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 $O(nr)$ for a $n \times n$ 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 RRLU-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 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

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

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

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

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

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

$$a(x)u(x) + \int \mathcal{K}(x, y)u(y)dy = f(x) \quad \forall x \in \mathcal{X}$$

where the discretization leads to a linear system of the form

$$(1) \quad a_i u_i + \sum_j K_{ij} u_j = f_i$$

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 $\mathcal{K}$ is smooth over $\mathcal{X} \times \mathcal{Y}$ and hence $\mathcal{K}(X, Y)$ is low-rank. This corresponds to a submatrix of the complete $K$. Being able to efficiently compute a low-rank factorization of such submatrix would lead to significant computational savings. By “smooth” we usually refer to 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.

---

*Institute for Computational & Mathematical Engineering, Stanford University. Huang Engineering Center, 475 Via Ortega, Suite B060, Stanford, CA-94305, USA. lcambier@stanford.edu, https://people.stanford.edu/lcambier

†Department of Mechanical Engineering, Stanford University. 452 Escondido Mall, Bldg 520, Room 125, Stanford, CA-94305, USA
The smooth kernel function $K$ over which $K$ is defined, i.e., $X \times Y$.

Variables, $x \in X$, $y \in Y$.

The mesh of points over which to approximate $K$, i.e., $X \times Y$.

The kernel matrix, $K = K(X, Y)$, $K_{ij} = K(x_i, y_j)$.

$m$, $n$ are the number of Chebyshev tensor nodes, $m = |X|$, $n = |Y|$.

The tensor grids of Chebyshev points $\hat{X}, \hat{Y}$.

The subsets of $X$ and $Y$ outputted by the algorithm used to build the low-rank approximation.

$m$, $\pi$ are the “interpolation” rank of $K$, i.e., $r_0 = \min(|X|, |Y|)$.

$r_1$ is the Skeletonized Interpolation rank of $K$, i.e., $r_1 = |\hat{X}| = |\hat{Y}|$.

$r$ is the rank of the continuous SVD of $K$.

$S(x, \hat{X})$, $T(y, \hat{Y})$ are row vectors of the Lagrange basis functions, based on $\hat{X}$ and $\hat{Y}$ and evaluated at $x$ and $y$, respectively. Each column is one Lagrange basis function.

$\hat{S}(x, \hat{X})$, $\hat{T}(y, \hat{Y})$ are row vectors of Lagrange basis functions, based on $\hat{X}$ and $\hat{Y}$, built using the Skeletonized Interpolation and evaluated at $x$ and $y$, respectively. Each column is one function.

$w_k$, $w_l$ are Chebyshev integration weights.

$\text{diag}([W_X])$, $\text{diag}([W_Y])$ are diagonal matrices of integration weights when integration is done at nodes $\hat{X}$ and $\hat{Y}$.

Table 1: Notations used in the paper

Low-rank factorization means that we seek a factorization of $K = \mathcal{K}(X, Y)$ 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 $\mathcal{K}$ 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 $\mathcal{K}(X, Y)$ denotes the matrix $K_{ij} = \mathcal{K}(x_i, y_j)$. Small-case letters $x$ and $y$ denote arbitrary variables, while capital-case letters $X$, $\hat{X}$, $\check{X}$, $\tilde{X}$ denotes 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 $O(n^3)$ time and even $O(n^2)$ storage complexity. The now traditional method used to deal with such matrices 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) \[27, 11, 1\] 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, \[9\] 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 \[17\] first computes a low-rank approximation of $K(x, y)$ as

$$ K(x, y) \approx \sum_{i} \kappa_i(x; y_0) \phi_i(y) $$

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

Bebendorf and Rjasanow proposed the Adaptive Cross Approximation \[3\], 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 \[6\]) if it cannot capture all necessary information to properly build the low-rank basis and lacks convergence guarantees.

In the realm of analytic approximations, \[30\] (and similarly \[5\], \[6\], \[9\] and \[29\] in the Fourier space) interpolate $K(x, y)$ over $\mathcal{X} \times \mathcal{Y}$ using classical interpolation methods (for instance, polynomial interpolation at Chebyshev nodes in \[9\]), resulting in expressions like

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

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 $K(\tilde{X}, \tilde{Y})$, for instance using SVD \[9\] or ACA \[6\]. Furthermore, \[30\] takes the SVD of a scaled $K(\tilde{X}, \tilde{Y})$ matrix to further recompress the approximation and obtain an explicit expression for $u_r$ and $v_r$ such that

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

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

Bebendorf \[2\] builds a low-rank factorization of the form

$$ K(x, y) = K(x, \tilde{Y}) \cdot K(\tilde{X}, \tilde{Y})^{-1} \cdot K(\tilde{X}, y) \quad \text{(2)} $$

where the nodes $\tilde{X}$ and $\tilde{Y}$ are interpolation nodes of an interpolation of $K(x, y)$ built iteratively.

Our method inserts itself amongst 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 $O(nr)$. In addition, \[\text{it is worth noting that our method differs from interpolation-based algorithm as we never}\]
explicitly build the $S(x, \tilde{X})$ and $T(y, \tilde{Y})$ matrices containing the basis functions. We merely rely on their existence.

$\mathcal{H}$-matrices [15, 16, 14] 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 & Ying [21] as well as Tyrtyshnikov [28] are of particular interest for us. They use Skeletonization of the matrix to reduce storage and computation cost. In [21], they combine Skeletonization and Sparsification to keep compressing blocks of $\mathcal{H}$-matrices. [28] uses a somewhat non-traditional Skeletonization technique to also compress hierarchical kernel matrices.

Finally, extending the framework of low-rank compression, [8] uses tensor-train compression to re-write $\mathcal{K}(X, Y)$ as a tensor with one dimension per coordinate, i.e., $\mathcal{K}(x_1, \ldots, x_d, y_1, \ldots, y_d)$ 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 $O(m + n)$. The main advantages of the method are as follows:

1. The complexity of our method is $O(r(m + n))$ (in terms of kernel function $\mathcal{K}$ 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 (GEMM) and LAPACK (RRQR, LU) kernels.

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

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

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

In practice however, this method may become inefficient as it requires assembling the matrix $\mathcal{K}(X, Y)$ 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 from 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 $\tilde{X}, \tilde{Y}$ and associated integration weights $\tilde{W}_X, \tilde{W}_Y$ and then consider the matrix

$$K_w = \text{diag}(\tilde{W}_X)^{1/2} \mathcal{K}(\tilde{X}, \tilde{Y}) \text{diag}(\tilde{W}_Y)^{1/2}$$

Denote $r_0 = \min(|\tilde{X}|, |\tilde{Y}|)$. Based on interpolation properties, we will show that this matrix is closely related to the continuous kernel $\mathcal{K}(x, y)$. In particular, they share
a similar spectrum. Then, we select the sets \( \hat{X} \subset X \), \( \hat{Y} \subset Y \) by performing a strong rank-revealing QR’s \([12]\) 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 \( \hat{X} \) by selecting the elements of \( P_x \) associated to the largest rows of \( R_x \) and similarly for \( \hat{Y} \) (if they differ in size, extend the smallest). We denote the rank of this factorization \( r_1 = |\hat{X}| = |\hat{Y}| \), and in practice, we observe that \( r_1 \approx r_{SV D} \), where \( r_{SV D} \) is the rank the truncated SVD of \( \mathcal{H}(X, Y) \) would provide. The resulting factorization is

\[
\mathcal{H}(X, Y) \approx \mathcal{H}(X, \hat{Y}) \mathcal{H}(\hat{X}, \hat{Y})^{-1} \mathcal{H}(\hat{X}, Y)
\]

Note that, in this process, at no point did we build any Lagrange basis function associated with \( X \) and \( Y \). We only evaluate the kernel \( \mathcal{H} \) at \( X \times Y \).

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

\[
\mathcal{H}(X, Y) \approx S(X, \hat{X}) \mathcal{H}(\hat{X}, \hat{Y}) T(Y, \hat{Y})^\top
\]

In this expression, we collect the Lagrange basis functions (each one associated to a node of \( \hat{X} \)) evaluated at \( X \) in the columns of \( S(X, \hat{X}) \) and similarly for \( T(Y, \hat{Y}) \). This provides a robust way of building a low-rank approximation. The rank \( r_0 = \min(|\hat{X}|, |\hat{Y}|) \), however, is usually much larger than the true rank \( r_{SV D} \) and than \( r_1 \) (given a tolerance). See subsubsection 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 \( O(n^3) \). 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 \([18]\), have a reduced computational cost of \( O(n^2 r) \), but still scale quadratically with \( n \).

Methods like ACA \([3]\), the rank-revealing LU factorization with rook pivoting \([10]\) and techniques that randomly sample from columns and rows of the matrix scale like \( O(nr) \), 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 offers strong stability and accuracy guarantees. The fact that new interpolation points, \( \hat{X} \) and \( \hat{Y} \), 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 the fast multipole method, etc., but they are limited to specific kernels. Other techniques, which are more general, like Taylor expansion and Chebyshev interpolation \([9]\), 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 $\mathcal{K}$ and its singular value decomposition [25, theorem VI.17]:

**Theorem 1 (Singular Value Decomposition).** Suppose $\mathcal{K} : [-1,1]^d \times [-1,1]^d$ is square integrable. Then there exist two sequences of orthogonal functions $\{u_i\}_{i=1}^\infty$ and $\{v_i\}_{i=1}^d$ and a non-increasing sequence of non-negative real number $\{s_i\}_{i=1}^\infty$ such that

$$\mathcal{K}(x,y) = \sum_{s=1}^\infty \sigma_s u_s(x)v_s(y)$$

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

$$\mathcal{K}(X,Y) \approx \sum_{s=1}^r \sigma_s u_s(X)v_s(Y) = [u_1(X) \cdots u_r(X)] [\sigma_1 \cdots \sigma_r] [v_1^T(Y) \cdots v_r^T(Y)]$$

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 $\mathcal{K}(x,y)$, the faster the decay of the $\sigma_s$’s and the lower the rank.

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

$$\mathcal{K}(X,Y) \approx S(X,X) \mathcal{K}(X,Y) T(Y,Y)^T$$

The interpolation functions $S(x,X)$ and $T(y,Y)$ 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 $\mathcal{K}$.

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, $S(x,X)$ and $T(y,Y)$ 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:

$$f(x) - S(x,X)f(X) = \frac{f^{(m)}(\xi)}{m!} \prod_{j=1}^m (x - X_j)$$

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

$$\prod_{j=1}^m (x - X_j) = x^m - r_{m-1}(x)$$

is minimal, where $r_{m-1}(x)$ is a degree $m-1$ polynomial. Since we are free to vary the interpolation points, then we have $m$ parameters (the location of the interpolation points) and $m$ coefficients in $r_{m-1}$. By varying the location of the interpolation points, we can recover any polynomial $r_{m-1}$. Chebyshev points are known to solve
this problem optimally. That is, they lead to an \( r_{m-1} \) such that \( \max_x |x^m - r_{m-1}(x)| \) 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 \( \mathcal{K}(x, \lambda) \), where \( \lambda \) 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:

\[
\min_{\{\lambda_s, v_s\}} \max_{\lambda} \left\| \mathcal{K}(x, \lambda) - \sum_{s=1}^{m} \mathcal{K}(x, \lambda_s)v_s(\lambda) \right\|_2^2, \quad v_s(\lambda_t) = \delta_{st}
\]

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

Define \( \Lambda = \{\lambda_1, \ldots, \lambda_m\} \). From there, we identify a set of \( m \) interpolation nodes \( \hat{X} \) such that the square matrix

\[
\mathcal{K}(\hat{X}, \hat{\Lambda}) := \begin{bmatrix} \mathcal{K}(\hat{X}, \lambda_1) & \cdots & \mathcal{K}(\hat{X}, \lambda_m) \end{bmatrix}
\]

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

\[
\tilde{S}(x, \hat{X}) = \mathcal{K}(x, \hat{\Lambda})\mathcal{K}(\hat{X}, \hat{\Lambda})^{-1}
\]

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

\[
\tilde{S}(x, \hat{X})\mathcal{K}(\hat{X}, \lambda_s) = \mathcal{K}(x, \lambda_s)
\]

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

\[
\tilde{S}(x, \hat{X})\mathcal{K}(\hat{X}, \lambda) \approx \sum_{s=1}^{m} \tilde{S}(x, \hat{X})\mathcal{K}(\hat{X}, \lambda_s)v_s(\lambda) = \sum_{s=1}^{m} \mathcal{K}(x, \lambda_s)v_s(\lambda) \approx \mathcal{K}(x, \lambda)
\]

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 and Chebyshev, is with rank-1 kernels:

\[
\mathcal{K}(x, \lambda) = u(x)v(\lambda)
\]

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

\[
\tilde{S}(x, \hat{X}) = \mathcal{K}(x, \lambda_1)\mathcal{K}(x_1, \lambda_1)^{-1}
\]

\[
\tilde{S}(x, \hat{X})\mathcal{K}(\hat{X}, \lambda) = u(x)v(\lambda_1)\frac{1}{u(x_1)v(\lambda_1)}u(x_1)v(\lambda) = u(x)v(\lambda)
\]

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 basis for its interpolation, a set of nearly optimal functions that approximate the left singular functions of \( \mathcal{K} \), rather than generic polynomial functions.
1.3.5. Proposed method. In this paper, we use the framework from subsubsection 1.3.4 to build an interpolation operator for the class of functions $\mathcal{K}(x, y)$, 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

$$\mathcal{K}(X, Y) \approx [\mathcal{K}(X, \hat{Y}), \mathcal{K}(\hat{X}, Y)^{-1}] \mathcal{K}(\hat{X}, \hat{Y}) \mathcal{K}(\hat{X}, Y)$$

and by comparing with Eq. 6, we recognize the interpolation operators:

$$\hat{S}(x, \hat{X}) = \mathcal{K}(x, \hat{Y}), \mathcal{K}(\hat{X}, \hat{Y})^{-1}, \quad \hat{T}(y, \hat{Y}) = \mathcal{K}(\hat{X}, y)^\top \mathcal{K}(\hat{X}, \hat{Y})^{-T}$$

These interpolation operators are nearly optimal; because of the way these operators are constructed we call the method “Skeletonized Interpolation.” The sets $\hat{X}$ and $\hat{Y}$ are the minimal sets such that if we sample $\mathcal{K}$ are these points we can interpolate $\mathcal{K}$ at any other point with accuracy $\epsilon$. In particular, $\hat{X}$ and $\hat{Y}$ are much smaller than their Chebyshev-interpolant counterparts $X$ and $Y$ and their size, $r_1$, is very close to $r$ in Eq. 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 $\mathcal{K}(\hat{X}, \Lambda)^{-1}$ as well as $\mathcal{K}(\hat{X}, Y)^{-1}$ in Eq. 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.

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 $m$ Chebyshev nodes of the first kind are defined as

$$\bar{x}_k = \cos\left(\frac{2k - 1}{2m} \pi\right) \quad k = 1, \ldots, m$$

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

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

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

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

- Recompress the grid by performing a strong rank-revealing QR factorization [12] of

$$\text{diag}(\overline{W}_X)^{1/2} \mathcal{K}(\overline{X}, \overline{Y}) \text{diag}(\overline{W}_Y)^{1/2}$$

and its transpose, up to accuracy $\epsilon$. This factorization is also named CUR decomposition [23, 7]. In the case of Chebyshev nodes of the first kind in 1D over $[-1, 1]$ the integration nodes are given by

$$w_k = \frac{\pi}{m} \sqrt{1 - \bar{x}_k^2} = \frac{\pi}{m} \sin\left(\frac{2k - 1}{2m} \pi\right)$$
The weights in $d$ dimensions are the products of the corresponding weights in 1D, and the \( \text{diag}(W_X) \) and \( \text{diag}(W_Y) \) matrices are simply the diagonal matrices of the integration weights. Denote

\[
r_1 = |\hat{X}| = |\hat{Y}|
\]

In case the sets $\hat{X}$ and $\hat{Y}$ outputted by those RRQR's are of slightly different size, extend the smallest to have the same size as the largest.

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

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

of rank $r_1 \approx r_{SV D}$.

**Algorithm 1** Skeletonized Interpolation

```
procedure SKELETONIZED_INTERPOLATION(\( \mathcal{K} : [-1,1]^d \times [-1,1]^d \rightarrow \mathbb{R} \), \( X \), \( Y \), \( \epsilon \), \( \delta \))

Build \( \bar{X} \) and \( \bar{Y} \), sets of Chebyshev nodes over \([-1,1]^d\) that interpolate \( \mathcal{K} \) with error \( \delta \) uniformly

Build \( K_w \) as

\[
K_w = \text{diag}(W_X)^{1/2} \mathcal{K}(X, Y) \text{diag}(W_Y)^{1/2}
\]

Extract $\hat{Y} \subseteq \bar{Y}$ by performing a strong RRQR over $K_w$ with tolerance $\epsilon$ ;

\[
K_w P_y = Q_y R_y
\]

Extract $\hat{X} \subseteq \bar{X}$ by performing a strong RRQR over $K_w^\top$ with tolerance $\epsilon$ ;

\[
K_w^\top P_x = Q_x R_x
\]

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

return

\( \mathcal{K}(X, Y) \approx \mathcal{K}(\hat{X}, \hat{Y}) \mathcal{K}(\hat{X}, \hat{Y})^{-1} \mathcal{K}(\hat{X}, \hat{Y}) \)
```

2.2. Theoretical Convergence.

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

2.2.2. Interpolation-related results. We first consider the interpolation itself. Consider \( \bar{X} \) and \( \bar{Y} \), constructed such as

\[
\mathcal{K}(x, y) = S(x, \bar{X}) \mathcal{K}(X, \bar{Y}) T(y, \bar{Y})^\top + E_{\text{INT}}(x, y)
\]

**Lemma 1** (Interpolation at Chebyshev Nodes). \( \forall x \in \mathcal{X} \) and \( \bar{X} \) tensor grids of Chebyshev nodes of the first kind,

\[
\|S(x, \bar{X})\|_2 = O(\log(|\bar{X}|)^d)
\]
where $\mathcal{X} \subset \mathbb{R}^d$. In addition, the weights, collected in the weight matrix $\text{diag}(W_X)$ are such that

$$\|\text{diag}(W_X)^{1/2}\|_2 \leq \frac{\pi^{d/2}}{\sqrt{m}} = O\left(\frac{1}{\sqrt{m}}\right)$$

$$\|\text{diag}(W_X)^{-1/2}\|_2 \leq \frac{m}{\pi^{d/2}} = O(m)$$

where $m = |X|$.

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.

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

Lemma 2 (CUR Decomposition of $K_w$). The partition $\overline{X} = \hat{X} \cup \tilde{X}, \overline{Y} = \hat{Y} \cup \tilde{Y}$ of Algorithm 1 is such that there exist $\hat{S}, \hat{T}, E_{QR}(\overline{X}, \overline{Y})$ matrices and a slowly-growing polynomial $p(r_0, r_1)$ such that

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

and where

$$\epsilon = \|E_{QR}(X, Y)\|_2 \leq p(r_0, r_1)\sigma_{r_1+1}(K_w)$$

$$\|\hat{S}\|_2 \leq p(r_0, r_1)$$

$$\|\hat{T}\|_2 \leq p(r_0, r_1)$$

Finally, we have

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

Proof. The first three results are direct applications of [7, theorem 3 and remark 5] as explained in the previous paragraph.

Finally, a less obvious result

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

$$\|\mathcal{K}(x, \hat{Y})\mathcal{K}(\tilde{X}, \hat{Y})^{-1}\|_2 = O(q(r_0, r_1))$$

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

We provide the proof in the appendix; the key ingredient is simply that $\|\hat{K}_w^{-1}\|_2 \leq p(r_0, r_1)^2\epsilon^{-1}$ from the RRQR properties; hence $K_w$ is ill-conditioned, but not arbitrarily. Its condition number grows like $\epsilon^{-1}$. Then, when multiplied by quantities like $\epsilon$ 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 $\text{diag}(W_X)^{1/2}K_w\text{diag}(W_Y)^{1/2}$.

Take the classical discrete SVD of $K_w$,

$$K_w = U \Sigma V^\top$$
We then have
\[
\mathcal{K}(x, y) = S_w(x, X) U \Sigma V^T T_w(y, Y)^T + E_{\text{INT}}(x, y)
\]

Then, denote the sets of new basis functions
\[
\mathbf{\pi}(x) = S_w(x, X) U \mathbf{v} \quad \mathbf{\pi}(y) = T_w(y, Y) V
\]

\[
\int_X \mathbf{\pi}_i(x) \mathbf{\pi}_j(x) \, dx = \sum_{k=1}^{r_0} w_k \left( \sum_{l=1}^{r_0} w_l^{-1/2} S_l(x_k) U_{li} \right) \left( \sum_{l=1}^{r_0} w_l^{-1/2} S_l(x_k) U_{lj} \right)
= \sum_{k=1}^{r_0} w_k \left( \delta_{kl} w_k^{-1/2} U_{li} \right) \left( \delta_{kl} w_k^{-1/2} U_{lj} \right)
= \sum_{k=1}^{r_0} w_k w_k^{-1/2} U_{ki} w_k^{-1/2} U_{kj} = \sum_{k=1}^{r_0} U_{ki} U_{kj} = \delta_{ij}
\]

The same result holds for \( \mathbf{\pi} \). This shows that we are implicitly building a factorization
\[
\mathcal{K}(x, y) = \sum_{s=1}^{\infty} \sigma_s u_s(x) v_s(y) = \sum_{s=1}^{r_0} \sigma_s(K_w) u_s(x) v_s(y) + E_{\text{INT}}(x, y)
\]

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

**2.2.5. Convergence of the Skeletonized interpolation.** We then present the main result of this paper:

**Theorem 2 (Convergence of Skeletonized Interpolation).** If \( \hat{X} \) and \( \hat{Y} \) are constructed following Algorithm 1, then there exist a polynomial \( r(r_0, r_1) \) such that for any \( x \in \mathcal{X} \) and \( y \in \mathcal{Y} \),
\[
|\mathcal{K}(x, y) - \mathcal{K}(x, \hat{Y}) \mathcal{K}(\hat{X}, \hat{Y})^{-1} \mathcal{K}(\hat{X}, y)| = \mathcal{O}(\epsilon r(r_0, r_1))
\]

The key here is that the error incurred during the CUR decomposition, \( \epsilon \) 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 \( \mathcal{K}(x, y) \).

**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
\[
\mathcal{K}(X, Y) \approx \mathcal{K}(X, \hat{Y}) \mathcal{K}(\hat{X}, \hat{Y})^{-1} \mathcal{K}(\hat{X}, Y)
\]

seems to be numerically challenging to compute. Indeed, as we showed in the previous section, we can only really expect \( \|K_w^{-1}\|_2 = \mathcal{O}(\epsilon^{-1}) \) which indicates that, roughly,
\[
\kappa(\mathcal{K}(\hat{X}, \hat{Y})) = \mathcal{O}\left(\frac{1}{\epsilon}\right)
\]
Eq. 10 illustrates the 4 even with the basis functions them independently. In particular, one can look at the quality of the interpolation of easy to see that the final factorization is accurate. Indeed, 

$$T(x) \text{ evaluated at } T(x)$$

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

This interpretation is interesting as it allows 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 $$\hat{S}$$ and $$\hat{T}$$. Indeed, if this is accurate, it is easy to see that the final factorization is accurate. Indeed,

$$\mathcal{K}(x, y) \approx \sum_{k=1}^{r} \sigma_k u_k(x) v_k(y)$$

$$\approx \sum_{k=1}^{r} \sigma_k (\hat{S}(x, \hat{X}) u_k(\hat{X}))(\hat{T}(y, \hat{Y}) v_k(\hat{Y}))^\top$$

$$= \hat{S}(x, \hat{X}) \left( \sum_{k=1}^{r} \sigma_k u_k(\hat{X}) v_k(\hat{Y})^\top \right) \hat{T}(y, \hat{Y})^\top$$

$$\approx \hat{S}(x, \hat{X}) \mathcal{K}(\hat{X}, y) \mathcal{K}(\hat{X}, \hat{Y})^{-1} \mathcal{K}(\hat{X}, y)$$

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

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

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

Fig. 1 illustrates the 4th Lagrange basis function in $$x$$, i.e., $$\hat{S}(x, \hat{X})$$ and the classical Lagrange polynomial basis function associated with the same set $$\hat{X}$$. We see that they are both 1 at $$\hat{X}_4$$ and 0 at the other. However, $$\hat{S}(x, \hat{X})_4$$ is much more stable and small 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.
Fig. 2 shows the effect of interpolating \( u_r(x) \) using \( \tilde{S}(x, \hat{X}) \) as well as using the usual polynomial interpolation at the nodes \( \hat{X} \). We see that \( \tilde{S}(x, \hat{X}) \) interpolates very well \( u_r(x) \), showing indeed that we implicitly build an accurate interpolant, even on the last (least smooth) eigenfunctions. The usual polynomial interpolation fails to capture any feature of \( u_r \) on the other hand. Note that we could have reached a similar accuracy using interpolation at Chebyshev nodes but only by using many more interpolation nodes.

Finally, Fig. 3 shows how well we approximate the various \( r \) eigenfunctions. As one can see, interpolation is very accurate on \( u_1(x) \), but grows for less smooth eigenfunctions. The growth is, roughly, similar to the growth of \( \frac{1}{\sigma_i} \). Notice how this is just enough so that the resulting factorization is accurate:

\[
\tilde{S}(x, \hat{X}), \mathcal{K}(\hat{X}, y) = \sum_{s=1}^{r} \sigma_s \tilde{S}(x, \hat{X}) u_s(\hat{X}) v_s(y) + \mathcal{O}(\epsilon)
\]

\[
= \sum_{s=1}^{r} \sigma_s \left( u_s(x) + \mathcal{O}\left( \frac{\epsilon}{\sigma_s} \right) \right) v_s(y) + \mathcal{O}(\epsilon)
\]

\[
= \sum_{s=1}^{r} \sigma_s u_s(x) v_s(y) + \sum_{s=1}^{r} \mathcal{O}(\epsilon) v_s(y) + \mathcal{O}(\epsilon)
\]

\[
= \mathcal{K}(x, y) + \mathcal{O}(\epsilon)
\]

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 \( \mathcal{K} \) with just enough accuracy so that the resulting interpolation is accurate up to the desired accuracy.

Fig. 1: 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.

5. **Numerical experiments.** The experiments are done using Julia [4] and the code is sequential. The code can be downloaded from [https://people.stanford.edu/](https://people.stanford.edu/)
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.

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.

5.1. Simple geometries. We begin this section with an elementary problem, as depicted on Fig. 4b. In this problem, we consider the usual kernel $K(x, y) = \frac{1}{\|x-y\|_2}$ where $x, y \in \mathbb{R}^2$. $X$ and $Y$ are two squares of side of length 1, centered at $(0.5, 0.5)$ and $(2.5, 2.5)$ respectively. Finally, $X$ and $Y$ are two uniform meshes of $50 \times 50$ mesh points each, i.e., $n = 2500$. 

Consider Fig. 4a. The \( r_0 \) line indicates the rank \( (r_0 = \min(\|X\|,\|Y\|)) \) of the low-rank expansion through interpolation. The Chebyshev grid is obtained using a fast heuristic where one get the number of node in each dimension (i.e., \( x_1, x_2, y_1, y_2 \)) independently by doing one-dimensional interpolation and using, as a reference point, the centers of each squares. Furthermore, the accuracy \( \delta \) of such interpolation is merely a fraction of the required tolerance. The determination of this fraction is problem dependent but critical to ensure decent performances in practical applications (i.e., to minimize the work required in building \( X, Y \) and \( \mathcal{K}(X, Y) \)). The \( r_1 \) line corresponds to the rank obtained after the RRQR over \( \mathcal{K}(X, Y) \) and its transpose, i.e., it is the rank of the final approximation

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

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

\[
\mathcal{K}(X, Y) = USV^T
\]

at the appropriate singular value, as to ensure

\[
\|\mathcal{K}(X, Y) - \mathcal{K}(\hat{X}, \hat{Y})\mathcal{K}(\hat{X}, Y)\|_F \approx \epsilon \|\mathcal{K}(X, Y)\|_F
\]

Similarly, “Rank 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 as to illustrate that \( r_1 \) is close to the optimal value.

The conclusion regarding Fig. 4a 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 \( \mathcal{K}(\tilde{X}, \tilde{Y}) \) allows us to find subsets \( \tilde{X} \subset X \) and \( \tilde{Y} \subset Y \) that are enough to represent well \( \mathcal{K}(X, Y) \), and the final rank \( r_1 \) is nearly optimal compared to the SVD-rank \( r \). We also see that the rank a blind RRQR over \( \mathcal{K}(X, Y) \) is higher than the SVD-rank and usually closer—if not identical—to \( r_1 \).

We want to re-emphasize that, in practice, the error of the sets \( \tilde{X}, \tilde{Y} \) —i.e., the error of the polynomial interpolation based on \( \tilde{X} \times \tilde{Y} \) — can be larger than the required tolerance. If they are large enough, the compressed sets \( \tilde{X}, \tilde{Y} \) will contain enough information so as to properly interpolate \( \mathcal{K} \) 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 motivation of this work), even though the final rank is small.

As a sanity check, Fig. 4c gives the relative error measured in the Frobenius norm

\[
\frac{\|\mathcal{K}(X, Y) - \mathcal{K}(X, \hat{Y})\mathcal{K}(\hat{X}, \hat{Y})^{-1}\mathcal{K}(\hat{X}, Y)\|_F}{\|\mathcal{K}(X, Y)\|_F}
\]

between \( \mathcal{K}(X, Y) \) and its interpolation as a function of the tolerance \( \epsilon \).

We see that both lines are almost next to each other, meaning our approximation indeed reaches the required tolerance. This is important as it means that one can effectively control the accuracy.

Finally, Fig. 4b also shows the resulting \( \tilde{X} \) and \( \tilde{Y} \). 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.

\(^1\)Choosing the Frobenius norm is not critical—very similar results are obtained in the 2-norm for instance.
We then consider results for the same Laplacian kernel $\mathcal{K}(x, y) = \|x - y\|_2^{-1}$ between two plates in 3D (Fig. 5b). We observe overall very similar results as for the previous case on Fig. 5a, where the initial rank $r_0$ is significantly decreased to $r_1$ while keeping the resulting accuracy close to the required tolerance as Fig. 5c shows. Finally, one can see on Fig. 5b 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 [3] and ‘Random CUR’ where one selects, at random, pivots $\tilde{X}$ and $\tilde{Y}$ and builds a factorization

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

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 ($\mathcal{K}(x, y) = \|x - y\|_2^{-1}$). 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 $\mathcal{F}(X, Y)$;
2. Given $r_1$, build the random CUR factorization by sampling uniformly at random $r_1$ points from $X$ and $Y$ to build $\tilde{X}$, $\tilde{Y}$. Then, compute its relative error with $\mathcal{F}(X, Y)$.

We then do so 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 to study the variance of the error and to collect some statistics.

Fig. 6 gives the resulting errors in relative Frobenius norm for the 3 algorithm 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 outliers depicted using large dots. We observe that the $(\tilde{X}, \tilde{Y})$ 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.

![Graph](image)

Fig. 6: Comparison between different algorithms: Chebyshev-based SI, ACA and purely random CUR decomposition. We consider two 2D squares of sides 1 with a variable distance from each other; for each distance, we run Chebyshev-based SI and find the smallest sets $X, Y$ of rank $r_0$ leading to a factorization using $\hat{X}, \hat{Y}$ 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 $100 \times 100$ points grid in each square.

5.3. Stability guarantees provided by RRQR.

5.4. The need for weights.

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:

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

So the total complexity of building the three factor is $O((m+n)r_1)$ kernel evaluations. If $m = n$ and $r_1 \approx r$, the total complexity is

$$O((m+n)r_1) \approx O(nr)$$

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


The geometry.

\[ L_y = 0.9 L_x, \]

each domain \( X_i, Y_i \) has 50 points uniformly distributed (hence, the weights are uniform) on an arc of angle \( \pi/4 \).

\( \hat{X}, \hat{Y} \) are \( r_1 \) points subsampled from \( X, Y \) using Algorithm 1.

Error

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

Error

(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 blue curve on the right panel, which uses weights, has much improved accuracy.

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 \( w_1 = \mathcal{K}(\hat{X}, Y)f(Y) \)
- \( \mathcal{O}(r_1^2) \) flops for computing \( w_2 = \mathcal{K}(\hat{X}, \hat{Y})^{-1}w_1 \) assuming a factorization of \( \mathcal{K}(\hat{X}, \hat{Y}) \) has already been computed
- \( \mathcal{O}(mr_1) \) flops for computing \( w_3 = \mathcal{K}(X, \hat{Y})w_2 \)

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, Fig. 9a 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 \( \mathcal{K}(X, Y) \) 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 \( \mathcal{O}(n^2) \) complexities clearly appear, and our algorithm scales much better than the naive one (or, really, that 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.

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

Fig. 9c 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 \( \epsilon \) fixed, we change the geometry and observe the resulting rank. This is done by moving the top-right square (see Fig. 4b) 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.

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 insensitive to the mesh point distribution, leading to more stable sets of “pivots” than Random Sampling or ACA.

Appendix A. Proofs of the theorems.

Lemma 1. This bound on the Lagrange basis is a classical result related to the growth of the Lebesgue constant in polynomial interpolation. For \( \overline{\pi} \) Chebyshev nodes of the first kind on \([-1,1]\) and the associated Lagrange basis functions \( \ell_1, \ldots, \ell_{\overline{\pi}} \) we have the following result [22, equation 13]

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

This implies that in one dimension,

\[
\|S(x, \overline{X})\|_2 \leq \|S(x, \overline{X})\|_1 = \mathcal{O}(\log(\overline{\pi}))
\]
Going from one to \(d\) dimensions can be done using Kronecker products. Indeed, for \(x \in \mathbb{R}^d\),

\[
S(x, \vec{X}) = S(x_1, \vec{X}_1) \otimes \cdots \otimes S(x_d, \vec{X}_d)
\]

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

\[
\|S(x, \vec{X})\|_2 = \|S(x_1, \vec{X}_1) \otimes \cdots \otimes S(x_d, \vec{X}_d)\|_2 = \prod_{i=1}^d \|S(x_i, \vec{X}_i)\|_2 = \prod_{i=1}^d O(\log m_i)
\]
This implies, using a fairly loose bound,
\[ \|S(x, X)\|_2 = O(\log(|X|)^d) \]
The same argument can be done for \( T(y, Y) \).
In 1D, the weights are
\[ w_k = \frac{\pi}{m} \sin \left( \frac{2k - 1}{2m} \pi \right) \]
for \( k = 1, \ldots, m \). Obviously, \( w_k > 0 \). Clearly, \( w_k < \frac{\pi}{m} \).
Also, the minimum being reached at \( k = 1 \) or \( k = m \),
\[ w_k \geq \frac{\pi}{m} \sin \left( \frac{\pi}{2m} \right) > \frac{2\pi}{m^2 \pi} = \frac{\pi}{m^2} \]
Since the nodes in \( d \) dimensions are products of the nodes in 1D, it follows that
\[ \| \text{diag}(W_X) \|_2 \leq \frac{\pi^d}{m} \]
\[ \| \text{diag}(W_X)^{-1} \|_2 \leq \frac{m^d}{\pi^d} \]
The result follows.

**Lemma 3.** We show the result for the second equation. This requires using, consecutively, the interpolation result and the CUR decomposition one. First, one can write from **Lemma 1** and the interpolation,
\[ \mathcal{X}(\hat{X}, \hat{Y})^{-1} \mathcal{X}(\hat{X}, y) = \mathcal{X}(\hat{X}, \hat{Y})^{-1} \left[ \mathcal{X}(\hat{X}, \hat{Y})T(y, \bar{Y})^\top + \text{E}_{\text{INT}}(\hat{X}, y) \right] \]
Then, introducing the weight matrices and applying **Lemma 2** on the interpolation matrix,
\[ \mathcal{X}(\hat{X}, \bar{Y}) = \]
\[ = \text{diag}(\hat{W}_X)^{-1/2} \text{diag}(\hat{W}_Y)^{1/2} \mathcal{X}(\hat{X}, \hat{Y}) \text{diag}(\hat{W}_Y)^{-1/2} \text{diag}(\hat{W}_Y)^{-1/2} \]
\[ = \text{diag}(\hat{W}_X)^{-1/2} \left\{ \text{diag}(\hat{W}_X)^{1/2} \mathcal{X}(\hat{X}, \hat{Y}) \text{diag}(\hat{W}_Y)^{1/2} \left[ I - \bar{T}^\top \right] \right. \]
\[ + \left. \text{E}_{\text{QR}}(\hat{X}, \bar{Y}) \right\} \text{diag}(\hat{W}_Y)^{-1/2} \]
Finally, combining and distributing all the factors gives us
\[ \mathcal{X}(\hat{X}, \hat{Y})^{-1} \mathcal{X}(\hat{X}, y) = \text{diag}(\hat{W}_Y) \left[ I - \bar{T}^\top \right] \text{diag}(\hat{W}_Y)^{-1/2} T(y, \bar{Y})^\top \]
\[ + \mathcal{X}(\hat{X}, \hat{Y})^{-1} \text{diag}(\hat{W}_X)^{-1/2} \text{E}_{\text{QR}}(\hat{X}, \bar{Y}) \text{diag}(\hat{W}_Y)^{-1/2} T(y, \bar{Y})^\top \]
\[ + \mathcal{X}(\hat{X}, \hat{Y})^{-1} \text{E}_{\text{INT}}(\hat{X}, y) \]
Here, we can bound all terms:
- For the first term, **Lemma 1** and **Lemma 2** show that the expression is bounded by a polynomial;
We conclude that there exists a polynomial

The last term can be bounded in a similar way using

$$E_{\text{INT}}(x, y) = O(\delta) \ll O(\epsilon)$$

We conclude that there exists a polynomial \( q \) such that

$$\|K^{-1}(x, y)\|_2 = O(q(r_0, r_1))$$

The proof is similar in \( x \).

**Theorem 2.** Combining interpolation and CUR decomposition results one can write

$$K(x, y) = S(x, X)K(X, Y)T(y, Y) + E_{\text{INT}}(x, y)$$

$$= S_w(x, X)K_w(X, Y)T_w(y, Y) + E_{\text{INT}}(x, y)$$

$$= S_w(x, X)\left[ I - \tilde{F}^\top \right] + E_{QR}(X, Y) + E_{\text{INT}}(x, y)$$

$$= S_w(x, X)K_w(X, Y) + E_{QR}(X, Y) + E_{\text{INT}}(x, y)$$

Distributing everything, factoring the weights matrices and simplifying, we obtain the following, where we indicate the bounds on each term on the right,

$$K(x, y) = K(x, \hat{Y})K(\hat{X}, \hat{Y})^{-1}K(\hat{X}, y)$$

$$+ E_{\text{INT}}(x, \hat{Y})K(\hat{X}, \hat{Y})^{-1}K(\hat{X}, y)$$

$$+ K(x, \hat{Y})K(\hat{X}, \hat{Y})^{-1}E_{\text{INT}}(\hat{X}, y)$$

$$+ E_{\text{INT}}(x, \hat{Y})K(\hat{X}, \hat{Y})^{-1}E_{\text{INT}}(\hat{X}, y)$$

$$+ S(x, X)\text{diag}(W_X)^{-1/2}E_{QR}(X, \hat{Y})\text{diag}(\hat{W}_Y)^{-1/2}$$

$$K(\hat{X}, \hat{Y})^{-1}K(\hat{X}, \hat{Y})T(y, Y)^\top$$

$$+ S(x, X)K(\hat{X}, \hat{Y})K(\hat{X}, \hat{Y})^{-1}\text{diag}(\hat{W}_X)^{-1/2}$$

$$O(\epsilon(\log r_0)^{2d}q(r_0, r_1)r_0^2$$
\[
E_{QR}(\hat{X}, \hat{Y}) \text{diag}(\hat{W}_Y)^{-1/2}T(y, \hat{Y})^\top + S(x, X) \text{diag}(\hat{W}_X)^{-1/2}E_{QR}(X, \hat{Y})
\]
\[
\text{diag}(\hat{W}_Y)^{-1/2} \epsilon(\hat{X}, \hat{Y})^{-1} \text{diag}(\hat{W}_X)^{-1/2}E_{QR}(X, \hat{Y})
\]
\[
\text{diag}(\hat{W}_Y)^{-1/2}T(y, \hat{Y})^\top + E_{\text{INT}}(x, y)
\]
\[
O(\epsilon(\log r_0)^{2d} q(r_0, r_1)^2)
\]
This concludes the proof.  

REFERENCES

[1] J. Barnes and P. Hut, A hierarchical o(n log n) force-calculation algorithm, nature, 324 (1986), pp. 446–449.
[2] M. Bebendorf, Approximation of boundary element matrices, Numerische Mathematik, 86 (2000), pp. 565–589.
[3] M. Bebendorf and S. Rjasanow, Adaptive low-rank approximation of collocation matrices, Computing, 70 (2003), pp. 1–24.
[4] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, Julia: A fresh approach to numerical computing, SIAM Review, 59 (2017), pp. 65–98.
[5] S. Börm and L. Grasedyck, Low-rank approximation of integral operators by interpolation, Computing, 72 (2004), pp. 325–332.
[6] S. Börm and L. Grasedyck, Hybrid cross approximation of integral operators, Numerische Mathematik, 101 (2005), pp. 221–249.
[7] H. Cheng, Z. Gimbutas, P.-G. Martinsson, and V. Rokhlin, On the compression of low rank matrices, SIAM Journal on Scientific Computing, 26 (2005), pp. 1389–1404.
[8] E. Corona, A. Rahimian, and D. Zorin, A tensor-train accelerated solver for integral equations in complex geometries, Journal of Computational Physics, 334 (2017), pp. 145–169.
[9] W. Fong and E. Darve, The black-box fast multipole method, Journal of Computational Physics, 229 (2009), pp. 8712–8725.
[10] G. H. Golub and C. F. Van Loan, Matrix computations, vol. 3, JHU Press, 2012.
[11] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, Journal of computational physics, 73 (1987), pp. 325–348.
[12] M. Gu and S. C. Eisenstat, Efficient algorithms for computing a strong rank-revealing qr factorization, SIAM Journal on Scientific Computing, 17 (1996), pp. 848–869.
[13] A. Guven, Quantitative Perturbation Theory for Compact Operators on a Hilbert Space, PhD thesis, Queen Mary University of London, 2016.
[14] K. Hackbusch, A sparse H-matrix arithmetic. part ii: application to multi-dimensional problems, Computing, 64 (2000), pp. 21–47.
[15] W. Hackbusch, A sparse matrix arithmetic based on H-matrices. part i: Introduction to H-matrices, Computing, 62 (1999), pp. 89–108.
[16] W. Hackbusch and S. Börm, Data-sparse approximation by adaptive H 2-matrices, Computing, 69 (2002), pp. 1–35.
[17] W. Hackbusch and Z. P. Nowak, On the fast matrix multiplication in the boundary element method by panel clustering, Numerische Mathematik, 54 (1989), pp. 463–491.
[18] N. Halko, P.-G. Martinsson, and J. A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM review, 53 (2011), pp. 217–288.
[19] J. J. Higgin, Accuracy and stability of numerical algorithms, vol. 80, Siam, 2002.
[20] K. L. Ho and S. Olver, LowRankApprox.jl: Fast low-rank matrix approximation in Julia, May 2018, http://dx.doi.org/10.5281/zenodo.1254148, https://doi.org/10.5281/zenodo.1254148.
[21] K. L. Ho and L. Ying, Hierarchical interpolative factorization for elliptic operators: integral equations, Communications on Pure and Applied Mathematics, (2015).
[22] B. A. Ibrahimoglu, Lebesgue functions and lebesgue constants in polynomial interpolation, Journal of Inequalities and Applications, 2016 (2016), p. 93.
[23] M. W. Mahoney and P. Drineas, *Cur matrix decompositions for improved data analysis*, Proceedings of the National Academy of Sciences, 106 (2009), pp. 697–702.

[24] K. B. Petersen, M. S. Pedersen, et al., *The matrix cookbook*, Technical University of Denmark, 7 (2008), p. 510.

[25] M. Reed and B. Simon, *Methods of modern mathematical physics. vol. 1. Functional analysis*, Academic, 1980.

[26] M. Renardy and R. C. Rogers, *An introduction to partial differential equations*, vol. 13, Springer Science & Business Media, 2006.

[27] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, Journal of computational physics, 60 (1985), pp. 187–207.

[28] E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method*, Computing, 64 (2000), pp. 367–380.

[29] Z. Wu and T. Alkhalifah, *The optimized expansion based low-rank method for wavefield extrapolation*, Geophysics, 79 (2014), pp. T51–T60.

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