_w)r'(r_0,r_1))\,.$$

Then, following the discussion from subsection 2.2.4, we expect

$$\sigma_i(K_w) = \sigma_i + \mathcal{O}(\delta)\,.$$

Hence, if $\mathscr{K}$ has rapidly decaying singular values, so does K_w. Assuming the singular values of K_w decay exponentially fast, i.e.,

$$\log \sigma_k(K_w) \approx \text{poly}(k),$$

we find

$$|\mathscr{K}(x,y) - \mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y)| \to 0$$

as $r_0, r_1 \to \infty$, or, alternatively, as $\epsilon \to 0$.

3. Numerical stability.

3.1. The problem. The previous section indicates that, at least theoretically, we can expect convergence as $\epsilon \to 0$. However, the factorization

$$\mathscr{K}(X,Y) \approx \mathscr{K}(X,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},Y) \tag{15}$$

seems to be numerically challenging to compute. Indeed, as we showed in the previous section, we can only really expect at best $\|\hat{K}_w^{-1}\|_2 = \mathcal{O}(\epsilon^{-1})$, which indicates that, roughly,

$$\kappa(\mathscr{K}(\widehat{X},\widehat{Y})) = \mathcal{O}\left(\frac{1}{\epsilon}\right),$$

i.e., the condition number grows with the desired accuracy, and convergence beyond a certain threshold (like 10^{-8} in double precision) seems impossible. Hence, we can reasonably be worried about the numerical accuracy of computing (15) even with a stable algorithm.

Note that this is not a pessimistic upper bound; by construction, $\hat{K}_w$ really is ill-conditioned, and experiments show that solving linear systems $\hat{K}_w x = b$ with random right-hand sides is numerically challenging and leads to errors of the order ϵ^{-1}.

3.2. Error analysis. Consider (15), and for simplicity let

$$K_x = \mathscr{K}(X,\widehat{Y}), \quad K = \mathscr{K}(\widehat{X},\widehat{Y}), \quad K_y = \mathscr{K}(\widehat{X},Y).$$

In this section, our goal is to show why one can expect this formula to be accurately computed if one uses backward stable algorithms. As proved in section 2, we have the following bounds on the interpolation operators:

$$\|K_x K^{-1}\|_2 \leq p(r_0,r_1),$$
$$\|K^{-1}K_y\|_2 \leq p(r_0,r_1)$$

for some polynomial p. The key is that there is no ϵ^{-1} in this expression. Those bounds essentially follow from the guarantees provided by the strong rank-revealing QR algorithm.

Now, let's compute the derivative of $K_x K^{-1} K_y$ with respect to K_x, K, and K_y [25]:

$$\begin{aligned}\partial(K_x K^{-1} K_y) &= (\partial K_x) K^{-1} K_y + K_x(\partial(K^{-1}))K_y + K_x K^{-1}(\partial K_y) \\ &= (\partial K_x) K^{-1} K_y - K_x K^{-1}(\partial K) K^{-1} K_y + K_x K^{-1}(\partial K_y).\end{aligned}$$

Then consider perturbing K_x, K, K_y by ε (assume all matrices are of order $\mathcal{O}(1)$ for the sake of simplicity), i.e., let δK_x, δK_y, and δK be perturbations of K_x, K_y, and K, respectively, with

$$\|\delta K_x\| = \mathcal{O}(\varepsilon)\,, \|\delta K_y\| = \mathcal{O}(\varepsilon)\,, \|\delta K\| = \mathcal{O}(\varepsilon)\,.$$

Then, using the above derivative as a first order approximation, we can write

$$\begin{aligned}&\|K_x K^{-1} K - (K_x + \delta K_x)(K + \delta K)^{-1}(K_y + \delta K_y)\| \\ &\quad \le \|\delta K_x\| \|K^{-1} K_y\| + \|K_x K^{-1}\| \|\delta K\| \|K^{-1} K_y\| + \|K_x K^{-1}\| \|\delta K_y\| + \mathcal{O}\big(\varepsilon^2\big) \\ &\quad \le 2\epsilon p(r_0, r_1) + \epsilon p(r_0, r_1)^2 + \mathcal{O}\big(\varepsilon^2\big) \\ &\quad = \varepsilon(2p(r_0, r_1) + p(r_0, r_1)^2) + \mathcal{O}\big(\varepsilon^2\big)\,.\end{aligned}$$

We see that the computed result is independent of the condition number of $K = \mathscr{K}(\widehat{X}, \widehat{Y})$ and depends on $p(r_0, r_1)$ only.

Assume now that we are using backward stable algorithms in our calculations [20]. We then know that the computed result is the result of an exact computation where the inputs have been perturbed by ε. The above result indicates that the numerical result (with roundoff errors) can be expected to be accurate up to ε times a small polynomial, hence stable.

4. Skeletonized interpolation as a new interpolation rule. As indicated in the introduction, one can rewrite

$$\begin{aligned}\mathscr{K}(x, y) &\approx \mathscr{K}(x, \widehat{Y}) \mathscr{K}(\widehat{X}, \widehat{Y})^{-1} \mathscr{K}(\widehat{X}, y) \\ &= \left[\mathscr{K}(x, \widehat{Y}) \mathscr{K}(\widehat{X}, \widehat{Y})^{-1}\right] \mathscr{K}(\widehat{X}, \widehat{Y}) \left[\mathscr{K}(\widehat{X}, \widehat{Y})^{-1} \mathscr{K}(\widehat{X}, y)\right] \\ &= \widehat{S}(x, \widehat{X}) \mathscr{K}(\widehat{X}, \widehat{Y}) \widehat{T}(y, \widehat{Y})^{\top},\end{aligned}$$

where we recognize two new "cross-interpolation" (because they are built by considering both the $\mathcal{X}$ and $\mathcal{Y}$ space) operators $\widehat{S}(x, \widehat{X}) = \mathscr{K}(x, \widehat{Y}) \mathscr{K}(\widehat{X}, \widehat{Y})^{-1}$ and $\widehat{T}(y, \widehat{Y}) = \mathscr{K}(\widehat{X}, y)^{\top} \mathscr{K}(\widehat{X}, \widehat{Y})^{-\top}$. In this notation, each column of $\widehat{S}(x, \widehat{X})$ and $\widehat{T}(y, \widehat{Y})$ is a Lagrange function associated to the corresponding node in $\widehat{X}$ or $\widehat{Y}$ and evaluated at x or y, respectively.

This interpretation is interesting as it allows one to "decouple" x and y and analyze them independently. In particular, one can look at the quality of the interpolation of the basis functions $u_k(x)$ and $v_k(y)$ using $\widehat{S}$ and $\widehat{T}$. Indeed, if this is accurate, it is

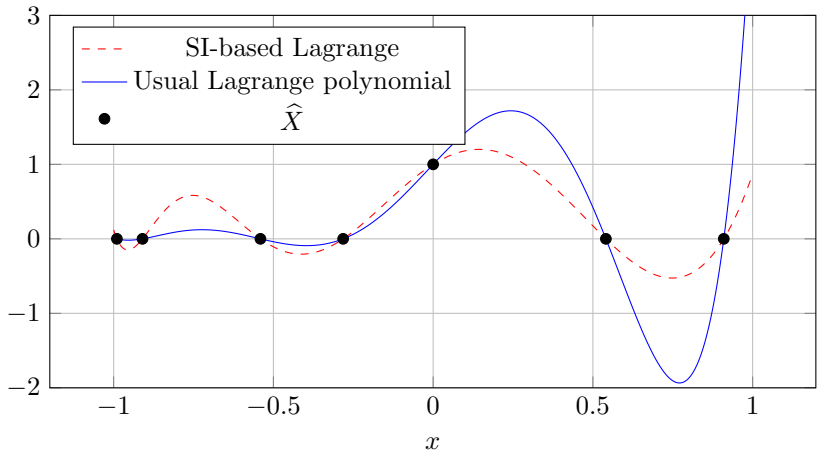


FIG. 1. *The 4th Lagrange basis function. We see that the Chebyshev–SI-based Lagrange basis function is more stable than the usual polynomial going through the same interpolation nodes.*

easy to see that the final factorization is accurate. Indeed,

$$\begin{aligned}
\mathscr{K}(x,y) &\approx \sum_{k=1}^{r} \sigma_k u_k(x) v_k(y) \\
&\approx \sum_{k=1}^{r} \sigma_k (\widehat{S}(x,\widehat{X}) u_k(\widehat{X}))(\widehat{T}(y,\widehat{Y}) v_k(\widehat{Y}))^\top \\
&= \widehat{S}(x,\widehat{X}) \left(\sum_{k=1}^{r} \sigma_k u_k(\widehat{X}) v_k(\widehat{Y})^\top \right) \widehat{T}(y,\widehat{Y})^\top \\
&\approx \widehat{S}(x,\widehat{X}) \mathscr{K}(\widehat{X},\widehat{Y}) \widehat{T}(y,\widehat{Y})^\top \\
&\approx \mathscr{K}(x,\widehat{Y}) \mathscr{K}(\widehat{X},\widehat{Y})^{-1} \mathscr{K}(\widehat{X},y).
\end{aligned}$$

To illustrate this, let us consider a simple 1D example. Let $x, y \in [-1,1]$ and consider

$$\mathscr{K}(x,y) = \frac{1}{4+x-y}.$$

Then approximate this function up to $\epsilon = 10^{-10}$ and obtain a factorization of rank r.

Figure 1 illustrates the 4th Lagrange basis function in x, i.e., $\widehat{S}(x,\widehat{X})_4$, and the classical Lagrange polynomial basis function associated with the same set $\widehat{X}$. We see that they are both 1 at $\widehat{X}_4$ and 0 at the other points. However, $\widehat{S}(x,\widehat{X})_4$ is much smaller and more stable than its polynomial counterpart. In the case of polynomial interpolation at equispaced nodes, the growth of the Lagrange basis function (or, equivalently, of the Lebesgue constant) is the reason for the inaccuracy and instability.

Figure 2 shows the effect of interpolating $u_r(x)$ using $\widehat{S}(x,\widehat{X})$ as well as using the usual polynomial interpolation at the nodes $\widehat{X}$. We see that $\widehat{S}(x,\widehat{X})$ interpolates very well $u_r(x)$, showing indeed that we implicitly build an accurate interpolant, even on the last (least smooth) eigenfunctions. On the other hand, the usual polynomial interpolation fails to capture any feature of u_r. Note that we could have reached a similar accuracy using interpolation at Chebyshev nodes but only by using many more interpolation nodes.

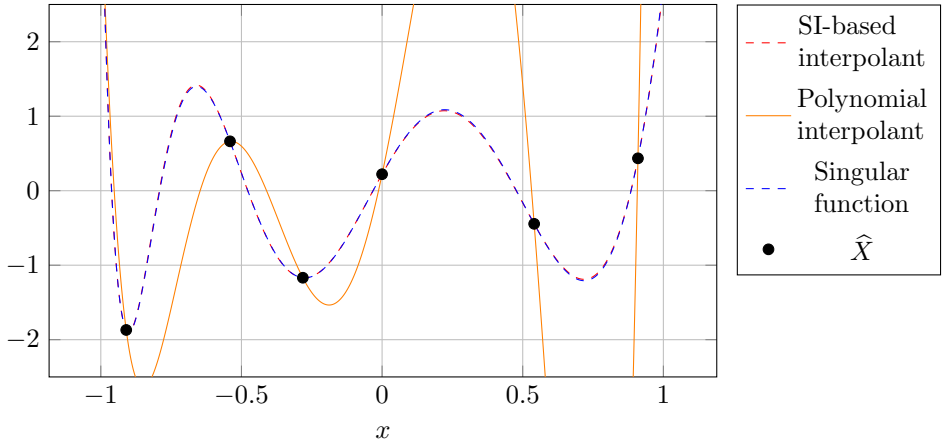


FIG. 2. *Interpolation of the last (and least smooth) eigenfunction. We see that the Chebyshev–SI-based interpolant is much more accurate than the polynomial interpolant going through the same interpolation nodes.*

Finally, Figure 3 shows how well we approximate the various r eigenfunctions. As one can see, interpolation is very accurate on $u_1(x)$, but the error grows for less smooth eigenfunctions. The growth is roughly similar to the growth of $\frac{\epsilon}{\sigma_i}$. Notice how this is just enough so that the resulting factorization is accurate:

$$\begin{aligned}
\widehat{S}(x,\widehat{X})\mathscr{K}(\widehat{X},y) &= \sum_{s=1}^{r} \sigma_s \widehat{S}(x,\widehat{X}) u_s(\widehat{X}) v_s(y) + \mathcal{O}(\epsilon) \\
&= \sum_{s=1}^{r} \sigma_s \left(u_s(x) + \mathcal{O}\left(\frac{\epsilon}{\sigma_s}\right)\right) v_s(y) + \mathcal{O}(\epsilon) \\
&= \sum_{s=1}^{r} \sigma_s u_s(x) v_s(y) + \sum_{s=1}^{r} \mathcal{O}(\epsilon)\, v_s(y) + \mathcal{O}(\epsilon) \\
&= \mathscr{K}(x,y) + \mathcal{O}(\epsilon)\,.
\end{aligned}$$

It is also consistent with the analysis of section 3. This illustrates how the algorithm works: it builds an interpolation scheme that allows for interpolating the various eigenfunctions of $\mathscr{K}$ with just enough accuracy so that the resulting interpolation is accurate up to the desired accuracy.

5. Numerical experiments. In this section we present some numerical experiments on various geometries. We study the quality (how far r_1 is from the optimal SVD-rank r and how accurate the approximation is) of the algorithm in subsections 5.1 and 5.2. We illustrate in subsection 5.3 the improved guarantees of RRQR and justify the use of weights in subsection 5.4. Finally, subsection 5.5 studies the algorithm's computational complexity.

The experiments are done using Julia [5], and the code is sequential. For the strong rank-revealing QR algorithm, we use the `LowRankApprox.jl` Julia package [21]. The code can be downloaded from https://stanford.edu/~lcambier/papers.html.

5.1. Simple geometries. We begin this section with an elementary problem, as depicted in Figure 4(b). In this problem, we consider the usual kernel $\mathscr{K}(x,y) =$

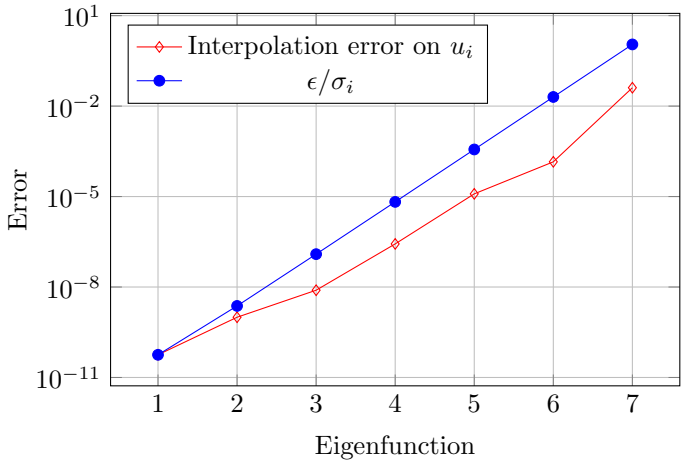


FIG. 3. *Interpolation error on the various eigenfunctions. The error grows just slowly enough with the eigenfunctions so that the overall interpolant is accurate up to the desired accuracy.*

$\|x-y\|_2^{-1}$, where $x, y \in \mathbb{R}^2$. $\mathcal{X}$ and $\mathcal{Y}$ are two squares with sides of length 1, centered at $(0.5, 0.5)$ and $(2.5, 2.5)$, respectively. Finally, X and Y are two uniform meshes of 50×50 mesh points each, i.e., $n = 2500$.

We pick the Chebyshev grids $\overline{X}$ and $\overline{Y}$ using a heuristic based on the target accuracy ε. Namely, we pick the number of Chebyshev nodes in each dimension (i.e., x_1, x_2, y_1, and y_2) independently (by using the midpoint of $\mathcal{X}$ and $\mathcal{Y}$ as reference), such that the interpolation error is approximately less than $\varepsilon^{3/4}$. This value is heuristic, but performs well for those geometries. Other techniques are possible. This choice is based in part on the observation that the algorithm is accurate even when $\delta > \varepsilon$, i.e., when the Chebyshev interpolation is *less accurate* than the actual final low-rank approximation through skeletonized interpolation.

Consider then Figure 4(a). The r_0 line indicates the rank ($r_0 = \min(|\overline{X}|, |\overline{Y}|)$) of the low-rank expansion through interpolation. The r_1 line corresponds to the rank obtained after the RRQR over $\mathcal{K}(\overline{X}, \overline{Y})$ and its transpose; i.e., it is the rank of the final approximation

$$\mathcal{K}(X, Y) \approx \mathcal{K}(X, \widehat{Y})\mathcal{K}(\widehat{X}, \widehat{Y})^{-1}\mathcal{K}(\widehat{X}, Y).$$

Finally, "SVD rank" is the rank one would obtain by truncating the SVD of

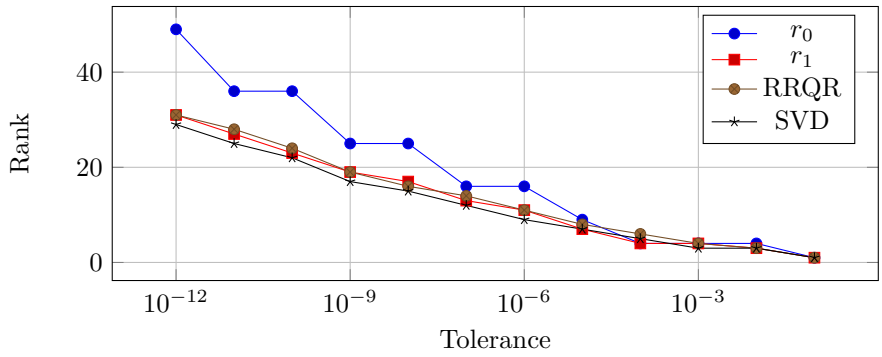
$$\mathcal{K}(X, Y) = USV^{\top}$$

at the appropriate singular value, so as to ensure

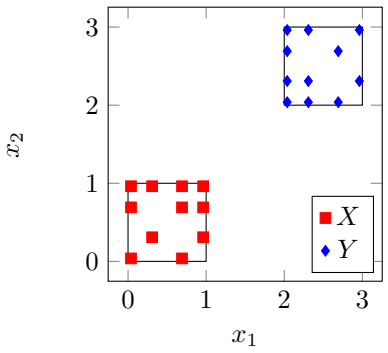
$$\|\mathcal{K}(X, Y) - \tilde{\mathcal{K}}(X, Y)\|_F \approx \epsilon \|\mathcal{K}(X, Y)\|_F.$$

Similarly, "RRQR" is the rank a rank-revealing QR on $\mathcal{K}(X, Y)$ would obtain. This is usually slightly suboptimal compared to the SVD. Those two values are there to illustrate that r_1 is close to the optimal value.

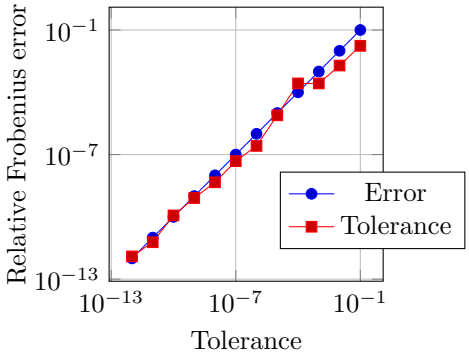
The conclusion regarding Figure 4(a) is that skeletonized interpolation is nearly optimal in terms of rank. While the rank obtained by the interpolation is clearly far from optimal, the RRQR over $\mathcal{K}(\overline{X}, \overline{Y})$ allows us to find subsets $\widehat{X} \subset \overline{X}$ and



(a) Ranks as a function of the desired accuracy.



(b) The geometry used, and the resulting choice of $\widehat{X}$, $\widehat{Y}$ for a tolerance of 10^{-6}.



(c) Relative Frobenius-norm error as a function of the desired accuracy.

FIG. 4. *Results for the 2D-squares example. The rank r_0 before compression is significantly reduced to r_1, very close to the true SVD- or RRQR-rank.*

$\widehat{Y} \subset \overline{Y}$ that are enough to represent $\mathscr{K}(X,Y)$ well, and the final rank r_1 is nearly optimal compared to the SVD-rank r. We also see that the rank of a blind RRQR over $\mathscr{K}(X,Y)$ is higher than the SVD-rank and usually closer—if not identical—to r_1.

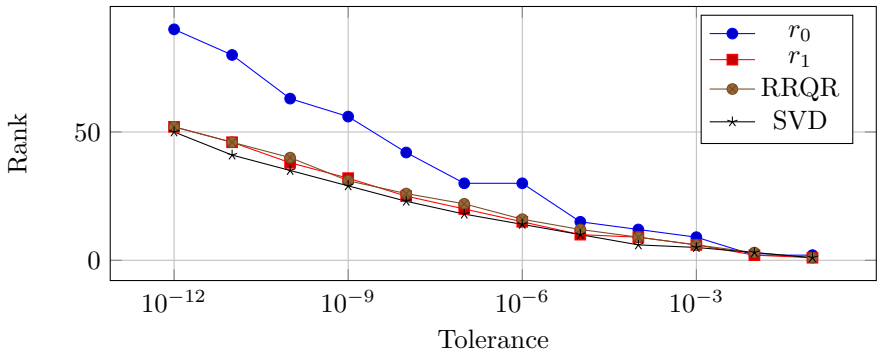
We want to reemphasize that, in practice, *the error of the sets $\overline{X}$, $\overline{Y}$—i.e., the error of the polynomial interpolation based on $\overline{X} \times \overline{Y}$—can be larger than the required tolerance.* If they are large enough, the compressed sets $\widehat{X}$, $\widehat{Y}$ will contain enough information so as to properly interpolate $\mathscr{K}$, and the final error will be smaller than the required tolerance. This is important, as the size of the Chebyshev grid for a given tolerance can be fairly large (as indicated in the introduction, and one of the main motivations of this work), even though the final rank is small.

As a sanity check, Figure 4(c) gives the relative error measured in the Frobenius norm

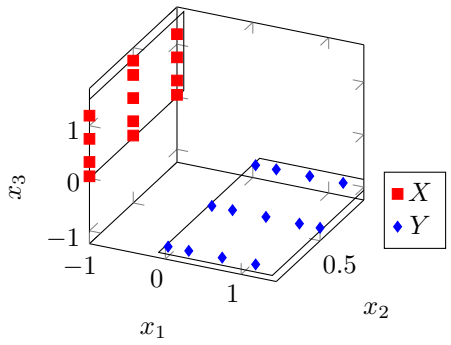
$$\frac{\|\mathscr{K}(X,Y) - \mathscr{K}(X,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},Y)\|_F}{\|\mathscr{K}(X,Y)\|_F}$$

between $\mathscr{K}(X,Y)$ and its interpolation as a function of the tolerance ϵ.[1] We see that both lines are almost next to each other, meaning our approximation indeed reaches

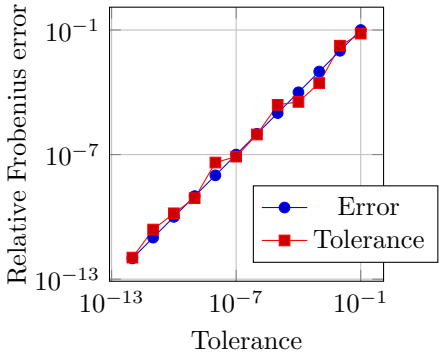
[1] Choosing the Frobenius norm is not critical—very similar results are obtained in the 2-norm, for instance.



(a) Ranks as a function of the desired accuracy.



(b) The geometry used, and the resulting choice of $\widehat{X}$, $\widehat{Y}$ for a tolerance of 10^{-6}.



(c) Relative Frobenius-norm error as a function of the desired accuracy.

FIG. 5. *Results for the perpendicular plates example. The rank r_0 before compression is significantly reduced to r_1, very close to the true SVD- or RRQR-rank.*

the required tolerance. This is important as it means that one can effectively *control* the accuracy.

Finally, Figure 4(b) also shows the resulting $\widehat{X}$ and $\widehat{Y}$. It is interesting to notice how the selected points cluster near the close corners, as one could expect since this is the area where the kernel is the least smooth.

We then consider results for the same Laplacian kernel $\mathcal{K}(x,y) = \|x-y\|_2^{-1}$ between two plates in 3D (Figure 5(b)). We observe overall very similar results as for the previous case in Figure 5(a), where the initial rank r_0 is significantly decreased to r_1 while keeping the resulting accuracy close to the required tolerance, as Figure 5(c) shows. Finally, one can see in Figure 5(b) the selected Chebyshev nodes. They again cluster in the areas where smoothness is the worst, i.e., at the closes edges of the plates.

5.2. Comparison with ACA and random sampling. We then compare our method with other standard algorithms for kernel matrix factorization. In particular, we compare it with ACA [4] and "random CUR," where one selects, at random, pivots $\widetilde{X}$ and $\widetilde{Y}$ and builds a factorization

$$\mathcal{K}(X,Y) \approx \mathcal{K}(X,\widetilde{Y})\mathcal{K}(\widetilde{X},\widetilde{Y})^{-1}\mathcal{K}(\widetilde{X},Y)$$

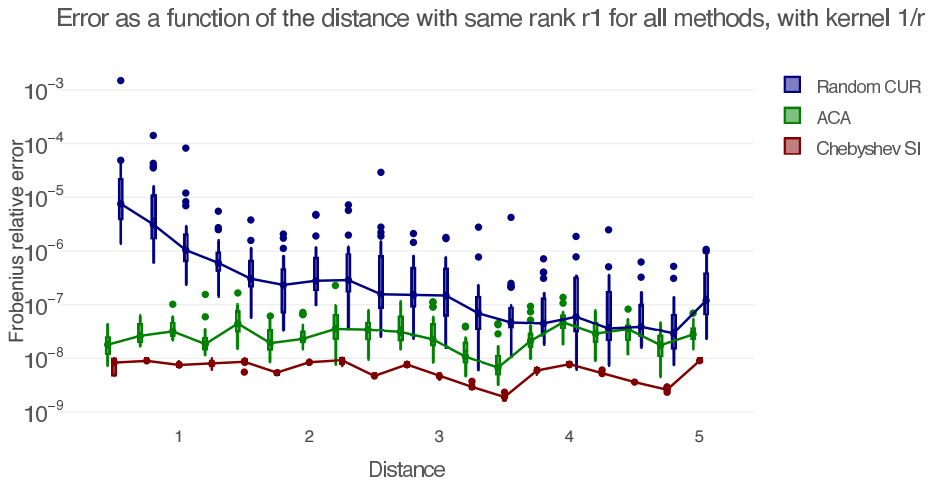


FIG. 6. *Comparison between different algorithms: Chebyshev-based SI, ACA, and purely random CUR decomposition. We consider two* 2*D squares of sides* 1 *with a variable distance from each other; for each distance, we run Chebyshev-based SI and find the smallest sets* $\overline{X}, \overline{Y}$ *of rank* r_0 *leading to a factorization using* $\widehat{X}, \widehat{Y}$ *of sizes* r_1 *with relative error at most* 10^{-8}*. Then* r_1 *is used as an a priori rank for ACA and random CUR. We randomize the experiments by subsampling* 500 *points from a large* 100×100 *points grid in each square.*

based on those. As we are interested in comparing the *quality* of the resulting sets of pivots for a given rank, we compare those algorithms for sets X and Y with variable distance between each other and for a fixed tolerance ($\epsilon = 10^{-8}$) and kernel ($\mathscr{K}(x, y) = \|x - y\|_2^{-1}$). The geometry is two unit-length squares side by side with a variable distance between their closest edges.

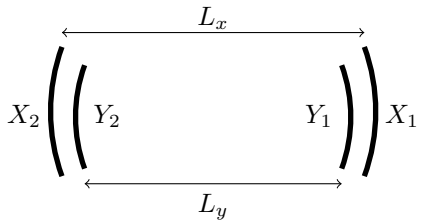
The comparison is done in the following way:

1. Given r_1, build the ACA factorization of rank r_1 and compute its relative error in Frobenius norm with $\mathscr{K}(X, Y)$.
2. Given r_1, build the random CUR factorization by sampling uniformly at random r_1 points from X and Y to build $\widetilde{X}$, $\widetilde{Y}$. Then compute its relative error with $\mathscr{K}(X, Y)$.

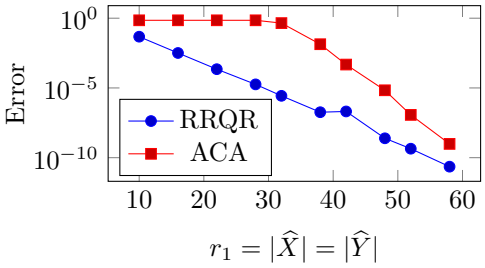
We then do the same for sets of varying distance, and for a given distance, we repeat the experiment 25 times by building X and Y at random within the two squares. This allows us to study the variance of the error and to collect some statistics.

Figure 6 gives the resulting errors in relative Frobenius norm for the 3 algorithms using box-plots of the errors to show distributions. The rectangular boxes represent the distributions from the 25% to the 75% quantiles, with the median in the center. The thinner lines represent the complete distribution, except for outliers, depicted using large dots. We observe that the ($\widehat{X}$, $\widehat{Y}$) sets based on Chebyshev–SI are, *for a common size* r_1, *more accurate* than the random or ACA sets. In addition, by design, they lead to more stable factorizations (as they have very small variance in terms of accuracy), while ACA, for instance, has a higher variance. We also see, as one may expect, that while ACA is still fairly stable even when the clusters get close, random CUR starts having higher and higher variance. This is understandable as the kernel gets less and less smooth as the clusters get close.

Finally, we ran the same experiments with several other kernels (r^{-2}, r^{-3}, $\log(r)$, $\exp(-r)$, $\exp(-r^2)$) and observed quantitatively very similar results.



(a) The geometry. $L_y = 0.9L_x$, each domain X_i, Y_i has 50 points uniformly distributed (hence, the weights are uniform) on an arc of angle $\pi/4$. $\overline{X} = X$ and $\overline{Y} = Y$, and $\widehat{X}$, $\widehat{Y}$ are r_1 points subsampled from $\overline{X}, \overline{Y}$ using Algorithm 1.



(b) Relative Frobenius error over $X \times Y$ using both RRQR and ACA to select $\widehat{X}, \widehat{Y}$ of size r_1 from $\overline{X}$, $\overline{Y}$ using Algorithm 1.

FIG. 7. *Failure of ACA. The geometry is such that the coupling between X_1/Y_1 and X_2/Y_2 is much stronger than between X_1/Y_2 and X_2/Y_1. This leads to ACA not selecting pivots properly. RRQR, on the other hand, has no issue and converges steadily.*

5.3. Stability guarantees provided by RRQR. In Algorithm 1, in principle, any rank-revealing factorization providing pivots could be used. In particular, ACA itself could be used. In this case, this is the HCAII (without the weights) algorithm as described in [7]. However, ACA is only a heuristic: unlike strong rank-revealing factorizations, it can't always reveal the rank. In particular, it may have issues when some parts of X and Y have strong interactions while others are weakly coupled. To highlight this, consider the following example. It can be extended to many other situations.

Let us use the rapidly decaying kernel

$$\mathscr{K}(x, y) = \frac{1}{\|x - y\|_2^3}$$

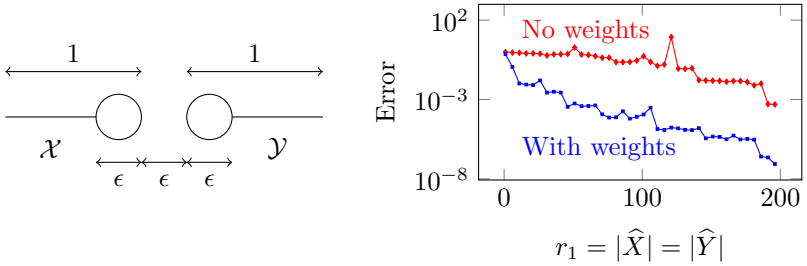
and the situation depicted in Figure 7 with $X = \begin{bmatrix} X_1 & X_2 \end{bmatrix}$ and $Y = \begin{bmatrix} Y_1 & Y_2 \end{bmatrix}$. We note that, formally, X and Y are not well separated.

Since $\mathscr{K}$ is rapidly decaying and X_1/Y_2 (resp., X_2/Y_1) are far away, the resulting matrix is *nearly* block diagonal, i.e.,

$$\mathscr{K}(X, Y) \approx \begin{bmatrix} \mathscr{K}(X_1, Y_1) & \mathcal{O}(\varepsilon) \\ \mathcal{O}(\varepsilon) & \mathscr{K}(X_2, Y_2) \end{bmatrix} \tag{16}$$

for some small ε. This is a challenging situation for ACA since it will need to sweep through the initial block completely before considering the other one. In practice, heuristics can help alleviate the issue; see ACA+ [7], for instance. Those heuristics, however, do not come with any guarantees. Strong RRQR, on the other hand, does not suffer from this drawback and picks optimal nodes in each cluster from the start. It guarantees stability and convergence.

5.4. The need for weights. Another characteristic of Algorithm 1 is the presence of weights. We illustrate here why this is necessary in general. Algorithm 1 uses $\overline{X}$ and $\overline{Y}$ both to select interpolation points (the "columns" of the RRQR) and to evaluate the resulting error (the "rows"). Hence, a nonuniform distribution of points



(a) The geometry. Kernel is $1/r$, $\epsilon = 0.01$.

(b) Error with and without weights in Algorithm 1.

FIG. 8. *Benchmark demonstrating the importance of using weights in the RRQR factorization. The setup for the benchmark is described in the text. The bottom curve in the right panel, which uses weights, has much improved accuracy.*

leads to over- or underestimated L_2 error and to a biased interpolation point selection. The weights, roughly equal to the (square root of) the inverse points density, alleviate this effect. This is a somewhat small effect in the case of Chebyshev nodes and weights as the weights have limited amplitudes.

To illustrate this phenomenon more dramatically, consider the situation depicted in Figure 8. We define $\overline{X}$ and $\overline{Y}$ in the following way. Align two segments of N points, separated by a small interval of length $\epsilon = 1/N$ with $N = 200$. At the close extremities we insert $25N$ additional points inside small 3D spheres of diameter ϵ. As a result, $|\overline{X}| = |\overline{Y}| = 26N = 5{,}200$, and $\overline{X}$, $\overline{Y}$ are strongly nonuniform. We see that the small spheres hold a large number of points in an interval of length N^{-1}. As a result, their associated weight should be proportional to $N^{-1/2}$, while the weight for the points on the segments should be proportional to 1. Then we apply Algorithm 1 with and without weights, and we evaluate the error on the segments using $|X| = |Y| = 10{,}000$ equispaced points as a proxy for the L_2 error.

When using a rank $r_0 = 200$, the CUR decomposition picks only 6 more points on the segments (outside the spheres) for the weighted case compared to the unweighted. However, this is enough to dramatically improve the accuracy, as Figure 8(b) shows. Overall, the presence of weights has a large effect, and this shows that in general, one should appropriately weigh the node matrix K_w to ensure maximum accuracy.

5.5. Computational complexity. We finally study the computational complexity of the algorithm. It's important to note that two kinds of operations are involved: kernel evaluations and classical flops. As they may potentially differ in cost, we keep those separated in the following analysis.

The cost of the various parts of the algorithm is the following:

- $\mathcal{O}(r_0^2)$ kernel evaluations for the interpolation, i.e., the construction of $\overline{X}$ and $\overline{Y}$ and the construction of $\mathscr{K}(\overline{X}, \overline{Y})$;
- $\mathcal{O}(r_0^2 r_1)$ flops for the RRQR over $\mathscr{K}(\overline{X}, \overline{Y})$ and $\mathscr{K}(\overline{X}, \overline{Y})^\top$;
- $\mathcal{O}((m+n)r_1)$ kernel evaluations for computing $\mathscr{K}(X, \widehat{Y})$ and $\mathscr{K}(\widehat{X}, Y)$, respectively (with $m = |X|$ and $n = |Y|$);
- $\mathcal{O}(r_1^3)$ flops for $\mathscr{K}(\widehat{X}, \widehat{Y})^{-1}$ (through, say, an LU factorization).

So the total complexity of building the three factor is $\mathcal{O}((m+n)r_1)$ kernel evaluations.

If $m = n$ and $r_1 \approx r$, the total complexity is

$$\mathcal{O}((m+n)r_1) \approx \mathcal{O}(nr)\,.$$

Also note that the memory requirements are, clearly, of order $\mathcal{O}((m+n)r_1)$.

When applying this low-rank matrix on a given input vector $f(Y) \in \mathbb{R}^n$, the cost is

- $\mathcal{O}(r_1 n)$ flops for computing $w_1 = \mathscr{K}(\widehat{X}, Y) f(Y)$;
- $\mathcal{O}\big(r_1^2\big)$ flops for computing $w_2 = \mathscr{K}(\widehat{X}, \widehat{Y})^{-1} w_1$, assuming a factorization of $\mathscr{K}(\widehat{X}, \widehat{Y})$ has already been computed;
- $\mathcal{O}(m r_1)$ flops for computing $w_3 = \mathscr{K}(X, \widehat{Y}) w_2$.

So the total cost is

$$\mathcal{O}((m+n)r_1) \approx \mathcal{O}(nr)$$

flops if $m = n$ and $r_1 \approx r$.

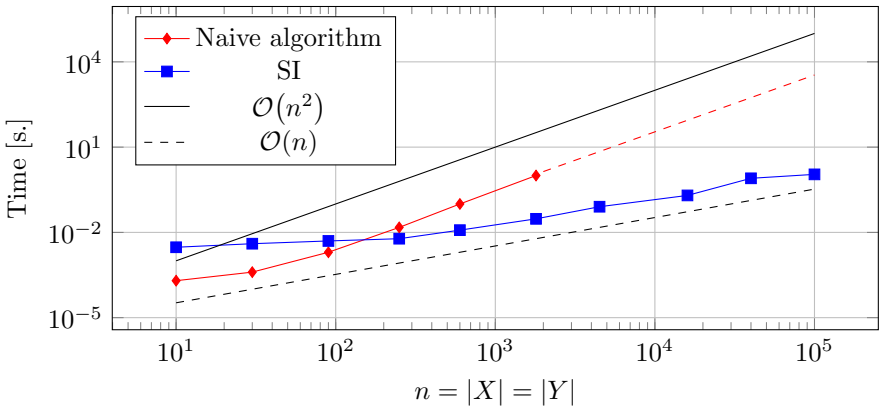
To illustrate those results, Figure 9(a) shows, using the same setup as in the 2D square example of subsection 5.1, the time (in seconds) taken by our algorithm versus the time taken by a naive algorithm that would first build $\mathscr{K}(X, Y)$ and then perform a rank-revealing QR on it. Time is given as a function of n for a fixed accuracy $\epsilon = 10^{-8}$. One should not focus on the absolute values of the timing but rather the asymptotic complexities. In this case, the $\mathcal{O}(n)$ and $\mathcal{O}\big(n^2\big)$ complexities clearly appear, and our algorithm scales much better than the naive one (or, really, than any algorithm that requires building the full matrix first). Note that we observe no loss of accuracy as n grows. Also note that the plateau at the beginning of the skeletonized interpolation curve is all the overhead involved in selecting the Chebyshev points $\overline{X}$ and $\overline{Y}$ using some heuristic. This is very implementation dependent and could be reduced significantly with a better or more problem-tailored algorithm. However, since this is by design independent of X and Y (and, hence, n) it does not affect the asymptotic complexity.

Figure 9(b) shows the time as a function of the desired accuracy ϵ for a fixed number of mesh points $n = 10^5$. Since the singular values of $\mathscr{K}(X, Y)$ decay exponentially, one has $r \approx \mathcal{O}\big(\log\big(\frac{1}{\epsilon}\big)\big)$. The complexity of the algorithm being $\mathcal{O}(nr)$, we expect the time to be proportional to $\log\big(\frac{1}{\epsilon}\big)$. This is indeed what we observe.

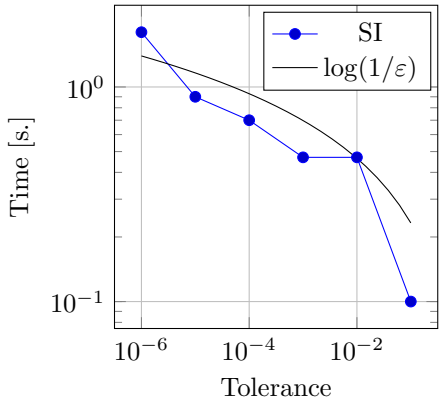
Figure 9(c) depicts the time as a function of the rank r for a fixed accuracy $\epsilon = 10^{-8}$ and number of mesh points $n = 10^5$. In that case, to vary the rank and keep ϵ fixed, we change the geometry and observe the resulting rank. This is done by moving the top-right square (see Figure 4(b)) towards the bottom-left one (keeping approximately one cluster diameter between them) or away from it (up to 6 diameters). The rank displayed is the rank obtained by the factorization. As expected, the algorithm scales linearly as a function of r.

6. Conclusion. In this work, we built a kernel matrix low-rank approximation based on skeletonized interpolation. This can be seen as an optimal way to interpolate families of functions using a custom basis.

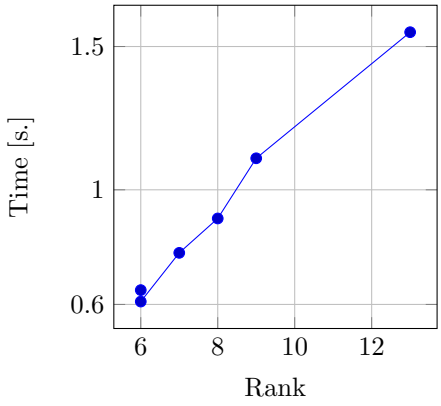
This type of interpolation, by design, is always at least as good as polynomial interpolation as it always requires the minimal number of basis functions for a given approximation error. We proved in this paper the asymptotic convergences of the scheme for kernels exhibiting fast (i.e., faster than polynomial) decay of singular values. We also proved the numerical stability of general Schur-complement types of formulas when using a backward stable algorithm.



(a) Time as a function of n for a fixed tolerance. The plateau is the overhead in the skeletonized interpolation algorithm that is independent of n.



(b) Time as a function of ϵ for fixed clusters of points.



(c) Time as a function of r. The rank is varied by increasing the distance between the two clusters for a fixed tolerance and number of points.

FIG. 9. *Timings experiments on skeletonized interpolation.*

In practice, the algorithm exhibits a low computational complexity of $\mathcal{O}(nr)$ with small constants and is very simple to use. Furthermore, the accuracy can be set a priori, and in practice we observe nearly optimal convergence of the algorithm. Finally, the algorithm is completely insensible to the mesh point distribution, leading to more stable sets of "pivots" than random sampling or ACA.

Appendix A. Proofs of the theorems.

Proof of Lemma 1. This bound on the Lagrange basis is a classical result related to the growth of the Lebesgue constant in polynomial interpolation. For $\overline{m}$ Chebyshev nodes of the first kind on $[-1, 1]$ and the associated Lagrange basis functions $\ell_1, \dots, \ell_{\overline{m}}$ we have the following result [23, equation (13)]:

$$\max_{x\in[-1,1]} \sum_{i=1}^{\overline{m}} |\ell_i(x)| \leq \frac{2}{\pi} \log(\overline{m}+1) + 0.974 = \mathcal{O}(\log(\overline{m}))\,.$$

This implies that in one dimension,

$$\|S(x,\overline{X})\|_2 \le \|S(x,\overline{X})\|_1 = \mathcal{O}(\log \overline{m})\,.$$

Going from one to d dimensions can be done using Kronecker products. Indeed, for $x \in \mathbb{R}^d$,

$$S(x,\overline{X}) = S(x_1,\overline{X}_1) \otimes \cdots \otimes S(x_d,\overline{X}_d),$$

where $x = (x_1,\ldots,x_d)$ and $\overline{X}_1,\ldots,\overline{X}_d$ are the one-dimensional Chebyshev nodes. Since for all $a \in \mathbb{R}^m, b \in \mathbb{R}^n$, $\|a \otimes b\|_2 = \sqrt{\sum_{i,j}(a_i b_j)^2} = \|ab^\top\|_F = \|a\|_2\|b\|_2$, it follows that

$$\|S(x,\overline{X})\|_2 = \|S(x_1,\overline{X}_1) \otimes \cdots \otimes S(x_d,\overline{X}_d)\|_2 = \prod_{i=1}^{d} \|S(x_i,\overline{X}_i)\|_2 = \prod_{i=1}^{d} \mathcal{O}(\log \overline{m}_i)\,.$$

This implies, using a fairly loose bound,

$$\|S(x,\overline{X})\|_2 = \mathcal{O}\big(\log(|\overline{X}|)^d\big)\,.$$

The same argument can be done for $T(y,\overline{Y})$.

In 1D, the weights are

$$w_k = \frac{\pi}{\overline{m}} \sin\left(\frac{2k-1}{2\overline{m}}\pi\right)$$

for $k = 1,\ldots,\overline{m}$. Obviously, $w_k > 0$. Clearly, $w_k < \frac{\pi}{\overline{m}}$. Also, the minimum being reached at $k = 1$ or $k = \overline{m}$,

$$w_k \ge \frac{\pi}{\overline{m}} \sin\left(\frac{\pi}{2\overline{m}}\right) > \frac{\pi}{\overline{m}} \frac{2\pi}{2\pi\overline{m}} = \frac{\pi}{\overline{m}^2}.$$

Since the nodes in d dimensions are products of the nodes in 1D, it follows that

$$\|\operatorname{diag}(\overline{W}_X)\|_2 \le \frac{\pi^d}{\overline{m}},$$
$$\|\operatorname{diag}(\overline{W}_X)^{-1}\|_2 \le \frac{\overline{m}^2}{\pi^d}.$$

The result follows. ∎

Proof of Lemma 3. We show the result for the second equation. This requires using, consecutively, the interpolation result and the CUR decomposition one. First, from Lemma 1 and the interpolation, one can write

$$\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y) = \mathscr{K}(\widehat{X},\widehat{Y})^{-1}\left[\mathscr{K}(\widehat{X},\overline{Y})T(y,\overline{Y})^\top + E_{\mathrm{INT}}(\widehat{X},y)\right].$$

Then, introducing the weight matrices and applying Lemma 2 on the interpolation matrix,

$$\begin{aligned}\mathscr{K}(\widehat{X},\overline{Y}) &= \operatorname{diag}(\widehat{W}_X)^{-1/2}\operatorname{diag}(\widehat{W}_X)^{1/2}\mathscr{K}(\widehat{X},\overline{Y})\operatorname{diag}(\overline{W}_Y)^{1/2}\operatorname{diag}(\overline{W}_Y)^{-1/2}\\ &= \operatorname{diag}(\widehat{W}_X)^{-1/2}\Big\{\operatorname{diag}(\widehat{W}_X)^{1/2}\mathscr{K}(\widehat{X},\widehat{Y})\operatorname{diag}(\widehat{W}_Y)^{1/2}\begin{bmatrix} I & \breve{T}^\top\end{bmatrix}\\ &\qquad + E_{\mathrm{QR}}(\widehat{X},\overline{Y})\Big\}\operatorname{diag}(\overline{W}_Y)^{-1/2}.\end{aligned}$$

Finally, combining and distributing all the factors gives us

$$\begin{aligned}\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y) &= \operatorname{diag}(\widehat{W}_Y)^{1/2}\begin{bmatrix} I & \breve{T}^\top\end{bmatrix}\operatorname{diag}(\overline{W}_Y)^{-1/2}T(y,\overline{Y})^\top \\ &+ \mathscr{K}(\widehat{X},\widehat{Y})^{-1}\operatorname{diag}(\widehat{W}_X)^{-1/2}E_{\mathrm{QR}}(\widehat{X},\overline{Y})\operatorname{diag}(\overline{W}_Y)^{-1/2}T(y,\overline{Y})^\top \\ &+ \mathscr{K}(\widehat{X},\widehat{Y})^{-1}E_{\mathrm{INT}}(\widehat{X},y).\end{aligned}$$

Here, we can bound all terms:

- For the first term, Lemmas 1 and 2 show that the expression is bounded by a polynomial.
- For the second term use the fact that

$$\|\hat{K}_w^{-1}\|_2 \le p^2(r_0,r_1)\frac{1}{\epsilon} \Rightarrow \|\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\|_2 = p'(r_0,r_1)\frac{1}{\epsilon};$$

  hence, since $\|E_{QR}(\overline{X},\overline{Y})\|_2 = \epsilon$, the product is again bounded by a polynomial since the ϵ cancel out.
- The last term can be bounded in a similar way using

$$E_{\mathrm{INT}}(x,y) = \mathcal{O}(\delta) \le \mathcal{O}(\epsilon)\,.$$

We conclude that there exists a polynomial q such that

$$\|\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y)\|_2 = \mathcal{O}(q(r_0,r_1))\,.$$

The proof is similar in x. ∎

Proof of Theorem 2. Combining interpolation and CUR decomposition results, one can write

$$\begin{aligned}\mathscr{K}(x,y) &= S(x,\overline{X})\mathscr{K}(\overline{X},\overline{Y})T(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y) \\ &= S_w(x,\overline{X})\mathscr{K}_w(\overline{X},\overline{Y})T_w(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y) \\ &= S_w(x,\overline{X})\left[\begin{bmatrix} I \\ \breve{S}\end{bmatrix}\mathscr{K}_w(\widehat{X},\widehat{Y})\begin{bmatrix} I & \breve{T}^\top\end{bmatrix} + E_{\mathrm{QR}}(\overline{X},\overline{Y})\right]T_w(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y) \\ &= S_w(x,\overline{X})\begin{bmatrix} \mathscr{K}_w(\widehat{X},\widehat{Y}) \\ \breve{S}\mathscr{K}_w(\widehat{X},\widehat{Y})\end{bmatrix}\mathscr{K}_w(\widehat{X},\widehat{Y})^{-1}\begin{bmatrix}\mathscr{K}_w(\widehat{X},\widehat{Y}) & \mathscr{K}_w(\widehat{X},\widehat{Y})\breve{T}^\top\end{bmatrix}T_w(y,\overline{Y})^\top \\ &\quad + S_w(x,\overline{X})E_{\mathrm{QR}}(\overline{X},\overline{Y})T_w(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y) \\ &= S_w(x,\overline{X})\left[\mathscr{K}_w(\overline{X},\widehat{Y}) + E_{\mathrm{QR}}(\overline{X},\widehat{Y})\right]\mathscr{K}_w(\widehat{X},\widehat{Y})^{-1}\left[\mathscr{K}_w(\widehat{X},\overline{Y}) + E_{\mathrm{QR}}(\widehat{X},\overline{Y})\right] \\ &\quad \times T_w(y,\overline{Y})^\top + S_w(x,\overline{X})E_{\mathrm{QR}}(\overline{X},\overline{Y})T_w(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y) \\ &= (\mathscr{K}(x,\widehat{Y}) + E_{\mathrm{INT}}(x,\widehat{Y}))\mathscr{K}(\widehat{X},\widehat{Y})^{-1}(\mathscr{K}(\widehat{X},y) + E_{\mathrm{INT}}(\widehat{X},y)) \\ &\quad + S_w(x,\overline{X})E_{\mathrm{QR}}(\overline{X},\widehat{Y})\mathscr{K}_w(\widehat{X},\widehat{Y})^{-1}\mathscr{K}_w(\widehat{X},\overline{Y})T_w(y,\overline{Y})^\top \\ &\quad + S_w(x,\overline{X})\mathscr{K}_w(\overline{X},\widehat{Y})\mathscr{K}_w(\widehat{X},\widehat{Y})^{-1}E_{\mathrm{QR}}(\widehat{X},\overline{Y})T_w(y,\overline{Y})^\top \\ &\quad + S_w(x,\overline{X})E_{\mathrm{QR}}(\overline{X},\widehat{Y})\mathscr{K}_w(\widehat{X},\widehat{Y})^{-1}E_{\mathrm{QR}}(\widehat{X},\overline{Y})T_w(y,\overline{Y})^\top \\ &\quad + S_w(x,\overline{X})E_{\mathrm{QR}}(\overline{X},\overline{Y})T_w(y,\overline{Y})^\top + E_{\mathrm{INT}}(x,y).\end{aligned}$$

Distributing everything, factoring the weights matrices, and simplifying, we obtain

the following, where we indicate the bounds on each term on the right:

$$
\begin{aligned}
\mathscr{K}(x,y) &= \mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y) && \text{Approximation}\\
&+ E_{\mathrm{INT}}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},y) && \mathcal{O}(\delta q(r_0,r_1))\\
&+ \mathscr{K}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}E_{\mathrm{INT}}(\widehat{X},y) && \mathcal{O}(\delta q(r_0,r_1))\\
&+ E_{\mathrm{INT}}(x,\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}E_{\mathrm{INT}}(\widehat{X},y) && \mathcal{O}(\delta p'(r_0,r_1))\\
&+ S(x,\overline{X})\,\mathrm{diag}(\overline{W}_X)^{-1/2}E_{\mathrm{QR}}(\overline{X},\widehat{Y})\,\mathrm{diag}(\widehat{W}_Y)^{-1/2}\\
&\quad \mathscr{K}(\widehat{X},\widehat{Y})^{-1}\mathscr{K}(\widehat{X},\overline{Y})T(y,\overline{Y})^{\top} && \mathcal{O}\big(\epsilon(\log r_0)^{2d}q(r_0,r_1)r_0^2\big)\\
&+ S(x,\overline{X})\mathscr{K}(\overline{X},\widehat{Y})\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\,\mathrm{diag}(\widehat{W}_X)^{-1/2}\\
&\quad E_{\mathrm{QR}}(\widehat{X},\overline{Y})\,\mathrm{diag}(\overline{W}_Y)^{-1/2}T(y,\overline{Y})^{\top} && \mathcal{O}\big(\epsilon(\log r_0)^{2d}q(r_0,r_1)r_0^2\big)\\
&+ S(x,\overline{X})\,\mathrm{diag}(\overline{W}_X)^{-1/2}E_{\mathrm{QR}}(\overline{X},\widehat{Y})\\
&\quad \mathrm{diag}(\widehat{W}_Y)^{-1/2}\mathscr{K}(\widehat{X},\widehat{Y})^{-1}\,\mathrm{diag}(\widehat{W}_X)^{-1/2}\\
&\quad E_{\mathrm{QR}}(\widehat{X},\overline{Y})\,\mathrm{diag}(\overline{W}_Y)^{-1/2}T(y,\overline{Y})^{\top} && \mathcal{O}\big(\epsilon(\log r_0)^{2d}r_0^2p(r_0,r_1)\big)\\
&+ S(x,\overline{X})\,\mathrm{diag}(\overline{W}_X)^{-1/2}E_{\mathrm{QR}}(\overline{X},\overline{Y})\\
&\quad \mathrm{diag}(\overline{W}_Y)^{-1/2}T(y,\overline{Y})^{\top} && \mathcal{O}\big(\epsilon(\log r_0)^{2d}r_0^2\big)\\
&+ E_{\mathrm{INT}}(x,y) && \mathcal{O}(\delta)\,.
\end{aligned}
$$

This concludes the proof. ∎

Acknowledgments. We would like to thank Cleve Ashcraft for his ideas and comments on the paper, as well as the anonymous reviewer for his careful reading and pertinent suggestions that greatly improved the paper.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Natl. Bureau Standards Appl. Math. Ser. 55, U.S. Department of Commerce, 1972.

[2] J. Barnes and P. Hut, *A hierarchical $O(N \log N)$ force-calculation algorithm*, Nature, 324 (1986), pp. 446–449.

[3] M. Bebendorf, *Approximation of boundary element matrices*, Numer. Math., 86 (2000), pp. 565–589.

[4] M. Bebendorf and S. Rjasanow, *Adaptive low-rank approximation of collocation matrices*, Computing, 70 (2003), pp. 1–24.

[5] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A fresh approach to numerical computing*, SIAM Rev., 59 (2017), pp. 65–98, https://doi.org/10.1137/141000671.

[6] S. Börm and L. Grasedyck, *Low-rank approximation of integral operators by interpolation*, Computing, 72 (2004), pp. 325–332.

[7] S. Börm and L. Grasedyck, *Hybrid cross approximation of integral operators*, Numer. Math., 101 (2005), pp. 221–249.

[8] 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.

[9] E. Corona, A. Rahimian, and D. Zorin, *A tensor-train accelerated solver for integral equations in complex geometries*, J. Comput. Phys., 334 (2017), pp. 145–169.

[10] W. Fong and E. Darve, *The black-box fast multipole method*, J. Comput. Phys., 228 (2009), pp. 8712–8725.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 2012.

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

[13] M. Gu and S. C. 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.
[14] A. Guven, *Quantitative Perturbation Theory for Compact Operators on a Hilbert Space*, Ph.D. thesis, Queen Mary University of London, 2016.
[15] K. Hackbusch, *A sparse $\mathcal{H}$-matrix arithmetic. Part* II: *Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.
[16] W. Hackbusch, *A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part* I: *Introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.
[17] W. Hackbusch and S. Börm, *Data-sparse approximation by adaptive $\mathcal{H}$ 2-matrices*, Computing, 69 (2002), pp. 1–35.
[18] W. Hackbusch and Z. P. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.
[19] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.
[20] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2002, https://doi.org/10.1137/1.9780898718027.
[21] K. L. Ho and S. Olver, *LowRankApprox.jl: Fast Low-Rank Matrix Approximation in Julia*, 2018, https://doi.org/10.5281/zenodo.1254148.
[22] K. L. Ho and L. Ying, *Hierarchical interpolative factorization for elliptic operators: Integral equations*, Comm. Pure Appl. Math., 69 (2016), pp. 1314–1353.
[23] B. A. Ibrahimoglu, *Lebesgue functions and Lebesgue constants in polynomial interpolation*, J. Inequal. Appl., 2016 (2016), 93.
[24] M. W. Mahoney and P. Drineas, *CUR matrix decompositions for improved data analysis*, Proc. Natl. Acad. Sci. USA, 106 (2009), pp. 697–702.
[25] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*, tech. report, Technical University of Denmark, 2012.
[26] M. Reed and B. Simon, *Methods of Modern Mathematical Physics. Vol.* 1. *Functional Analysis*, Academic, 1980.
[27] M. Renardy and R. C. Rogers, *An Introduction to Partial Differential Equations*, Texts Appl. Math. 13, Springer Science & Business Media, 2004.
[28] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.
[29] E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method*, Computing, 64 (2000), pp. 367–380.
[30] Z. Wu and T. Alkhalifah, *The optimized expansion based low-rank method for wavefield extrapolation*, Geophys., 79 (2014), pp. T51–T60.
[31] N. Yarvin and V. Rokhlin, *Generalized Gaussian quadratures and singular value decompositions of integral operators*, SIAM J. Sci. Comput., 20 (1998), pp. 699–718, https://doi.org/10.1137/S1064827596310779.