Received: 6 December 2022 | Revised: 10 May 2023 | Accepted: 11 June 2023

DOI: 10.1002/nla.2519

RESEARCH ARTICLE

WILEY

# Data-driven linear complexity low-rank approximation of general kernel matrices: A geometric approach

**Difeng Cai**[1] | **Edmond Chow**[2] | **Yuanzhe Xi**[1]

[1]Department of Mathematics, Emory University, Atlanta, Georgia, USA

[2]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA

**Correspondence**
Difeng Cai, Department of Mathematics, Emory University, Atlanta, GA 30322, USA.
Email: dcai7@emory.edu

**Funding information**
NSF OAC, Grant/Award Numbers: 2003683, 2003720

**Summary**

A general, *rectangular* kernel matrix may be defined as $K_{ij} = \kappa(x_i, y_j)$ where $\kappa(x, y)$ is a kernel function and where $X = \{x_i\}_{i=1}^m$ and $Y = \{y_i\}_{i=1}^n$ are two sets of points. In this paper, we seek a low-rank approximation to a kernel matrix where the sets of points $X$ and $Y$ are large and are arbitrarily distributed, such as away from each other, "intermingled", identical, and so forth. Such rectangular kernel matrices may arise, for example, in Gaussian process regression where $X$ corresponds to the training data and $Y$ corresponds to the test data. In this case, the points are often high-dimensional. Since the point sets are large, we must exploit the fact that the matrix arises from a kernel function, and avoid forming the matrix, and thus ruling out most algebraic techniques. In particular, we seek methods that can scale linearly or nearly linearly with respect to the size of data for a fixed approximation rank. The main idea in this paper is to *geometrically* select appropriate subsets of points to construct a low rank approximation. An analysis in this paper guides how this selection should be performed.

**KEYWORDS**

error analysis, hierarchical matrix, high-dimensional data, kernel matrix, low-rank approximation

## 1 | INTRODUCTION

Given a function $\kappa(x, y)$ and two sets of points $X = \{x_i\}_{i=1}^m$ and $Y = \{y_i\}_{i=1}^n$, the $m$-by-$n$ matrix with entries

$$K_{ij} = \kappa(x_i, y_j), \quad x_i \in X, \quad y_j \in Y \tag{1}$$

and denoted by $K_{XY}$ is called a *kernel matrix* and $\kappa(x, y)$ is called a kernel function. Kernel matrices associated with various kernel functions arise in diverse computations such as those involving integral equations,[1-5] $N$-body simulations,[6,7] Gaussian processes,[8,9] and others.[10-14]

One frequently encounters the problem of finding a low-rank factorization, exactly or approximately, of a kernel matrix. We first note that algebraic techniques such as the singular value decomposition and some pseudoskeleton[15-18] and CUR decompositions[19,20] do not take advantage of the fact that a matrix is a *kernel* matrix. We further note that when the kernel function $\kappa(x, y)$ is smooth (but possibly singular at $x = y$) and the datasets $X, Y$ are well-separated, then the corresponding kernel matrix $K_{XY}$ generally has low numerical rank and there exists a variety of efficient methods for finding the low-rank approximation (e.g., degenerate approximations of the kernel function[4,7,21-25] and proxy point methods[26,27]).

In this paper, we seek a low-rank approximation to a kernel matrix where the sets of points $X$ and $Y$ are large and are arbitrarily distributed, such as away from each other, "intermingled", identical, and so forth. Since the point sets are large, we must exploit the fact that the matrix arises from a kernel function, and avoid forming the matrix, and thus ruling out most algebraic techniques. In particular, we seek methods that can scale linearly or nearly linearly for a fixed rank. Such kernel matrices arise, for example, in Gaussian process regression where $X$ corresponds to the training data and $Y$ corresponds to the test data. In this case, the points are often high-dimensional, which also rules out the use of any existing methods (e.g., degenerate approximations and proxy point methods) that are limited by the curse of dimensionality.

An existing method called adaptive cross approximation (ACA)[28,29] is often suitable for our case. ACA scales linearly with the number of points. ACA corresponds to a pivoted partial LU factorization and only needs to compute matrix elements used in the partial factorization. However, ACA may fail in some circumstances since it does not perform full pivoting.[30,31] We will numerically compare our proposed method to ACA later in this paper.

The main idea in this paper is to *geometrically* select a subset of points $S_1$ in $X$ and/or a subset of points $S_2$ in $Y$ to construct a low rank approximation. An analysis in this paper guides how this selection should be performed.

We analyze the use of these subsets of points to construct two forms of low-rank factorizations. The first is a two-sided form:

$$K_{XY} \approx K_{XS_2} K^{+}_{S_1 S_2} K_{S_1 Y}, \quad S_1 \subseteq X, \quad S_2 \subseteq Y, \tag{2}$$

where $K^{+}_{S_1 S_2}$ denotes the pseudoinverse of $K_{S_1 S_2}$. This form is a CUR decomposition, except that we will treat $K_{XY}$ as a kernel matrix. Note that this form is similar to that of a Nyström factorization, except that a Nyström factorization[32] expects the kernel matrix to be symmetric, with $Y = X$, since eigenvalues of the kernel matrix are implicitly being approximated in the Nyström factorization. The matrix $K_{XY}$ in (2) is rectangular in general.

The second form of low-rank factorization that we study is the one-sided form of the interpolative decomposition:[33]

$$K_{XY} \approx U K_{\mathcal{I} Y}, \quad U = P \begin{bmatrix} I \\ G \end{bmatrix}, \tag{3}$$

where $\mathcal{I} \subseteq X$, $P$ is a permutation matrix, $I$ is an identity matrix and $G$ is a general dense matrix. This form can be computed algebraically using the strong rank-revealing QR factorization[34] with the property that the max-norm of $G$ is bounded by a prescribed constant larger than 1. However, this algebraic factorization requires the entire matrix $K_{XY}$ to be formed explicitly.

Instead, it is common to algebraically compute the interpolative decomposition of the smaller matrix

$$K_{XS_2} \approx U K_{\mathcal{I} S_2}, \quad U = P \begin{bmatrix} I \\ G \end{bmatrix}, \tag{4}$$

where $S_2 \subseteq Y$ or $S_2$ is an entirely different set of points altogether, and then use $U$ and $\mathcal{I}$ computed in (4) for the approximation (3). Examples of this approach can be found in References 26,30,31. In these approaches, the choice of $S_2$ is made analytically (e.g., Chebyshev points[30,31] or proxy surface points[26]) or algebraically (e.g., ACA).[30] In this paper, for the one-sided approximation (3), we will analyze a *geometric* choice of the subset $S_2$. After $S_2$ is chosen, the subset $\mathcal{I}$ is selected by the algebraic interpolative decomposition via strong rank-revealing QR factorization.

Low-rank methods based on subset selection are useful in improving the scalability of Gaussian process, often under the name of "sparse Gaussian process"(cf. References 35-37), where "sparse" refers to the fact that the selected subsets, for example, $S_1$, $S_2$ in (2), are much smaller than (thus sparsely distributed in) the original data sets. Thus one application of the paper is the design of scalable Gaussian process.

This paper will show that the low-rank approximation error in the maximum norm depends on the quantities $\delta_{X,S_1}$ and/or $\delta_{Y,S_2}$, where

$$\delta_{Z,S} := \max_{x \in Z} \operatorname{dist}(x, S)$$

measures the closeness between $Z$ and $S$. In order for $\delta_{X,S_1}$ (or $\delta_{Y,S_2}$) to be small, points in $S_1$ (or $S_2$) should be close to as many points in $X$ (or $Y$) as possible. This implies that selecting sample points that are evenly distributed over the entire dataset can yield better approximations than, for example, choosing clustered points in small regions that fail to capture the geometry of the entire dataset. A similar geometric selection can be used in a version of skeletonized interpolation[38] but has only been studied in the case of well-separated sets of points.

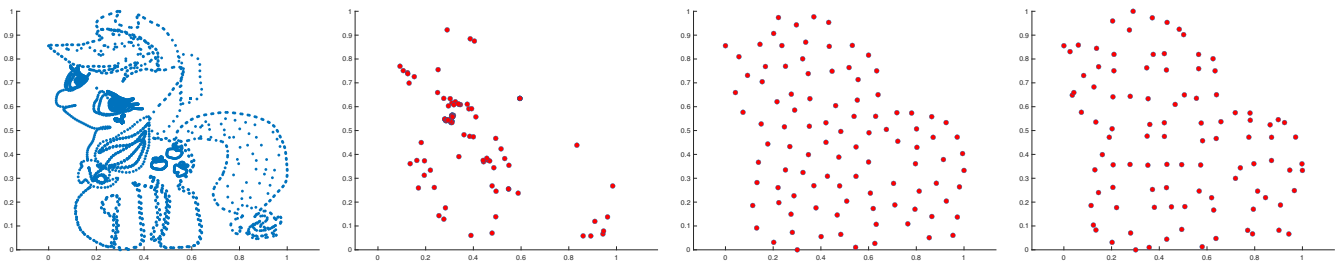
Several known methods can be used to select $O(1)$ sample points that are evenly distributed over a dataset with a complexity that scales *linearly* with the size of the dataset. For example, *farthest point sampling* (FPS)[39] constructs a subset $S$ of $X$ by first initializing $S$ with one point and then sequentially adding the point in $X$ not in $S$ that is farthest from the current points in $S$. The complexity for selecting $r$ samples from $n$ points in $\mathbb{R}^d$ is $O(dr^2n)$. FPS produces highly evenly distributed samples and is often used in mesh generation,[40] computer graphics,[41] and so forth, but primarily where the data are at most three dimensional. It has not previously been used for the low-rank compression of matrices or applied to high dimensional datasets. Computationally, for high dimensional datasets, FPS can be potentially slow in practice due to its sequential nature. One can combine FPS with uniform random sampling for faster speed, for example, by generating approximately 20% of samples using FPS and 80% using uniform random sampling. As will be shown in Section 4.2, the resulting *mixed method* tends to yield an approximation that is less accurate than FPS and more accurate than random sampling.

Another method for selecting evenly distributed sample points is the *anchor net method*.[42] This method was proposed for the efficient generation of landmark points for Nyström approximations such that the resulting approximation is accurate and numerically stable. It leverages discrepancy theory to generate evenly-spaced samples and was shown in[42] to achieve better accuracy and robustness than uniform random sampling and $k$-means clustering for low-rank approximations. The anchor net method has the optimal complexity $O(drn)$ for selecting $r$ points from $n$ points in $\mathbb{R}^d$ and is efficient for a wide range of problems from low to high dimensions. However, the anchor net method has only been used for approximating symmetric kernel matrices and its performance for approximating general rectangular kernel matrices is as yet unknown.

Figure 1 shows the 100 samples obtained from FPS and the anchor net method for a highly irregular dataset in two dimensions. Results for uniform random sampling is also shown, which does not generally produce a uniform distribution of points over the data.

In summary, we seek a linear or nearly linear complexity low-rank factorization approach for kernel matrices where the points may be intermingled and the points may be high-dimensional. Some low-rank approximation techniques are matrix-based (e.g., ACA) and don't rely on knowing the specific kernel function or sets of points, except for assuming that the kernel function is smooth and gives rise to a kernel matrix $K_{XY}$ that is low rank. Other techniques only need knowledge of the kernel function and bounding boxes for the sets of points, and do not depend on the points themselves when selecting the set $S_2$ in (4), for example. The method we propose is based on the sets $X$ and $Y$ and is independent of the kernel function. We thus call our method a *data-driven* method. By choosing $S_2$ to be existing points rather than a new set of points that sample possibly high-dimensional space, the data-driven method is not limited by the curse of dimensionality.

Our proposed method relies on the geometric selection of the subsets $S_1 \subseteq X$ and/or $S_2 \subseteq Y$. We address the following questions: (1) how does the data selection affect the low-rank approximation error? (2) given two subsets with equal numbers of points, how can one tell which one leads to a more accurate low-rank approximation? (3) how can one perform the desired data selection efficiently?



(a) Pony dataset (b) Uniform random sampling (c) Farthest point sampling (d) Anchor net method

**FIGURE 1** Different geometric selection schemes for the Pony dataset.

The rest of the paper is organized as follows. Section 2 proposes the data-driven approach for efficiently computing the *two-sided* factorization (2). Section 3 similarly considers the *one-sided* factorization (3). We will show that the one-sided factorization is more stable than the two-sided factorization. The two-sided factorization, however, is slightly cheaper to compute than the one-sided factorization. The results of numerical experiments are presented in Section 4, and a conclusion given in Section 5. Unless otherwise stated, all norms used in this paper are the 2-norm, denoted by $\|\cdot\|$. The Euclidean distance between $x, y \in \mathbb{R}^d$ is denoted by $|x - y|$.

## 2 | TWO-SIDED LOW-RANK KERNEL MATRIX APPROXIMATION

This section analyzes the data-driven geometric approach for the two-sided low-rank approximation (2).

### 2.1 | Algorithm

The two-sided factorization (2) can be computed immediately once the subsets $S_1 \subseteq X$ and $S_2 \subseteq Y$ are determined. The subsets can be computed in linear time with suitable geometric selection schemes. The full algorithm is given in Algorithm 1. Depending on the specific geometric selection scheme, the total complexity of Algorithm 1 is $O(dr(m+n))$ for uniform random sampling and the anchor net method, or $O(dr^2(m+n))$ for farthest point sampling, where $r = \max(r_1, r_2)$ denotes the maximum sample size. The choice of subsets has a strong impact on the low-rank approximation accuracy, robustness of the algorithm, as well as numerical stability, and thus the subset has to be chosen judiciously. Theoretical guidance on geometric selection is provided in Section 2.2 via analyzing the approximation error of the two-sided factorization. Experiments in Section 4.2 show that different geometric selections can yield dramatically different results for approximating the kernel matrix, with FPS and the anchor net method yielding the best results, which is consistent with our analysis.

**Algorithm 1.** Data-driven two-sided compression of $K_{XY}$ with two sets of points $X, Y$

*Input:* Datasets $X = \{x_1, \dots, x_m\}$, $Y = \{y_1, \dots, y_n\} \subset \mathbb{R}^d$, kernel function $\kappa$, numbers of sample points $r_1, r_2$ for $X, Y$, respectively

*Output:* Approximation $K_{XY} \approx K_{XS_2} K^{+}_{S_1S_2} K_{S_1Y}$ with $\mathrm{card}(S_1) = r_1$, $\mathrm{card}(S_2) = r_2$

Apply a linear complexity geometric selection algorithm to $X$ and $Y$ to generate $r_1$ and $r_2$ samples $S_1 \subseteq X$ and $S_2 \subseteq Y$, respectively

Return $K_{XS_2}, K_{S_1S_2}, K_{S_1Y}$

### 2.2 | Error analysis for two-sided approximation

The goal of this section is to derive an error estimate of the two-sided approximation that is able to provide a straightforward geometric understanding of how the subsets $S_1, S_2$ affect the approximation accuracy. This analysis is independent of how the subsets $S_1, S_2$ are selected in Algorithm 1. To prepare for the derivation of the geometric estimates, in Section 2.2.1, we derive error bounds involving only submatrices of $K_{XY}$. The geometric estimates are presented in Section 2.2.2.

#### 2.2.1 | Algebraic preparation

In order to estimate for the approximation error of (2) for arbitrary subsets $S_1 \subseteq X$ and $S_2 \subseteq Y$, we first review one lemma from [ 42, Lemma 3.1], which is stated below.

**Lemma 1.** *Assume $A$ is an m-by-n matrix, $\alpha, \hat{\alpha}$ are m-by-1 vectors and $\beta, \hat{\beta}$ are n-by-1 vectors. Define $\epsilon_1(u) := \|\hat{\alpha} - \alpha\|$ and $\epsilon_2 := \|\hat{\beta} - \beta\|$. Then*

$$|\hat{\alpha}^T A\hat{\beta} - \alpha^T A\beta| \le \|\alpha^T A\| \cdot \epsilon_2 + \|A\beta\| \cdot \epsilon_1(u) + \|A\| \cdot \epsilon_1(u)\epsilon_2. \tag{5}$$

In the next theorem, we derive the estimate for the entrywise approximation error of (2) at an arbitrary pair of points $(x, y)$. This can be viewed as an error estimate for the "*algebraic*" separable approximation to the kernel function $\kappa(x, y)$.

**Theorem 1.** *Consider finite sets $X, Y \subset \mathbb{R}^d$ and a kernel function $\kappa(x, y)$ defined on $X \times Y$. For any non-empty subsets $S_1 \subseteq X$ and $S_2 \subseteq Y$, the entrywise error of the approximation in (2) satisfies*

$$|\kappa(x,y) - K_{xS_2}K^+_{S_1S_2}K_{S_1y}| \le \min_{\substack{u\in S_1\\ v\in S_2}} \left( |\kappa(x,y) - \kappa(u,v)| + \epsilon_1(u) + \epsilon_2(v) + \|K^+_{S_1S_2}\|\epsilon_1(u)\epsilon_2(v) \right), \tag{6}$$

*where $\epsilon_1(u) = \|K_{xS_2} - K_{uS_2}\|$ and $\epsilon_2(v) = \|K_{S_1y} - K_{S_1v}\|$.*

*Proof.* Since $K_{S_1S_2}K^+_{S_1S_2}K_{S_1S_2} = K_{S_1S_2}$, we have $\forall u \in S_1, v \in S_2$

$$\kappa(u,v) = K_{uS_2}K^+_{S_1S_2}K_{S_1v}. \tag{7}$$

For any $x \in X$, $y \in Y$, $u \in S_1$, $v \in S_2$, define the column vectors

$$\alpha := K^T_{uS_2}, \quad \hat{\alpha} := K^T_{xS_2}, \quad \beta := K_{S_1v}, \quad \hat{\beta} := K_{S_1y}.$$

Then it is easy to see that $\epsilon_1(u) = \|\hat{\alpha} - \alpha\|$ and $\epsilon_2(v) = \|\hat{\beta} - \beta\|$. With (7), we obtain

$$\kappa(x,y) - K_{xS_2}K^+_{S_1S_2}K_{S_1y} = \kappa(x,y) - \hat{\alpha}^T K^+_{S_1S_2}\hat{\beta} = (\kappa(x,y) - \kappa(u,v)) + \left( \alpha^T K^+_{S_1S_2}\beta - \hat{\alpha}^T K^+_{S_1S_2}\hat{\beta} \right), \tag{8}$$

for any $u \in S_1, v \in S_2$. According to Lemma 1, we get

$$\begin{aligned} \left|\hat{\alpha}^T K^+_{S_1S_2}\hat{\beta} - \alpha^T K^+_{S_1S_2}\beta\right| &\le \left\|\alpha^T K^+_{S_1S_2}\right\|\epsilon_2(v) + \left\|K^+_{S_1S_2}\beta\right\|\epsilon_1(u) + \left\|K^+_{S_1S_2}\right\|\epsilon_1(u)\epsilon_2(v) \\ &\le \epsilon_2(v) + \epsilon_1(u) + \left\|K^+_{S_1S_2}\right\|\epsilon_1(u)\epsilon_2(v). \end{aligned} \tag{9}$$

The last inequality in (9) follows from the fact that $\alpha^T K^+_{S_1S_2} = K_{uS_2}K^+_{S_1S_2}$ is a row in $K_{S_1S_2}K^+_{S_1S_2}$ and $K^+_{S_1S_2}\beta = K^+_{S_1S_2}K_{S_1v}$ is a column in $K^+_{S_1S_2}K_{S_1S_2}$, and meanwhile

$$\left\|K_{S_1S_2}K^+_{S_1S_2}\right\| = \left\|K^+_{S_1S_2}K_{S_1S_2}\right\| = 1.$$

We see from (8) to (9) that

$$\left|\kappa(x,y) - \hat{\alpha}^T K^+_{S_1S_2}\hat{\beta}\right| \le \left|\kappa(x,y) - \kappa(u,v)\right| + \epsilon_1(u) + \epsilon_2(v) + \left\|K^+_{S_1S_2}\right\|\epsilon_1(u)\epsilon_2(v), \quad \forall u \in S_1, \quad v \in S_2. \tag{10}$$

Minimizing the upper bound in (10) over all $u \in S_1, v \in S_2$ yields (6), which completes the proof. ∎

The entrywise estimate in Theorem 1 immediately leads to a matrix max norm estimate, which is proved in the next theorem.

**Theorem 2.** *Consider finite sets $X, Y \subset \mathbb{R}^d$ and kernel function $\kappa(x, y)$ defined on $X \times Y$. For any non-empty subsets $S_1 \subseteq X$ and $S_2 \subseteq Y$, denote by $\mathcal{X} = X \times Y$, $\mathcal{S} = S_1 \times S_2$. Then the approximation in (2) satisfies the following estimate*

$$\left\|K_{XY} - K_{XS_2}K^+_{S_1S_2}K_{S_1Y}\right\|_{\max} \le \max_{\substack{x\in X\\ y\in Y}} \min_{\substack{u\in S_1\\ v\in S_2}} \left( |\kappa(x,y) - \kappa(u,v)| + \epsilon_1(u) + \epsilon_2(v) + \left\|K^+_{S_1S_2}\right\|\epsilon_1(u)\epsilon_2(v) \right), \tag{11}$$

*where $\epsilon_1(u) = \left\|K_{xS_2} - K_{uS_2}\right\|$ and $\epsilon_2(v) = \left\|K_{S_1y} - K_{S_1v}\right\|$.*

*Proof.* Taking maximum of both sides of (6) over $x \in X, y \in Y$ yields (11). ∎

Assuming $\kappa$ is Lipschitz continuous, Theorems 1 and 2 imply that the bounds will be small if for any point $x \in X$ there is a point $u \in S_1$ nearby and for any point $y \in Y$ there is a point $v \in S_2$ nearby. As a result, Theorems 1 and 2 indicate that $S_1$ and $S_2$ should be evenly distributed inside $X$ and $Y$ in order to achieve a small approximation error. This can be more easily identified when the special case $S_1 = X$ or $S_2 = Y$ is considered.

**Corollary 1.** *Let $X, Y \subset \mathbb{R}^d$ be finite sets and $\kappa(x, y)$ be defined on $X \times Y$. For any non-empty subsets $S_1 \subseteq X$ and $S_2 \subseteq Y$, the following estimates hold*

$$\begin{aligned} \left\| K_{XY} - Q^{|}_{XS_2} K_{XY} \right\|_{\max} &\leq \max_{\substack{x\in X \\ y\in Y}} \min_{v\in S_2} \left( |\kappa(x,y) - \kappa(x,v)| + \|K_{Xy} - K_{Xv}\| \right), \\ \left\| K_{XY} - K_{XY} Q^{-}_{S_1 Y} \right\|_{\max} &\leq \max_{\substack{x\in X \\ y\in Y}} \min_{u\in S_1} \left( |\kappa(x,y) - \kappa(u,y)| + \|K_{xY} - K_{uY}\| \right), \end{aligned} \tag{12}$$

*where $Q^{|}_{XS_2} := K_{XS_2} K^{+}_{XS_2}$, $Q^{-}_{S_1 Y} := K^{+}_{S_1 Y} K_{S_1 Y}$.*

*Proof.* We only show the first inequality in (12) and the second one can be proved in a similar fashion. Note that the first inequality in (12) is a special case of (11) with $S_1 = X$. In this case, in the upper bound of (11), the minimum over $u \in X$ is no greater than the value achieved by choosing $u = x$. Hence we see that if $S_1 = X$ and $u = x$, then $\epsilon_1 = \|K_{xS_2} - K_{xS_2}\| = 0$ and (11) becomes

$$\left\| K_{XY} - Q^{|}_{XS_2} K_{XY} \right\|_{\max} \leq \max_{\substack{x\in X \\ y\in Y}} \min_{v\in S_2} \left( |\kappa(x,y) - \kappa(x,v)| + \left\| K_{Xy} - K_{Xv} \right\| \right),$$

which is the first inequality in (11). ∎

Assuming $\kappa$ is Lipschitz continuous, Corollary 1 further reveals the interconnection between the approximation accuracy and the geometry of sample points. Algebraically, $\|K_{XY} - Q^{|}_{XS_2} K_{XY}\|_{\max}$ and $\|K_{XY} - K_{XY} Q^{-}_{S_1 Y}\|_{\max}$ measure how well $K_{XS_2}$ and $K_{S_1 Y}$ capture the column and row spaces of $K_{XY}$, respectively. Geometrically, the bound on the right-hand side of (12) will be small if $S_1$ and $S_2$ are able to capture the global geometry of $X$ and $Y$, respectively.

### 2.2.2 | Geometric estimates

In the following, we reveal the geometric implication of the error bounds in Theorem 2 and Corollary 1 with the help of the so-called *discrete Lipschitz constant* as defined below. It is used to derive new error bounds that give a more straightforward interpretation of how the sets of landmark points $S_1$ and $S_2$ affect the accuracy of the approximation $K_{XY} \approx K_{XS_1} K^{+}_{S_1 S_2} K_{S_1 Y}$.

**Definition 1** (Discrete Lipschitz constant). Let $\kappa(x, y)$ be a function defined on $X \times Y$. Denote $\mathcal{Z} = Z_1 \times Z_2$, $\mathcal{S} = S_1 \times S_2$, $W_1 \times W_2$ as three non-empty subsets of $X \times Y$. The discrete Lipschitz constants of $\kappa$ associated with these three subsets are defined by

$$\begin{aligned} L(\mathcal{Z}, \mathcal{S}) &:= \min\{C : |\kappa(x,y) - \kappa(u,v)|^2 \leq C^2(|x-u|^2 + |y-v|^2) \ \ \forall (x,y) \in \mathcal{Z}, (u,v) \in \mathcal{S}\}, \\ L(Z_2, S_2)_{W_1} &:= \min\{C : |\kappa(x,y) - \kappa(x,v)|^2 \leq C^2 |y-v|^2 \ \ \forall x \in W_1, y \in Z_2, v \in S_2\}, \\ L(Z_1, S_1)_{W_2} &:= \min\{C : |\kappa(x,y) - \kappa(u,y)|^2 \leq C^2 |x-u|^2 \ \ \forall y \in W_2, x \in Z_1, u \in S_1\}. \end{aligned} \tag{13}$$

Since $X, Y$ are finite sets, each minimum in (13) exists. Note that in general $L(\mathcal{Z}, \mathcal{S})$ is *not* the Lipschitz constant of $\kappa$ since we do *not* assume $\kappa$ to be Lipschitz continuous or even defined outside $X \times Y$. If $\kappa(x, y)$ is Lipschitz continuous in a region containing $X \times Y$ with Lipschitz constant $L$, then it is easy to see that $L(\mathcal{Z}, \mathcal{S}) \leq L$, as stated in Proposition 1.

**Proposition 1.** *Let $\kappa(x, y)$ be a Lipschitz continuous function on a domain $D_1 \times D_2$ with Lipschitz constant $L$. For any discrete subset $X \times Y \subset D_1 \times D_2$, the discrete Lipschitz constants defined in (13) are all smaller than or equal to $L$.*

The discrete Lipschitz constants are introduced to make the result derived in this section applicable to general kernel functions with as few constraints as possible. In many applications, the kernel functions are actually not only Lipschitz continuous but also smooth in the domain of interest. Hence it is sufficient to use the Lipschitz constant. For example, in

machine learning and statistics, the Gaussian kernel $\exp\left(-\frac{|x-y|^2}{\sigma^2}\right)$ is smooth; radial basis functions like $\sqrt{1+|x-y|^2}$ and $(1+|x-y|^2)^{-1/2}$ are smooth; in potential theory, kernels like $\frac{1}{|x-y|}$ are smooth in $D_1 \times D_2$ with well-separated domains $D_1$ and $D_2$, which is a key assumption in the fast multipole method[4,7,23] and hierarchical matrices in general.[11,28,43,44]

Using the discrete Lipschitz constant, we can show in the following that the low-rank approximation error bound depends on the geometric quantity

$$\delta_{Z,S} := \max_{x\in Z} \operatorname{dist}(x,S) \quad S \subseteq Z. \tag{14}$$

The quantity $\delta_{Z,S}$ measures the closeness between $Z$ and $S$. The smaller $\delta_{Z,S}$ is, the "closer" $S$ is to $Z$. In fact, if $\delta_{Z,S}$ is small, then for any $x \in Z$, there exists a point in $S$ that is close to $x$.

We can now derive an error bound for (2) in terms of the geometric quantities $\delta_{X,S_1}$, $\delta_{Y,S_2}$ for subsets $S_1 \subseteq X$, $S_2 \subseteq Y$, respectively. The result is stated in Theorem 3.

**Theorem 3.** *Let $X, Y \subset \mathbb{R}^d$ be finite sets and $\kappa(x,y)$ be a function defined on $X \times Y$. For any non-empty subsets $S_1 \subseteq X$ and $S_2 \subseteq Y$, define $\mathcal{X} = X \times Y$, $\mathcal{S} = S_1 \times S_2$. Then the following estimate holds*

$$\left\| K_{XY} - K_{XS_2} K^+_{S_1S_2} K_{S_1Y} \right\|_{\max} \le C_1\delta_{X,S_1} + C_2\delta_{Y,S_2} + C_3\delta_{X,S_1}\delta_{Y,S_2},$$

*where*

$$\begin{aligned} C_1 &= L(\mathcal{X},\mathcal{S}) + \sqrt{r_2}L(X,S_1)_{S_2}, \\ C_2 &= L(\mathcal{X},\mathcal{S}) + \sqrt{r_1}L(Y,S_2)_{S_1}, \\ C_3 &= \left\|K^+_{S_1S_2}\right\| \sqrt{r_1r_2}L(X,S_1)_{S_2}L(Y,S_2)_{S_1}, \end{aligned} \tag{15}$$

*with $r_i = \operatorname{card}(S_i)$. Furthermore, if $\kappa(x,y)$ is Lipschitz continuous over $D_1 \times D_2$ containing $X \times Y$ with Lipschitz constant $L$, then*

$$\left\| K_{XY} - K_{XS_2} K^+_{S_1S_2} K_{S_1Y} \right\|_{\max} \le (1+\sqrt{r_2})L\delta_{X,S_1} + (1+\sqrt{r_1})L\delta_{Y,S_2} + \left\|K^+_{S_1S_2}\right\| \sqrt{r_1r_2}L^2\delta_{X,S_1}\delta_{Y,S_2}.$$

*Proof.* The result can be proved using Theorem 2 and the definition in (13). First we estimate the terms in the upper bound in Theorem 2. The definition of Lipschitz constants in (13) implies that

$$\begin{aligned} |\kappa(x,y) - \kappa(u,v)| &\le L(\mathcal{X},\mathcal{S})\left(|x-u|^2 + |y-v|^2\right)^{1/2} \le L(\mathcal{X},\mathcal{S})\left(|x-u| + |y-v|\right), \\ \epsilon_1(u) = \left\|K_{xS_2} - K_{uS_2}\right\| &\le \left(\sum_{v\in S_2} L(X,S_1)^2_{S_2}|x-u|^2\right)^{1/2} \le \sqrt{r_2}L(X,S_1)_{S_2}|x-u|, \\ \epsilon_2(v) = \left\|K_{S_1y} - K_{S_1v}\right\| &\le \left(\sum_{u\in S_1} L(Y,S_2)^2_{S_1}|y-v|^2\right)^{1/2} \le \sqrt{r_1}L(Y,S_2)_{S_1}|y-v|. \end{aligned} \tag{16}$$

Define $C_1, C_2, C_3$ as in (15). The estimates in (16), which separate $(x,u)$ and $(y,v)$ into different terms, allow us to organize the upper bound in (11) in terms of $|x-u|$ and $|y-v|$ and deduce that

$$\begin{aligned} &\max_{\substack{x\in X\\ y\in Y}} \min_{\substack{u\in S_1\\ v\in S_2}} \left( |\kappa(x,y) - \kappa(u,v)| + \epsilon_1(u) + \epsilon_2(v) + \left\|K^+_{S_1S_2}\right\|\epsilon_1(u)\epsilon_2(v)\right) \\ &\quad\le \max_{\substack{x\in X\\ y\in Y}} \min_{\substack{u\in S_1\\ v\in S_2}} \left(C_1|x-u| + C_2|y-v| + C_3|x-u||y-v|\right) \\ &\quad= \max_{\substack{x\in X\\ y\in Y}} \left(C_1\operatorname{dist}(x,S_1) + C_2\operatorname{dist}(y,S_2) + C_3\operatorname{dist}(x,S_1)\operatorname{dist}(y,S_2)\right) \\ &\quad= C_1\max_{x\in X}\operatorname{dist}(x,S_1) + C_2\max_{y\in Y}\operatorname{dist}(y,S_2) + C_3\max_{x\in X}\operatorname{dist}(x,S_1)\max_{y\in Y}\operatorname{dist}(y,S_2) \\ &\quad= C_1\delta_{X,S_1} + C_2\delta_{Y,S_2} + C_3\delta_{X,S_1}\delta_{Y,S_2}. \end{aligned}$$

This, together Theorem 2, completes the proof of the first inequality. The special case where $\kappa(x, y)$ is Lipschitz continuous follows immediately from the first inequality and Proposition 1. ∎

The estimate in Theorem 3 implies that in order to obtain a good approximation, $S_1, S_2$ should be chosen such that $\delta_{X,S_1}, \delta_{Y,S_2}$ are small. Geometrically, according to the definition of $\delta$ in (14), this means that $S_1$ and $S_2$ should represent the geometry of $X$ and $Y$ as much as possible. In the context of integral equations, a recent analytical study[45] also discussed the relationship between the approximation error of adaptive cross approximation (ACA) and the selected subsets measured by the geometric concept of *fill distance* (cf. Reference 46), which reflects how well the subset spans the computational domain. Both fill distance and $\delta$ in (14) provide similar geometric interpretations of the quality of the selected subsets, where the fill distance in [ 46, sect. 1.3]: $d(\Omega, X) = \sup_{y\in\Omega} \inf_{x\in X} |y - x|$ is defined for continuous regions $\Omega$ while $\delta$ focuses on finite sets of points. As a result, $\delta$ is always computable but fill distance is not in general.

The error estimates derived in this section apply to *any* subsets $S_1 \subset X$, $S_2 \subset Y$, regardless of the algorithm used to generate $S_1$, $S_2$. Thus when $S_1, S_2$ are poorly chosen (i.e. corresponding to poor low-rank approximation), we expect the bounds to reflect the fact that the matrix approximation error is large. This motivates the use of the estimates as indicators to distinguish "good" subsets and "bad" subsets, which will be investigated next in Section 2.3.

*Remark*. The estimate in Theorem 3 (as well as the one in Theorem 6) is derived to offer guidance to the fast and general algorithm based on subset selection, and it is not necessarily "tight". Since the goal is to design an algorithm with linear (or nearly linear) complexity in time and space for computing accurate low-rank kernel matrix approximations by subset selection, it is desirable to obtain a straightforward characterization of "good" subsets via analyzing the approximation error, in order to inspire the algorithm design. The geometric quantity $\delta$ serves the purpose. In fact, for $O(1)$ subsets $S_1, S_2$, the quantities $\delta_{X,S_1}, \delta_{Y,S_2}$ are not only easy to compute (with linear complexity in the size of $X, Y$), but also consistent with the practical result when distinguishing "good" and "bad" choices of subsets for low-rank approximation as illustrated in the following section. Hence we see that the geometric quantity $\delta$ from the theoretical result in Theorem 3 (or Theorem 6) leads to error *indicators* for subset selection.

## 2.3 | Subset quality indicators

The error bounds in Theorems 2 and 3 are fully computable and can be used to relate the choice of subset to the low-rank approximation error. Error bounds of this kind often arise in a posteriori error estimates for the numerical solution of partial differential equations using adaptive mesh refinement (AMR). In AMR, an error indicator, usually a computable term in the a posteriori error estimate, is used to indicate the quality of the numerical solution and determine whether further refinement is needed without knowing the exact solution (cf. References 47-53). Inspired by this philosophy, in low-rank compression methods based on geometric selection, we can use the error estimates to construct *subset quality indicators* for inferring the quality of the selected subsets. For any choice of subset $S_1 \times S_2 \subseteq X \times Y$, we consider the following five subset quality indicators:

$$\begin{aligned} &\text{indicator } 1 = \max_{\substack{x\in X\\ y\in Y}} \min_{\substack{u\in S_1\\ v\in S_2}} |\kappa(x, y) - \kappa(u, v)|, \quad \text{indicator } 2 = \max_{x\in X} \min_{u\in S_1} \|K_{xS_2} - K_{uS_2}\|, \\ &\text{indicator } 3 = \delta_{X,S_1}, \quad \text{indicator } 4 = \delta_{Y,S_2}, \quad \text{indicator } 5 = \left\|K^+_{S_1S_2}\right\|. \end{aligned} \tag{17}$$

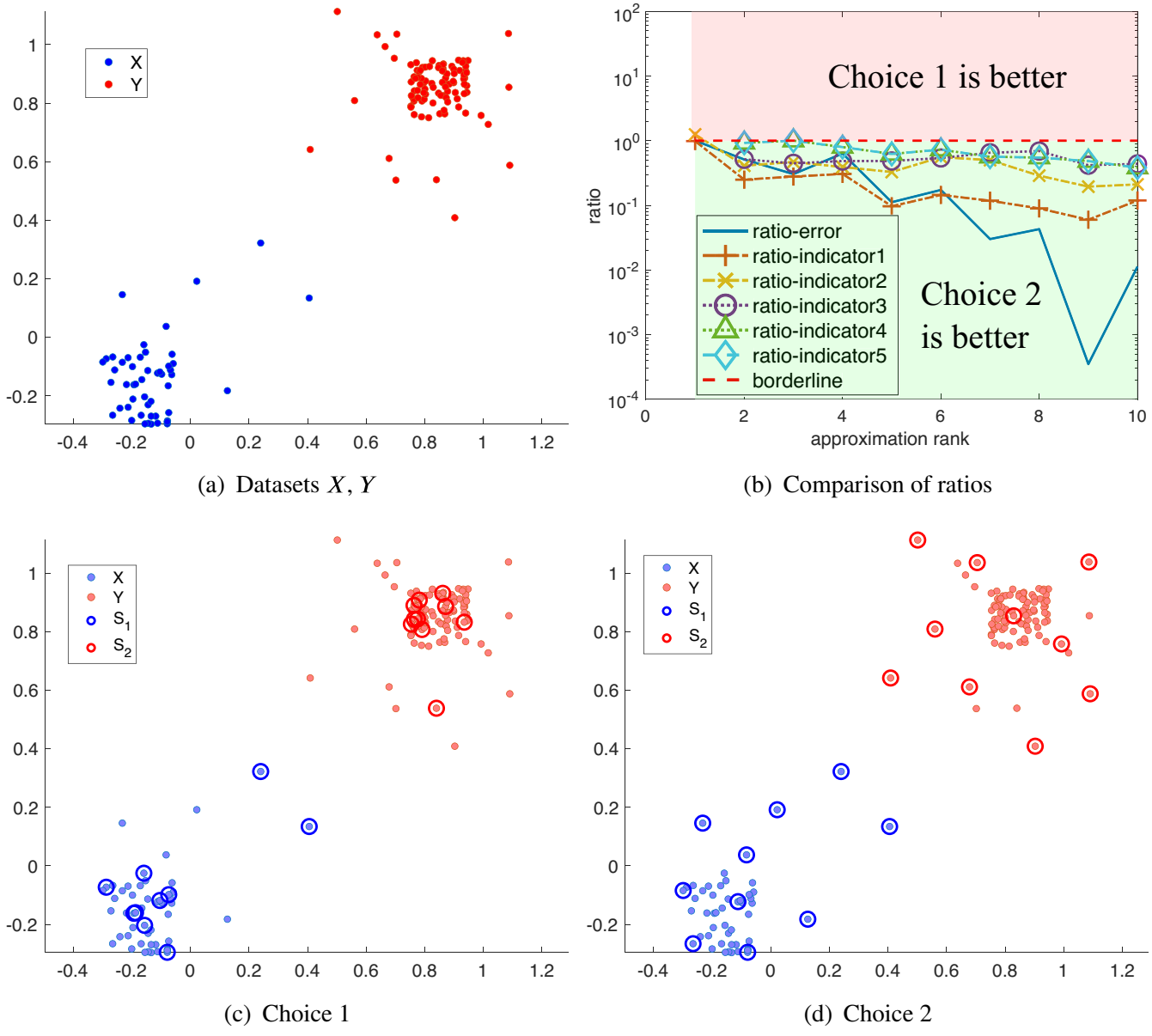
The first two indicators are related to the upper bound derived in Theorem 2, while the last three indicators are from the estimate in Theorem 3. The costs for computing the indicators are *not* the same. In fact, assume $K_{XY}$ is $m$-by-$n$ and there are $O(1)$ points in $S_1$ and $S_2$. The computational complexities for the five indicators in (17) are: $O(mn)$, $O(m)$, $O(m)$, $O(n)$, $O(1)$, respectively. Hence in practice, it is more convenient to use the latter four indicators.

Given different choices of subsets, we present numerical experiments below to demonstrate how to use the subset quality indicators to predict which choice is more likely to yield a better approximation without computing the exact matrix approximation error. The results also underscore the impact of the geometry of the selected subset on the low-rank approximation accuracy. We perform two experiments, one with a rectangular kernel matrix associated with *two* sets of points and the other with a symmetric positive definite kernel matrix associated with *one* set of points.

*Experiment* 1. We consider the approximation $K_{XY} \approx K_{XS_2} K^{+}_{S_1S_2} K_{S_1Y}$. The kernel function is chosen as $\kappa(x, y) = \log|x - y|$ and the rectangular kernel matrix $K_{XY}$ is associated with $X$, $Y$ (illustrated in Figure 2a), where $X$ contains 50 points and $Y$ contains 100 points. We considers two choices for $S_1 \times S_2 \subset X \times Y$. Choice 1 generates points $S_1$, $S_2$ via random sampling from $X$, $Y$. Choice 2 chooses evenly distributes points to form $S_1$, $S_2$ using FPS. These subsets are shown in Figure 2c,d. To determine which choice yields the better approximation, we take the ratio of the respective indicators and compare it to the ratio of the exact matrix approximation errors from the two choices. Namely, we compute

$$\text{ratio-indicator } k = \frac{\text{indicator } k \text{ of Choice 2}}{\text{indicator } k \text{ of Choice 1}}, \quad \text{ratio-error} = \frac{\text{matrix error of Choice 2}}{\text{matrix error of Choice 1}},$$

where the matrix approximation error is measured in max norm. If the ratio-indicator is larger than 1, then the prediction is that Choice 1 is better. Otherwise, the prediction is that Choice 2 is better. We then compare the indicator ratios to the ground truth: the ratio of matrix approximation errors between Choice 2 and Choice 1.




(a) Datasets $X$, $Y$

(b) Comparison of ratios

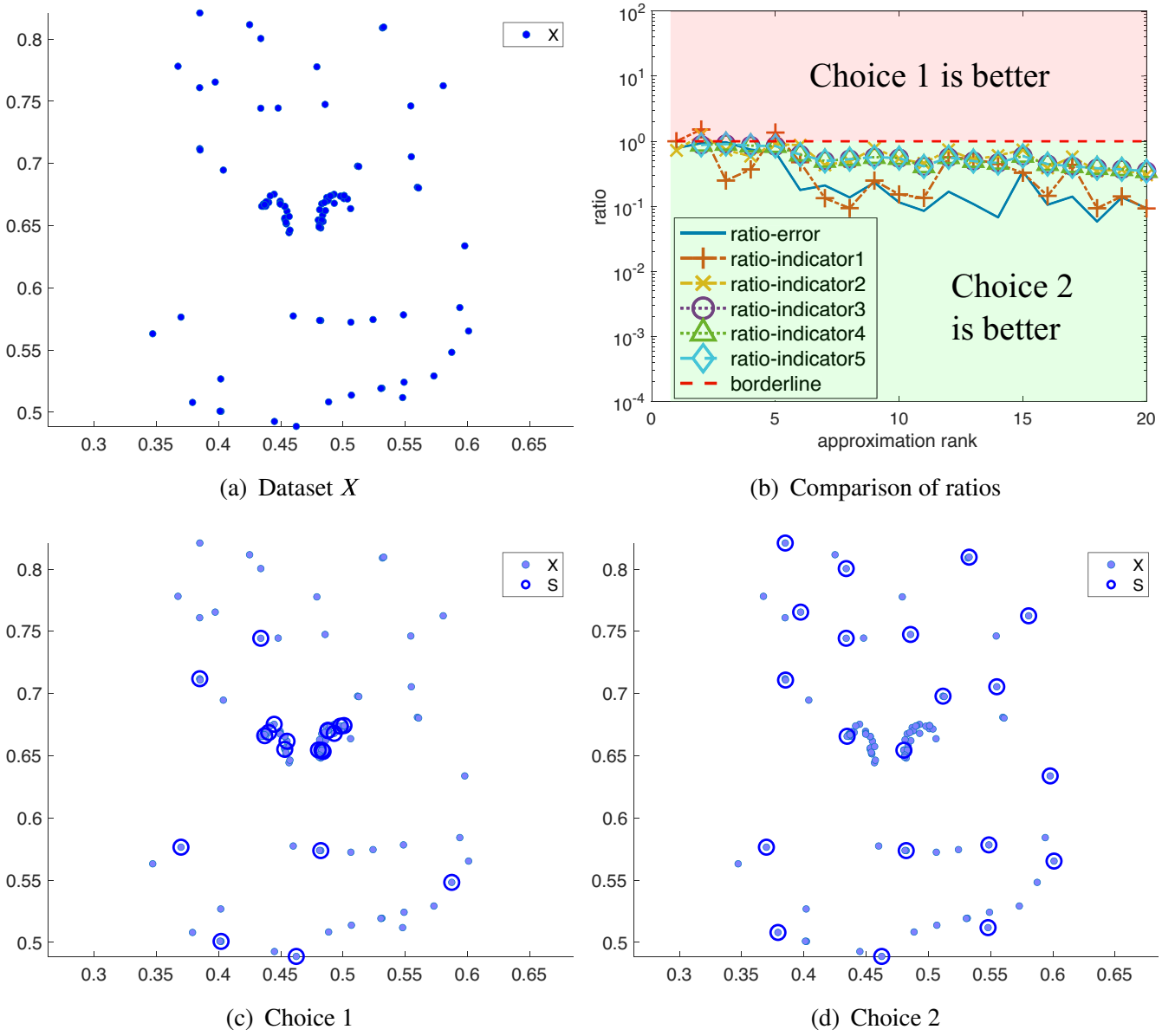
(c) Choice 1

(d) Choice 2

**FIGURE 2** Experiment 1: Predicting the better choice of subsets $S_1, S_2$ using subset indicators in (17).

If the indicator ratio is consistent with the error ratio, that is, both larger than 1 or both smaller than 1, then the prediction based on the indicator is correct. The result is shown in Figure 2b. It is easily seen that, for different approximation ranks, the indicator ratios and the error ratio always stay below the horizontal line $y = 1$. Hence the indicators correctly predict the fact that Choice 2 of subsets yields a better low-rank approximation than Choice 1. Furthermore, note that unlike Choice 1, the points in Choice 2 are evenly distributed over the dataset and thus are expected to yield a better approximation according to the theoretical results in Section 2.2. ■

*Experiment* 2. We consider the Gaussian kernel $\kappa(x, y) = \exp(-|x - y|^2/0.09)$ and the symmetric approximation $K_{XX} \approx K_{XS}K_{SS}^{+}K_{SX}$, where the dataset $X$ contains 100 points as shown in Figure 3a. We follow the same choices of subset as in Experiment 1, that is, Choice 1 selects random samples while Choice 2 selects evenly distributed points. These two choices of subset $S$ are shown in Figure 3c,d. We compute the same indicators as in (17), where in this case $Y = X$ and $S_2 = S_1 = S$. The result is shown in Figure 2b. We see that when the approximation rank is larger than 5, all indicator ratios and the error ratio stay below the horizontal line $y = 1$ simultaneously. This implies that Choice 2 yields a better approximation and the indicators give the




(a) Dataset $X$ (b) Comparison of ratios

(c) Choice 1 (d) Choice 2

**FIGURE 3** Experiment 2: Predicting the better choice of subset $S$ using subset indicators in (17).

correct prediction. Again, we see that evenly distributed points yield a better approximation, as discussed in Section 2.2. ■

## 3 | ONE-SIDED LOW-RANK KERNEL MATRIX APPROXIMATION

This section analyzes the data-driven geometric approach for the one-sided low-rank approximation (3). Compared to the two-sided case, the algorithm in the one-sided case only samples points from one set of points and applies an algebraic factorization to postprocess the selected submatrix. Numerical experiments in Section 4 show that the one-sided algorithm is slightly more accurate.

The key algebraic technique we use is the strong rank-reveal QR factorization (SRRQR[34]). The original setting in Reference 34 considers "tall" matrices and direct application of the result yields pessimistic computational complexity in the current setting where "short" matrices are of interest. To resolve this issue, we discuss the new setting and derive a nearly optimal complexity estimate in Section 3.1. We present the algorithm for the one-sided low-rank approximation in Section 3.2 and provide the error analysis in Section 3.3. The special case of a symmetric kernel matrix $K_{XX}$ with a symmetric kernel $\kappa(x, y) = \kappa(y, x)$ is discussed in Section 3.4.

### 3.1 | Strong rank-revealing QR for "short" matrices

The classical result on SRRQR is cited in Proposition 2. Algorithms for computing SRRQR are proposed in Reference 34 and we use [34, alg. 4] in our approach to postprocess the $r \times n$ matrix $K_{XS_2}^T$, where $S_2 \subseteq Y$ is the selected subset with $r$ points. We point out that the original result on SRRQR (cited in Proposition 2) only considers "tall" matrices of size $m \times n$ with $m \geq n$ and the complexity contains a term of $O(n^3)$. Such a complexity will be too pessimistic for the "short" matrix $K_{XS_2}^T$ of size $r \times n$ with $r \ll n$ in general. To obtain a complexity estimate that is nearly optimal in $n$, we present in this section a rigorous analysis of SRRQR for "short" matrices. The result in Proposition 3 shows that the complexity of SRRQR is in between $O(r^2 n)$ and $O(r^2 n \log_s n)$ for $r \times n$ matrices with rank $r$. That is, the complexity is linear or nearly linear in $n$.

**Proposition 2** (Strong Rank-revealing QR Factorization[34]). *Let $M$ be an $m \times n$ matrix with $m \geq n$. The SRRQR of $M$ yields $M = Q \begin{bmatrix} A_k & B_k \\ & C_k \end{bmatrix} \Pi$, where $Q$ is $m \times m$ orthogonal, $\Pi$ is a permutation matrix, $A_k$ is a well-conditioned $k \times k$ upper triangular matrix with the $i$th $(1 \leq i \leq k)$ singular value $\sigma_i(A_k) \geq \sigma_i(M)/\sqrt{1 + s^2 k(n-k)}$, $C_k$ satisfies $\sigma_i(C_k) \leq \sigma_{k+j}(M)\sqrt{1 + s^2 k(n-k)}$ with $1 \leq j \leq n-k$, $\|A_k^{-1} B_k\|_{\max} \leq s$. Here $s > 1$ is a user-specified constant. The complexity for SRRQR is $O(mn^2 + n^3 \log_s n)$.*

We are interested in applying SRRQR to "short" $r \times n$ matrices as described below and the complexity in Proposition 2 does not reflect the efficiency in the new setting, where $M$ is $r \times n$ with rank $r$.

Algorithm [34, alg. 4] for computing the SRRQR in Proposition 3:

1. Compute $R = [A_r, B_r] := \mathcal{QR}(M)$ and define $\Pi = I$, where $\mathcal{QR}$ denotes the QR factorization;
2. `while` $\|A_r^{-1} B_r\|_{\max} > s$ `do`
3. Find $i, j$ such that $|(A_r^{-1} B_r)_{i,j}| > s$;
4. Compute $R = [A_r, B_r] := \mathcal{QR}(R\Pi_{i,j+r})$ and $\Pi := \Pi\, \Pi_{i,j+k}$, where $\Pi_{i,j+k}$ denotes the permutation that interchanges the $i$th and $j+k$ th columns;
5. `endwhile`

We analyze SRRQR for "short" matrices and prove the nearly optimal complexity of the algorithm above. The result is summarized in Proposition 3. A straightforward corollary of Proposition 3 gives a stable interpolative decomposition for "tall" matrices (Corollary 2) that will be used in the one-sided low-rank approximation in Algorithm 2.

**Proposition 3** (SRRQR for "short" matrices). *Let $M$ be an $r \times n$ matrix with rank $r$ (thus $r \leq n$). The SRRQR of $M$ yields $M = Q \begin{bmatrix} A_r & B_r \end{bmatrix} \Pi$, where $Q, P, \Pi, A_r, B_r$ are as in Proposition 2, with $\|A_r^{-1} B_r\|_{\max} \leq s$. Here $s > 1$ is a user-specified constant. The complexity for computing such a factorization is $O(n_{\text{iter}} r^2 n)$, where $n_{\text{iter}}$ denotes the number of* `while` *loops in Line 2 of the SRRQR algorithm above, and $n_{\text{iter}}$ is between $O(1)$ and $O(\log_s n)$. That is, the complexity of SRRQR is between $O(r^2 n)$ and $O(r^2 n \log_s n)$.*

*Proof.* The QR factorization in Line 1: $[A_r, B_r] := \mathcal{QR}(M)$ for has complexity $O(r^2 n)$. The number of `while` loops $n_{\text{iter}}$ is at most $O(\log_s n)$ according to the estimate in [34, sect. 4.4]. In fact, the $O(\log_s n)$ estimate is calculated for the *two* nested while loops in [34, alg. 5] in which $n_{\text{iter}}$ corresponds to the inner while loop. As a result, the complexity of $n_{\text{iter}}$ must not exceed $O(\log_s n)$. In the `while` loop, computing each $\|A_r^{-1}B_r\|_{\max}$ requires $O(r^2 n)$ complexity, since $A_r$ is triangular and $B_r$ is $r \times (n-r)$.

Next we analyze the complexity for each QR factorization to $\tilde{R} := R\Pi_{i,j+r}$ in the `while` loop. Without loss of generality, we assume $i = 1$. This is because in this case, $\tilde{R} = R\Pi_{1,j+r}$ has the following sparsity pattern (blank entries denote zeros) and QR will be applied to the whole matrix, which results in the highest complexity. If $i > 1$, then the first $i-1$ columns already form an upper triangular matrix and QR is applied to the non-triangular submatrix in the lower right part of $\tilde{R}$ whose size is strictly smaller than $r$-by-$n$.

$$\tilde{R} := \begin{bmatrix} * & * & \dots & * & * & \dots & \dots & \dots \\ * & * & \dots & \vdots & * & \dots & \dots & \dots \\ * & & \ddots & \vdots & * & \dots & \dots & \dots \\ * & & & * & * & \dots & \dots & \dots \\ * & & & & * & \dots & \dots & \dots \end{bmatrix}$$

To compute the QR factorization of $\tilde{R}$ efficiently, we apply Householder reflection or Givens rotation to submatrices of row size two in a *bottom-up* fashion, which will reduce the matrix into an upper Hessenberg form. Then we apply Householder reflection or Givens rotation to the upper Hessenberg form in a *top-down* fashion to obtain an upper triangular matrix, which completes the QR factorization.

In the bottom-up reduction, we first apply Householder reflection or Givens rotation to the last two rows of $\tilde{R}$ to zero out the entry in the bottom left corner (see (18)), i.e., entry $(r, 1)$ in $\tilde{R}$. Note that this will introduce a nonzero entry at $(r, r-1)$, denoted by '•' in (18).

$$\begin{bmatrix} * & & * & * & \dots & \dots & \dots \\ * & & & * & \dots & \dots & \dots \end{bmatrix} \to \begin{bmatrix} * & & * & * & \dots & \dots & \dots \\ & & \bullet & * & \dots & \dots & \dots \end{bmatrix} \tag{18}$$

Applying the same process recursively to the two-row submatrices (rows $k-1, k$) with $k = r-1, r-2, \dots, 3$, we obtain an upper Hessenberg form:

$$\begin{bmatrix} * & * & \dots & * & * & \dots & \dots & \dots \\ * & * & \dots & \vdots & * & \dots & \dots & \dots \\ & \bullet & \ddots & \vdots & * & \dots & \dots & \dots \\ & & \ddots & * & * & \dots & \dots & \dots \\ & & & \bullet & * & \dots & \dots & \dots \end{bmatrix} \tag{19}$$

The total complexity of this bottom-up procedure is

$$O\left((n-r+3) + (n-r+4) + \cdots + (n-r+r)\right) = O(rn),$$

where the number in each inner parenthesis denotes the number of nonzero columns in each two-row matrix.

Then we reduce the upper Hessenberg form in (19) into an upper triangular matrix by applying Householder reflection or Givens rotation sequentially (in a top-down fashion) to the two-row submatrix (rows $k, k+1$) with $k = 1, 2, \dots, r-1$, in order to zero out the subdiagonal entries. Similar to the bottom-up procedure, it is easy to see that the total complexity of the top-down procedure is also $O(rn)$.

Therefore, we see that the each execution in the `while` loop is donimated by the cost of computing $\|A_r^{-1}B_r\|_{\max}$, with $O(r^2 n)$ complexity. The total complexity of the entire algorithm is then

$$O(r^2 n + n_{\text{iter}} r^2 n) = O(n_{\text{iter}} r^2 n).$$

Given the fact that $n_{\text{iter}}$ is between $O(1)$ and $O(\log_s n)$, the complexity of SRRQR is between $O(r^2 n)$ and $O(r^2 n \log_s n)$. ∎

**Corollary 2.** *Let $M$ be an $n \times r$ matrix with rank $r$. Then $M$ can be factorized via SRRQR as $M = P \begin{bmatrix} I \\ G \end{bmatrix} M_1$, where $P$ is a permutation matrix, $I$ is an identity matrix, $M_1$ is the matrix that consists of the first $r$ rows of $P^T M$, and $\|G\|_{\max} \le s$. Here $s > 1$ is a user-specified constant. The computational complexity is at most $O(r^2 n \log_s n)$.*

*Proof.* Applying SRRQR to the $r \times n$ matrix $M^T$ yields

$$M^T = Q \begin{bmatrix} A_r & B_r \end{bmatrix} \Pi, \tag{20}$$

where $Q, \Pi, A_r, B_r$ are matrices in Proposition 3. In particular, $\|A_r^{-1} B_r\|_{\max} \le s$. Meanwhile, the complexity is at most $O(r^2 n \log_s n)$ according to Proposition 3.

Since $\Pi$ is a permutation matrix, $QA_r$ is a submatrix of $M^T$ containing the first $r$ columns of $M^T \Pi^T$. Define $M_1^T = QA_r$. We see that $M_1$ contains the first $r$ rows of $\Pi M$. We can then rewrite (20) as

$$M^T = QA_r \begin{bmatrix} I & A_r^{-1} B_r \end{bmatrix} \Pi = M_1^T \begin{bmatrix} I & A_r^{-1} B_r \end{bmatrix} \Pi.$$

Transposing both sides yields the desired factorization with $P := \Pi^T$, $G := A_r^{-1} B_r$, $\|G\|_{\max} \le s$. The complexity is at most $O(r^2 n \log_s n)$ thanks to Proposition 3. ∎

*Remark* 1. Note that in the original SRRQR,[34] the permutation is performed for the smaller dimension, that is, $n$ columns for a "tall" $m \times n$ matrix with $m \ge n$. In Proposition 3 and Corollary 2, the permutation is performed over the larger dimension. This calls for the new complexity analysis in the proof of Proposition 3 different from the original estimate in Reference 34.

## 3.2 | Algorithm

The one-sided approximation method consists of two stages. In the first stage, a subset $S_2 \subseteq Y$ is selected using a linear complexity geometric selection scheme (Section 1). In the second stage, we compute the interpolative decomposition in (3) via applying SRRQR[34] to guarantee the maximum norm of the column basis matrix is bounded by a prescribed number $s > 1$. More precisely, we apply SRRQR to the "short" matrix $K_{XS_2}^T$ and then transpose the output to obtain $K_{XS_2} = P \begin{bmatrix} I \\ G \end{bmatrix} K_{\mathcal{I}_1 S_2}$, with $\mathcal{I}_1 \subseteq X$, $P$ a permutation matrix and $\|G\|_{\max} \le s$. See Corollary 2 for a more detailed discussion.

The full one-sided compression algorithm is summarized in Algorithm 2. Notice that in Step 2 of Algorithm 2, SRRQR is only used to obtain a stable factorization of $K_{XS_2}$ and thus *no* approximation error is introduced.

**Algorithm 2.** Data-driven one-sided compression of $K_{XY}$ with two sets of points $X, Y$

*Input:* Datasets $X = \{x_1, \dots, x_m\}$, $Y = \{y_1, \dots, y_n\} \subset \mathbb{R}^d$, kernel function $\kappa$, number of sample points $r$ for $Y$
*Output:* Low-rank approximation $K_{XY} \approx U K_{\mathcal{I}_1 Y}$ in (3)

Apply a linear complexity geometric selection algorithm to $Y$ to generate $r$ sample points $S_2 \subseteq Y$

Apply SRRQR-based ID to the $m$-by-$r$ kernel matrix $K_{XS_2}$ : $K_{XS_2} = P \begin{bmatrix} I \\ G \end{bmatrix} K_{\mathcal{I}_1 S_2}$, where $I$ is an identity matrix, $\mathcal{I}_1 \subseteq X$, $P$ is a permutation matrix that maps the row indices of $I$ to the indices for $\mathcal{I}_1$ in $X$, and $\|G\|_{\max} \le 2$.

Define $U = P \begin{bmatrix} I \\ G \end{bmatrix}$.

Return $U, K_{\mathcal{I}_1 Y}$

Compared to purely algebraic methods such as LU, QR, rank-revealing QR, and SVD decompositions, Algorithm 2 does not access the full kernel matrix and scales linearly or nearly linearly with respect to the data size. Compared to the

proxy point methods,[26,27] hybrid cross approximation,[30] Algorithm 2 does not require the evaluation of the kernel function outside the given dataset (where the function may not necessarily be defined) and is able to scale to high dimensions. In terms of numerical stability, Algorithm 2 leverages the robustness of algebraic methods to obtain a stable factorization compared to the two-sided approximation. As we shall see in Section 4, despite being more stable and more general, the error-time trade-off of the proposed method can be noticeably better than that of existing methods.

In addition to the factorization $K \approx UK_{\mathcal{I}_1 Y}$, a similar one-sided factorization $K \approx K_{X\mathcal{I}_2}V^*$ can be computed by applying Algorithm 2 to $K^*_{XY}$. That is, we first select a subset from $X$ and then apply ID to obtain a subset $\mathcal{I}_2 \subseteq Y$. Both options first apply geometric selection to $X$ or $Y$ to obtain a small submatrix and then apply algebraic factorization to it. They differ in which data set the geometric selection is applied to, that is, $X$ or $Y$. If one set contains significantly more points than the other one (for example, $m \gg n$), for efficiency, it is better to perform geometric selection on the larger set to reduce its size to $O(1)$, so that the following algebraic factorization, which is more expensive than geometric selection, is applied to a submatrix with a smaller dimension $n$-by-$O(1)$ instead of $m$-by-$O(1)$.

### 3.3 | Complexity and error analysis

In this section, we analyze the complexity and the approximation error of Algorithm 2. First, we show that Algorithm 2 scales as $O(r^2(m+n))$ for obtaining a rank-$r$ approximation to an $m \times n$ kernel matrix.

**Theorem 4.** *Given $X = \{x_i\}_{i=1}^m$, $Y = \{y_i\}_{i=1}^n$ in $\mathbb{R}^d$ and kernel function $\kappa$, the complexity of Algorithm 2 to compute a rank-$r$ approximation $K_{XY} \approx UK_{\mathcal{I}_1 Y}$ is $O(dr^2(m+n))$.*

*Proof.* Compressing a set of $n$ points into $r$ points with any scheme in Section 1 has a complexity at most $O(dr^2 n)$. The cost of applying ID on a $m$-by-$r$ matrix $K_{XS_2}$ is $O(r^2 m)$. Therefore, the overall complexity of Algorithm 2 is $O(dr^2(m+n))$. ∎

Next we analyze the approximation error for $K_{XY} \approx UK_{\mathcal{I}_1 Y}$ computed by Algorithm 2. We will see that, different from the two-sided factorization, the error bound for $K_{XY} \approx UK_{\mathcal{I}_1 Y}$ does not involve the norm of the pseudoinverse of the matrix.

**Theorem 5.** *Let $X$ and $Y$ be finite sets in $\mathbb{R}^d$ and $\kappa(x,y)$ be defined on $X \times Y$. For any non-empty subset $S \subseteq Y$, let $K_{XS}$ be decomposed by SRRQR-based ID as $K_{XS} = UK_{\mathcal{I}S} = P\begin{bmatrix} I \\ G \end{bmatrix} K_{\mathcal{I}S}$ with $\|G\|_{\max} \le 2$. Then*

$$\begin{aligned} \|K_{XY} - UK_{\mathcal{I}Y}\|_{\max} \le & \max_{\substack{x\in X \\ y\in Y}} \min_{v\in S} \left( |\kappa(x,y) - \kappa(x,v)| + \|K_{Xy} - K_{Xv}\| \right) \\ & + 2r \max_{\substack{x\in \mathcal{I} \\ y\in Y}} \min_{v\in S} \left( |\kappa(x,y) - \kappa(x,v)| + \|K_{Xy} - K_{Xv}\| \right), \end{aligned} \tag{21}$$

*where $r = \mathrm{card}(\mathcal{I})$.*

*Proof.* We decompose $K_{XY}$ as

$$\begin{aligned} K_{XY} &= K_{XS}K^+_{XS}K_{XY} + E_1 \quad \text{with} \quad E_1 = K_{XY} - K_{XS}K^+_{XS}K_{XY} \\ &= P\begin{bmatrix} I \\ G \end{bmatrix} K_{\mathcal{I}S}K^+_{XS}K_{XY} + E_1 \\ &= P\begin{bmatrix} I \\ G \end{bmatrix} (K_{\mathcal{I}Y} + E_2) + E_1 \quad \text{with} \quad E_2 = K_{\mathcal{I}S}K^+_{XS}K_{XY} - K_{\mathcal{I}Y} \\ &= P\begin{bmatrix} I \\ G \end{bmatrix} K_{\mathcal{I}Y} + P\begin{bmatrix} I \\ G \end{bmatrix} E_2 + E_1. \end{aligned} \tag{22}$$

According to Corollary 1,

$$\|E_1\|_{\max} \le \max_{\substack{x\in X \\ y\in Y}} \min_{v\in S} \left( |\kappa(x,y) - \kappa(x,v)| + \|K_{Xy} - K_{Xv}\| \right). \tag{23}$$

Similarly, for $E_2$, we have

$$\|E_2\|_{\max} \le \max_{\substack{x\in I\\ y\in Y}} \min_{v\in S} \left( |\kappa(x,y) - \kappa(x,v)| + \|K_{Xy} - K_{Xv}\| \right). \tag{24}$$

Since $\|G\|_{\max} \le 2$ and the row size of $E_2$ is equal to $r = \mathrm{card}(\mathcal{I})$, it follows that

$$\|GE_2\|_{\max} \le 2r\|E_2\|_{\max}.$$

Together with (24), (23), and (22), we deduce the inequality in (21). ∎

The estimate in Theorem 5 relates the approximation error to the subset $S$. We remark that in the estimate, $r$ is fixed and $S$ is viewed as a variable since we aim to study how the choice of $S$ affects the low-rank approximation accuracy. This is different from estimates that study how error decays with $r$. We show a more geometric characterization of the error bound of Theorem 5 in the following theorem. This theorem implies that the approximation error depends on the ability of $S$ to capture $Y$, which is similar to the two-sided approximation case described in Theorem 3.

**Theorem 6.** *Let $X, Y, \kappa, S, \mathcal{I}$ be given in Theorem 5 and let $K_{XY} \approx UK_{\mathcal{I}Y}$ be the approximation in Theorem 5. Then*

$$\begin{aligned}\|K_{XY} - UK_{\mathcal{I}Y}\|_{\max} \le{}& L(X\times Y, X\times S)\delta_{Y,S} + (1+2r)\sqrt{m}L(Y,S)_X\delta_{Y,S}\\ &+ 2rL(\mathcal{I}\times Y, \mathcal{I}\times S)\delta_{Y,S},\end{aligned} \tag{25}$$

*where $m = \mathrm{card}(X)$, $r = \mathrm{card}(\mathcal{I})$. Furthermore, if $\kappa(x,y)$ is Lipschitz continuous over $D_1 \times D_2$ containing $X \times Y$ with Lipschitz constant $L$, then*

$$\|K_{XY} - UK_{\mathcal{I}Y}\|_{\max} \le L\delta_{Y,S} + (1+2r)\sqrt{m}L\delta_{Y,S} + 2rL\delta_{Y,S}.$$

*Proof.* The proof is analogous to that of Theorem 3. According to (16), we deduce that

$$\begin{aligned}\max_{\substack{x\in X\\ y\in Y}} \min_{v\in S} \left( |\kappa(x,y) - \kappa(x,v)| + \|K_{Xy} - K_{Xv}\| \right) &\le \max_{\substack{x\in X\\ y\in Y}} \min_{v\in S} \left( L(X\times Y, X\times S)|y-v| + \sqrt{m}L(Y,S)_X|y-v| \right)\\ &= \max_{\substack{x\in X\\ y\in Y}} \left( L(X\times Y, X\times S)\mathrm{dist}(y,S) + \sqrt{m}L(Y,S)_X\mathrm{dist}(y,S) \right)\\ &= L(X\times Y, X\times S)\delta_{Y,S} + \sqrt{m}L(Y,S)_X\delta_{Y,S}.\end{aligned}$$

Similarly, it can be deduced that

$$\max_{\substack{x\in I\\ y\in Y}} \min_{v\in S} \left( |\kappa(x,y) - \kappa(x,v)| + \|K_{Xy} - K_{Xv}\| \right) \le L(\mathcal{I}\times Y, \mathcal{I}\times S)\delta_{Y,S} + \sqrt{m}L(Y,S)_X\delta_{Y,S}.$$

Inserting the above two inequalities into (21) completes the proof of (25). The special case of $\kappa(x,y)$ being Lipschitz follows immediately from (25) and Proposition 1. ∎

From Theorem 6, it is easy to see that smaller $\delta_{Y,S}$ contributes to better approximation and the approximation error is zero if $S = Y$. Also, we see that the smoother the kernel function is (small Lipschitz constant), the more accurate the low-rank approximation will be. This is consistent with the fact that smooth kernel functions yield kernel matrices with rapidly decaying singular values.

Compared to the error estimates in Theorems 2 and 3 for the *two-sided* factorization, the estimates for the *one-sided* factorization in Theorems 5 and 6 appear to be better since they do not contain the norm of any matrix, for example, the possibly large factor $\|K^+_{S_1S_2}\|$ in Theorems 2 and 3. This factor disappears when only *one* geometric selection is performed (for either rows or columns), as shown in Corollary 1 and Theorem 5.

### 3.4 | The symmetric case

In this section, we consider a variant of the approximation (3) when the kernel matrix $K_{XX} = [\kappa(x,y)]_{x,y\in X}$ is associated with *one* set of points $X$ and a symmetric kernel $\kappa(x,y)$. This type of kernel matrix arises frequently as covariance or correlation matrices in statistics and machine learning. In order to preserve the symmetry of $K_{XX}$, we compute a symmetric factorization of the form

$$K_{XX} \approx U K_{\mathcal{I}\mathcal{I}} U^T \quad \text{with} \quad \mathcal{I} \subseteq X \tag{26}$$

whose structure-preserving properties are shown in the next proposition. This is the symmetric version of the "double-sided ID".[54]

**Proposition 4.** *If $K_{XX}$ is symmetric, then the low-rank approximation $UK_{\mathcal{I}\mathcal{I}}U^T$ in (26) is also symmetric. If $K_{XX}$ is assumed to be positive semi-definite, then $UK_{\mathcal{I}\mathcal{I}}U^T$ is also positive semi-definite.*

*Proof.* Since $\mathcal{I} \subseteq X$, $K_{\mathcal{I}\mathcal{I}}$ is a principal submatrix of $K_{XX}$. If $K_{XX}$ is symmetric, $K_{\mathcal{I}\mathcal{I}}$ is also symmetric, which implies that $UK_{\mathcal{I}\mathcal{I}}U^T$ is symmetric.

If $K_{XX}$ is positive semi-definite, then $K_{\mathcal{I}\mathcal{I}}$ is also positive semi-definite since it is a principal submatrix of $K_{XX}$. As a result, $UK_{\mathcal{I}\mathcal{I}}U^T$ is symmetric positive semi-definite. ∎

The symmetric factorization in (26) is a straightforward extension of the one-sided factorization and the algorithm is summarized in Algorithm 3.

**Algorithm 3.** Data-driven compression of $K_{XX}$ with one set of points $X$

*Input:* Dataset $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, kernel function $\kappa$, number of sample points $r$
*Output:* Low-rank approximation $K_{XX} \approx UK_{\mathcal{I}\mathcal{I}}U^T$

Apply a linear complexity geometric selection algorithm to $X$ to generate $r$ sample points $S$
Apply SRRQR-based ID to the $n$-by-$r$ kernel matrix $K_{XS}$:

$$K_{XS} = [\kappa(x,y)]_{\substack{x\in X\\ y\in S}} = P \begin{bmatrix} I \\ G \end{bmatrix} K_{\mathcal{I}S}, \tag{27}$$

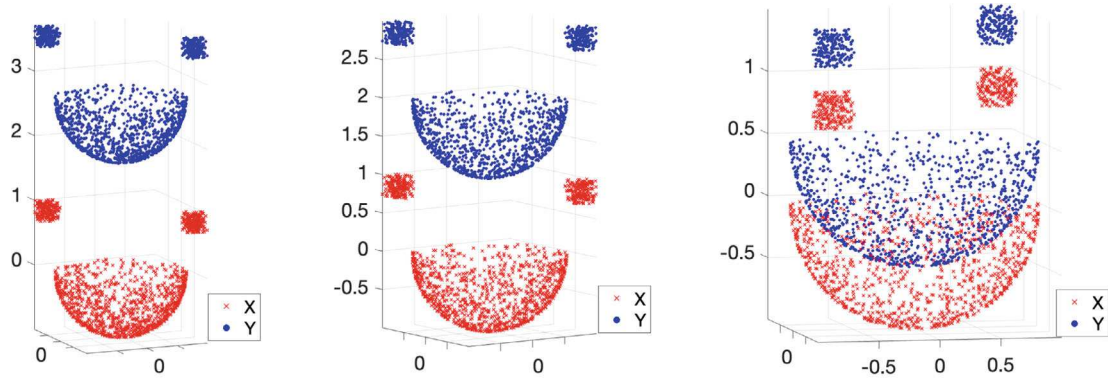
where $\mathcal{I} \subseteq X$, $P$ is a permutation matrix that maps the row indices in $I$ to the indices for $\mathcal{I}$ in $X$, and $\|G\|_{\max} \leq 2$
Define $U = P \begin{bmatrix} I \\ G \end{bmatrix}$
Return $U, K_{\mathcal{I}\mathcal{I}}$

## 4 | NUMERICAL EXPERIMENTS

In this section, we illustrate the data-driven geometric approach using both low- and high-dimensional data. All experiments were conducted in MATLAB R2021a on a MacBook Pro with Apple M1 chip and 8 GB of RAM.

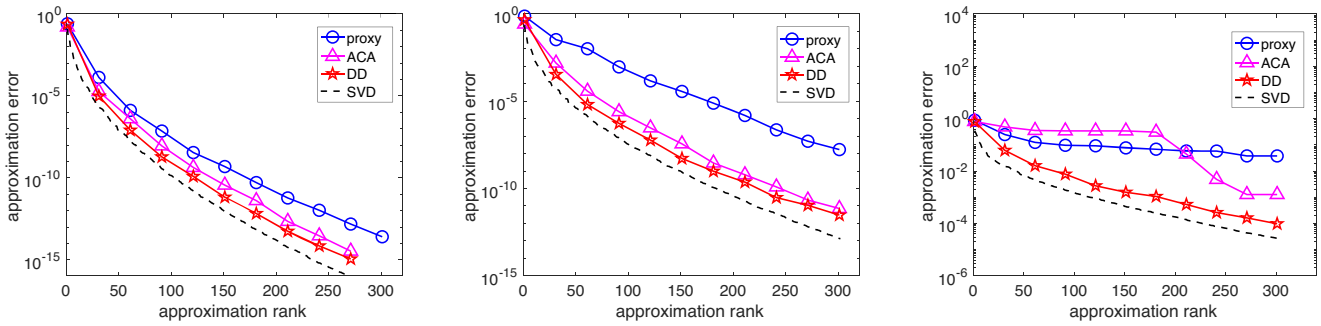
### 4.1 | Data on a manifold in three dimensions

For data in low dimensional ambient space, for example, $d = 3$, there exist several effective methods for compressing kernel matrices. However, their efficiency may decrease when the separation between the sets $X$ and $Y$ decreases and when the data lies on a manifold rather than be distributed relatively uniformly in the ambient space. To illustrate the advantages of the geometric approach in these cases, we use a sequence of three datasets as illustrated in Figure 4. In each dataset, $X$ and $Y$ each contain 1400 points, with 400 on each small cube and 600 on the hemisphere in Figure 4. From




(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

**FIGURE 4** Sequence of three datasets in three dimensions. From Datasets 1 to 3, $Y$ is a vertical shift of $X$ by 2.7, 2, and 0.5, respectively. The minimum distance between points in $X$ and points in $Y$ from Dataset 1 to 3 is 1, 0.43, 0.12, respectively.




(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

**FIGURE 5** Accuracy comparison of different methods for constructing low-rank factorizations on the kernel matrices defined by Datasets 1, 2, and 3 shown in Figure 4 and the kernel function $\kappa(x, y) = 1/|x - y|$.

Dataset 1 to 3, $Y$ is a vertical shift of $X$ by 2.7, 2, and 0.5, respectively. The minimum distance between points in $X$ and points in $Y$ from Datasets 1 to 3 is 1, 0.43, 0.12, respectively. The smallest bounding boxes for $X$ and $Y$ are well-separated in Dataset 1, adjacent in Dataset 2, and overlapping in Dataset 3. With these data, kernel matrices were constructed using the kernel function $\kappa(x, y) = 1/|x - y|$.

**Test 1. Robustness with respect to data geometries.** For above the settings, we compare the approximation error of the data-driven geometric approach with that of an algebraic method, ACA,[29] and proxy point method ('proxy').[27] For the data-driven method ('DD'), we construct a one-sided factorization (Algorithm 2) using farthest point sampling with sample size at most $2r$ for a rank-$r$ approximation. Namely, $2r$ points are chosen for $S_2$ and SRRQR is applied to $K_{XS_2}$. For the proxy point method, the sample size is 2000 for $\Omega_X$ (the smallest bounding box containing $X$) and 10000 for $\Omega_Y$, independent of the approximation rank. Figure 5 shows, for the various methods, the relative matrix approximation error as a function of the rank of the approximation. The relative error is defined as $\|K - \tilde{K}\|/\|K\|$, where $\tilde{K}$ denotes the low-rank approximation to $K$ and $\|\cdot\|$ denotes the 2-norm. The optimal relative approximation error as computed by the SVD is also shown.

We observe that all methods are effective for Dataset 1, with the data-driven method–DD–being the most efficient and most closely tracking the SVD approximation error. We remark that the large number of random samples in the bounding box is not effectively used in proxy point method when the data is unstructured. Hence it is computationally more expensive than DD and ACA in this experiment.

For Dataset 2, we are at the boundary at which hybrid methods are effective, that is, those methods that assume a separation of the bounding boxes for $X$ and $Y$. However, DD and ACA still closely track the SVD approximation error.

For Dataset 3, the points in $X$ and $Y$ are actually "intermingled" (overlapping bounding boxes). ACA might not effectively sample $X$ and $Y$ in this dataset, especially since the dataset contains disjoint clusters (points on a half-shell and points in small cubes). However, DD continues to closely track the SVD approximation error.

## 4.2 | Data in high-dimensional ambient space

The data-driven geometric approach can be efficient for data in high-dimensional ambient space, whereas many other existing low-rank compression methods have cost that is exponentially dependent on the dimensionality of ambient space. To demonstrate the data-driven approach for high-dimensional data, we use two datasets from the UCI machine learning repository:[*] Covertype ($n = 581{,}012$, $d = 54$) and Gas Sensor Array Drift ($n = 13{,}910$, $d = 128$). Each dataset is standardized to have mean zero and variance along each dimension equal to one. Instead of using the entire datasets, we choose $X$ and $Y$ to be two subsets of random samples, selected without replacement, with 8000 points for $X$ and 10,000 points for $Y$.

For both datasets, the kernel matrix $K_{XY}$ is a 8000-by-10,000 matrix associated with the Gaussian kernel $\kappa(x, y) = \exp(-|x - y|^2/\sigma^2)$. Since the bandwidth parameter $\sigma$ controls the smoothness of the kernel, we consider $\sigma$ from among three values, denoted as $\sigma_1, \sigma_2, \sigma_3$ and chosen to be 100%, 50%, 25% of the radius of $X$, respectively.

**Test 2. Different geometric selection schemes.** We first examine the effect of different geometric selection schemes (see Section 1) used to construct the two-sided low-rank factorization (2). It has been observed in Reference 42 that the approximation error can be extremely large if the subset is not chosen properly. In this case, to deal with the pseudoinverse, a stable implementation proposed in Reference 55 was used in Reference 42:

$$K_{XY} \approx (K_{XS_2} R_\epsilon^+)(Q^T K_{S_1 Y}), \tag{28}$$

where $K_{S_1 S_2} = QR$ is the QR factorization of $K_{S_1 S_2}$ and $R_\epsilon$ is derived from $R$ by truncating singular values smaller than $\epsilon$ in the SVD. It is also noted in Reference 42 that the above stabilization is *not* needed if the subset is well-chosen, that is, spread evenly over the data. For a detailed numerical study on the effect of stabilization, we refer to [42, sect. 5.2]. To make a fair comparison of different selection schemes, the stabilization in (28) is used in this experiment. Namely, in Algorithm 1, $K_{S_1 S_2}$ is replaced with its truncated QR factors: $Q$ and $R_\epsilon^+$.

For these experiments, we use the Gas Sensor dataset with a Gaussian kernel with $\sigma = \sigma_1 \approx 307.52$.

Figure 6 shows the low-rank approximation error (relative error as in Figure 5) versus approximation rank when different geometric selection schemes are used. We compare the following schemes: anchor net ('ANC'), farthest point sampling ('FPS'), uniform random sampling ('Unif'), as well as mixtures of uniform random sampling and FPS (some points are generated by FPS, the rest by random sampling). In the mixed scheme, random sampling is used to reduce the cost of FPS and we use the experiment to observe the effect of augmenting FPS samples with random samples. In these

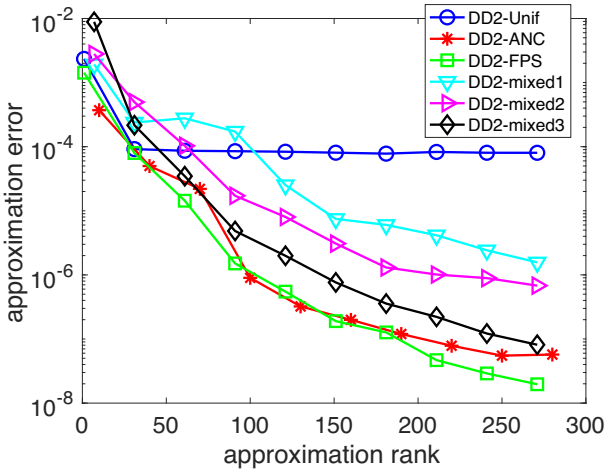



**FIGURE 6** Accuracy comparison of different geometric selection schemes for constructing two-sided data-driven low-rank factorizations on the kernel matrix defined by the Gas Sensor dataset (d = 128) and a Gaussian kernel with the bandwidth $\sigma_1 \approx 307.5$.

cases, the mixtures are denoted 'mixed1', 'mixed2', 'mixed3', for 5%, 10%, 50% FPS samples with the remainder of the samples selected by uniform random sampling. We observe that ANC and FPS perform similarly. These methods have a clear advantage over pure random sampling, suggesting that the data has structure to be exploited that is hidden from pure random sampling. However, random sampling can reduce the cost of a pure FPS method. The accuracy of ANC and FPS is attributed to the use of evenly spaced points. The generation of evenly distributed points is studied in discrepancy theory for the unit cube (cf. References 56,57) and recently extended to general geometries using deep neural networks (cf. References 58,59). We also note that the approximation error for 'DD2-Unif' does not improve after approximation rank about 50, creating the "flat" portion in the plot. In fact, this is due to the stabilization (28) that prevents the approximation error from "blowing up".

**Test 3. One-sided versus two-sided low-rank factorization.** With the same dataset and kernel as immediately above, we compared the approximation error for one-sided and two-sided low-rank factorization. Figure 7 shows these results. The one- and two-sided cases are denoted as 'DD1' and 'DD2', respectively, and each is tested with 'Unif', 'ANC', and 'FPS' geometric selection.

We observe that one-sided factorization is generally more accurate. This is consistent with the theoretical results in Theorems 2 and 5, where the two-sided approximation error estimate contains the norm of a pseudoinverse matrix while the one-sided approximation estimate doesn't contain any matrix norm.

We notice again the stagnating accuracy for 'DD2-Unif' when the approximation rank is larger than 50. On the contrary, "DD1-Unif" gives effective error reduction as the approximation rank increases. The difference reveals the fact that the two-sided factorization is *not* as numerically stable as the one-sided factorization. It should be noted, however, that the one-sided factorization is more expensive to compute than the two-sided factorization. The one-sided factorization uses geometric selection on only $X$ or only $Y$ rather than both $X$ and $Y$ and applies algebraic compression to a much larger intermediate matrix than the two-sided factorization.

**Test 4. Comparison with ACA and robustness with respective to kernel parameters.** There exist a few low-rank compression algorithms that are both efficient in the high-dimensional case and able to handle intermingled data. One notable method is the ACA method,[29] which does not require access to the full kernel matrix, has linear complexity, and produces a one-sided factorization. We thus now compare the data-driven geometric approach with ACA. Specifically, we use the data-driven approach to compute a one-sided low-rank factorization with ANC and FPS geometric selection schemes, corresponding to "DD-ANC" and "DD-FPS" in Figures 8–10.

Figures 8 and 9 show the approximation errors for the Covertype and Gas Sensor datasets, respectively, each with three values of bandwidth $\sigma$ for the Gaussian kernel. For the Covertype dataset, the data-driven methods yield much better accuracy than ACA for all choices of $\sigma$. The accuracy of ACA stagnates as the approximation rank is increased, suggesting that clusters in the dataset have prevented ACA from selecting rows and columns that help represent the kernel matrix.

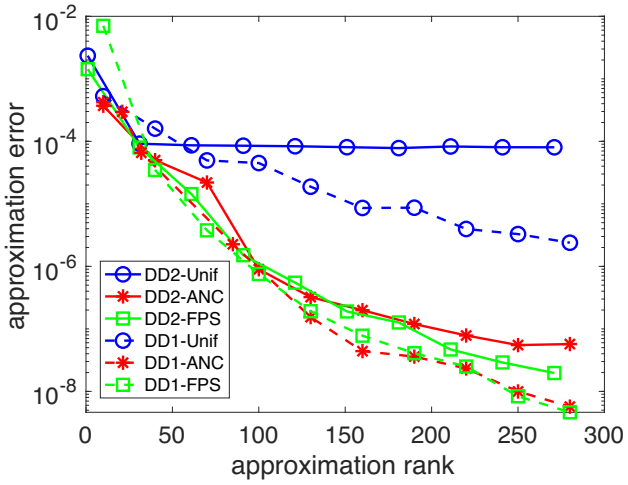
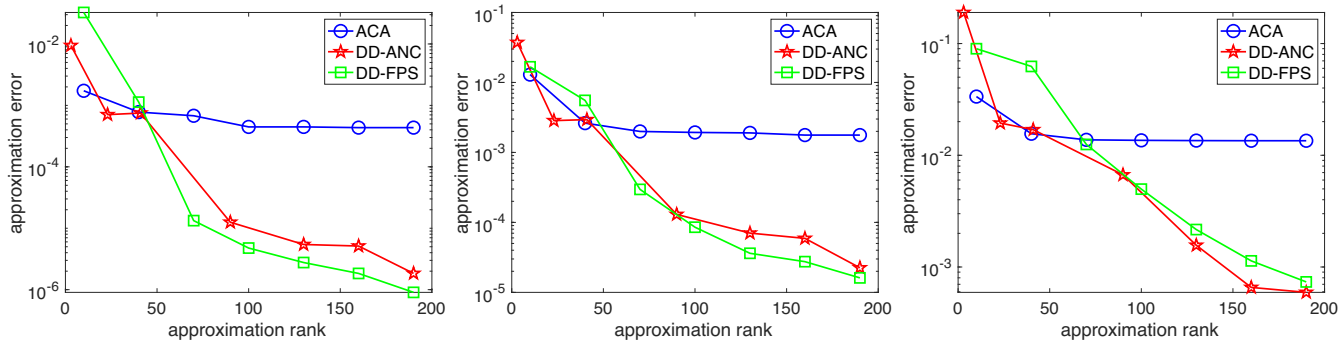
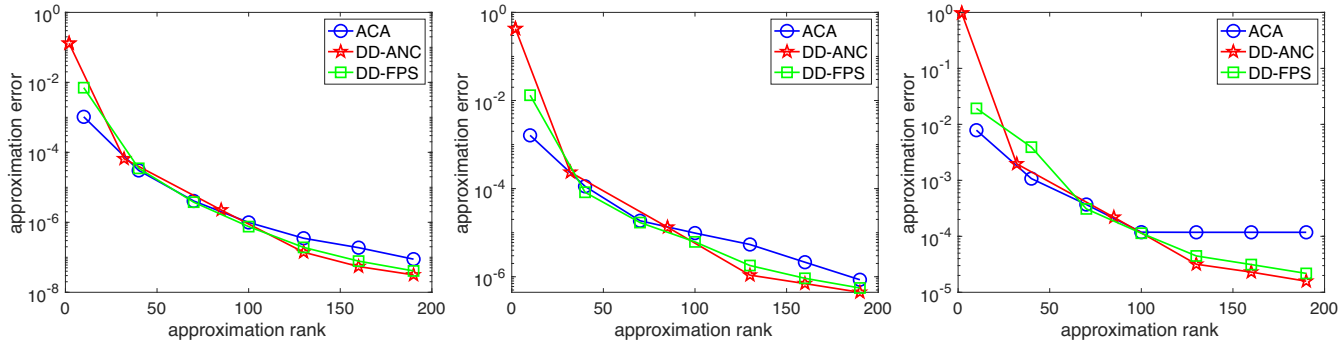



**FIGURE 7** Accuracy comparison of one-sided (dashed lines) versus two-sided (solid lines) data-driven factorizations on the kernel matrix defined by the Gas Sensor dataset (d = 128) and a Gaussian kernel with the bandwidth $\sigma_1 \approx 307.5$.




(a) $\sigma = \sigma_1 \approx 74.46$ (b) $\sigma = \sigma_2 \approx 37.23$ (c) $\sigma = \sigma_3 \approx 14.89$

**FIGURE 8** Accuracy comparison of one-sided data-driven factorizations (DD-ANC and DD-FPS) with ACA on kernel matrices defined by the Covertype dataset (d =54) and Gaussian kernel with three different bandwidths $\sigma$.



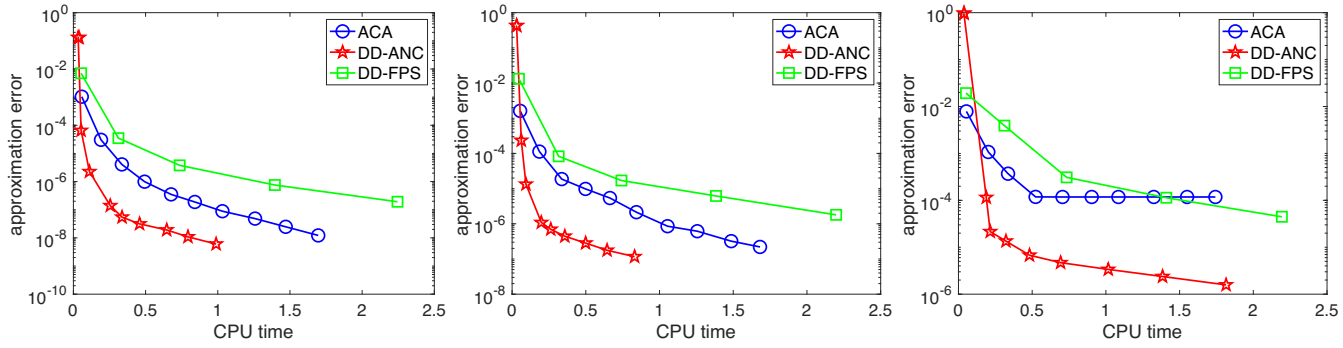

(a) $\sigma = \sigma_1 \approx 307.52$ (b) $\sigma = \sigma_2 \approx 153.76$ (c) $\sigma = \sigma_3 \approx 61.50$

**FIGURE 9** Accuracy comparison of one-sided data-driven factorizations (DD-ANC and DD-FPS) with ACA on the kernel matrices defined by the Gas Sensor dataset (d =128) and the Gaussian kernel with three different bandwidths $\sigma$.

For the Gas Sensor dataset, all methods behave similarly for a large bandwidth $\sigma_1$. For a smaller bandwidth $\sigma_3$, ACA displays stagnation in accuracy for approximation rank greater than 100. The smaller bandwidth makes the Gaussian kernel less smooth, and accentuates the effect of clusters in the data. These issues in ACA have been explored previously.[30,31] In general, these issues reflect the challenge in approximating kernel matrices associated with high ambient dimensions but possibly lower dimensional structures within these dimensions.

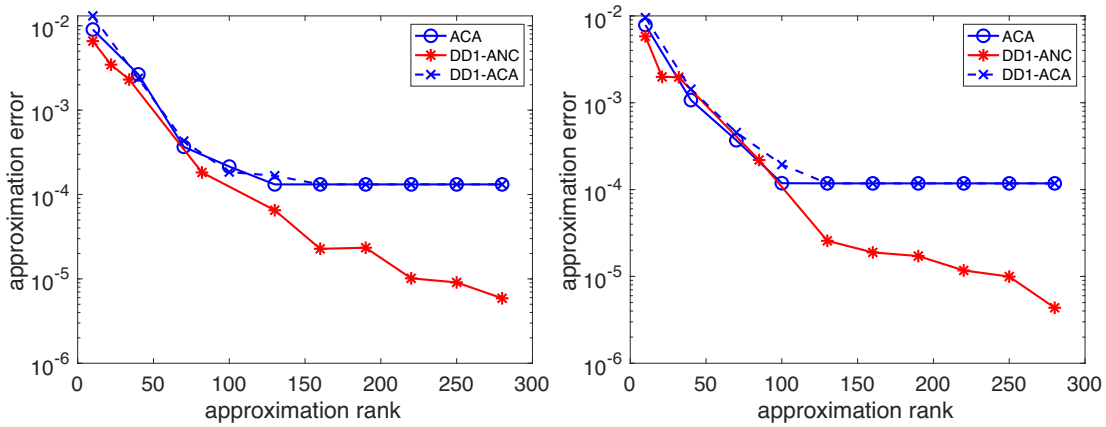
Figure 10 shows timings for computing the low-rank approximations. Although our timings are limited to MATLAB execution, the results indicate that a geometric approach using a fast geometric selection scheme (e.g., ANC) can be faster than ACA for the same approximation error. The results also suggest that ANC is significantly faster than FPS.

**Test 5. Comparison of algebraically generated subsets and geometrically generated subsets.** ACA[28,29] performs a column-pivoted partial LU decomposition where the pivots are chosen algebraically based on the residual of each rank-1 approximation in the sequential process. The resulting triangular factorization (LU) is mathematically equivalent to $K_{XS_2}K_{S_1S_2}^{-1}K_{S_1Y}$ where $K_{S_1S_2}$ is a square matrix and the subsets (corresponding to the pivots) $S_1$, $S_2$ are generated by ACA (cf. [ 28, Lemma 3]). Hence we can use ACA to generate subsets for Algorithms 1 and 2. Now, given the proposed geometric approach and the algebraic approach via ACA for generating subsets, a natural question is which approach yields better performance in practice. In this experiment, we investigate the quality of subsets by comparing the following methods: (1) ACA; (2) one-sided factorization in Algorithm 2 with subset $S_2$ generated by ACA ('DD1-ACA'); (3) one-sided factorization with subset generated by anchor net ('DD1-ANC'). We use the Gas Sensor dataset ($d = 128$) and consider both symmetric positive definite (SPD) matrix and rectangular matrix. The SPD matrix is the Gaussian kernel matrix $K_{XX}$ with $X$ containing 8000 random samples from the Gas Sensor dataset. For the rectangular matrix $K_{XY}$, we choose $X$ and




(a) $\sigma = \sigma_1 \approx 307.52$ (b) $\sigma = \sigma_2 \approx 153.76$ (c) $\sigma = \sigma_3 \approx 61.50$

**FIGURE 10** Time comparison of one-sided data-driven factorizations (DD-ANC and DD-FPS) with ACA on the kernel matrices defined by the Gas Sensor dataset (d =128) and the Gaussian kernel with three different bandwidths $\sigma$. CPU timings (in seconds) are the average of 10 runs.



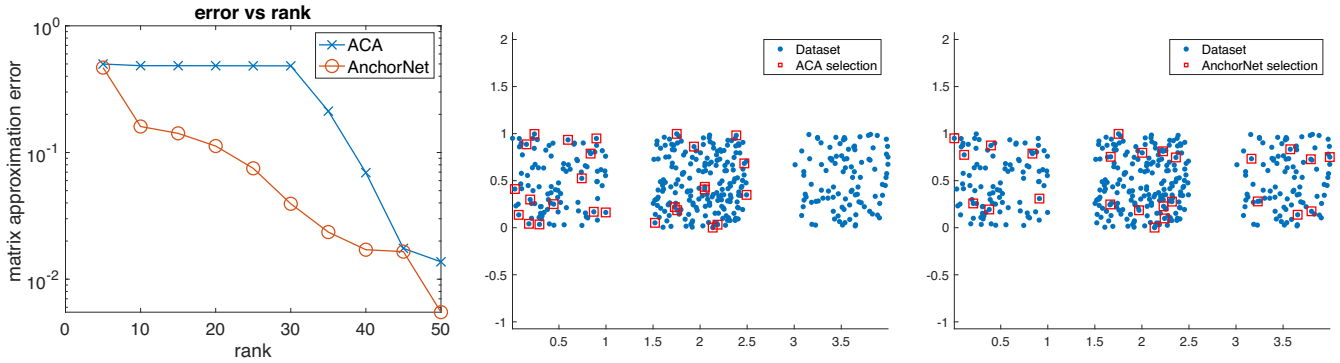

(a) SPD Gaussian kernel matrix $K_{XX}$ (b) Rectangular Gaussian kernel matrix $K_{XY}$

**FIGURE 11** Accuracy comparison of ACA, one-sided factorizations with ACA-generated subsets ('DD1-ACA') and anchor net-generated subsets ('DD1-ANC') on the Gaussian kernel matrices with Gas Sensor dataset ($d = 128$).

$Y$ to contain 8000 and 10,000 samples, respectively, as in **Test 4**. The bandwidth paragraph is chosen as $\sigma = \sigma_3 \approx 61.50$. The result is shown in Figure 11.

It is clearly seen from Figure 11 that, for this problem, ACA and 'DD1-ACA' yield almost the same performance, indicating that the subset generated by ACA (or, algebraically, pivots) does *not* yield an accurate low-rank approximation regardless of whether the underlying matrix is SPD. The geometric method with anchor net, on the other hand, generates better subsets with more accurate low-rank approximations. Considering the time efficiency demonstrated in Figure 10, we see that the geometric approach gives overall better performance in terms of accuracy, speed, and robustness.

To provide a straightforward illustration of the subset selected by ACA, we performed an experiment in two dimensions ($X \subset \mathbb{R}^2$) for the Gaussian kernel $\kappa(x, y) = \exp(-|x - y|^2/0.25)$. The dataset $X$ contains 400 points splitted into three clusters (left to right) with 100, 200, and 100 points respectively. Thus the kernel matrix $K_{XX}$ is a 400-by-400 SPD matrix. The dataset $X$ is shown as blue points in Figure 12b,c. We compare the points generated by ACA and AnchorNet and show the corresponding low-rank approximation error measured by relative error in matrix 2-norm. The result is shown in Figure 12. We see from Figure 12a that when the rank is smaller than 30, ACA entirely fails to improve the approximation accuracy despite the rank increase. To understand this from a geometric point of view, we show the scatter plots in (b) and (c) for the case when the rank equals 25. It is clearly seen that the points selected by ACA are "locked" in the first two clusters in $X$ with no point selected in the third cluster. This results in the stagnation of accuracy. Algebraically, this "locking" phenomenon is analogous to performing Gaussian elimination *only* on the first two diagonal blocks of a




(a) Error-rank plot for approximating $K_{XX}$

(b) $X$ and 25 points selected by ACA (error=0.48)

(c) $X$ and 25 points selected by AnchorNet (error=0.07)

**FIGURE 12** Comparison of ACA-based selection and AnchorNet-based selection.

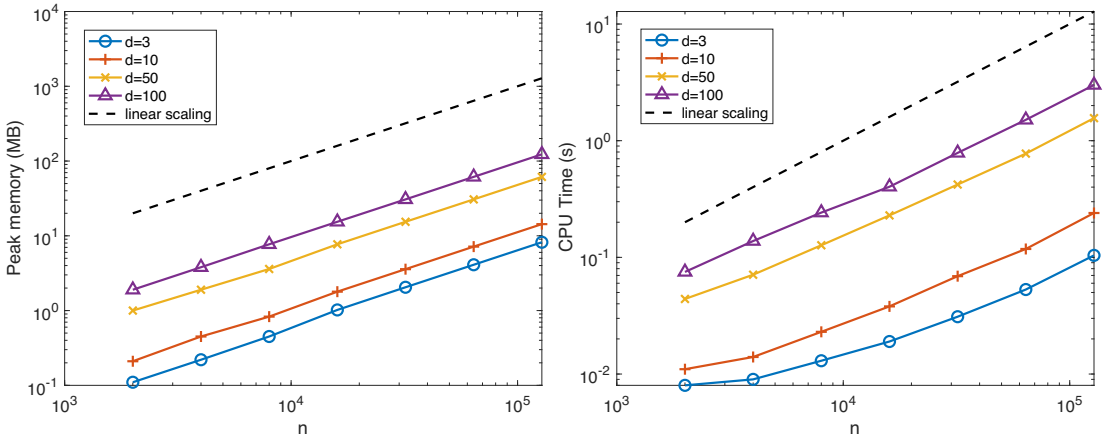



**FIGURE 13** Linear scaling tests of the data-driven factorization method on the kernel matrices defined by the synthetic data $X, Y$ sampled from the uniform distribution over $[0,1]^d$ and $[2,3]^d$, respectively, and the kernel function $\log|x-y|$. Left: peak memory use. Right: CPU time (average of 10 runs).

matrix that is composed of three diagonal blocks of similar rank structure. On the contrary, AnchorNet generates points from all three clusters in a more balanced fashion and achieves consistently better accuracy than ACA according to Figure 12a.

## 4.3 | Complexity test

In this section, we perform experiments to investigate the complexity of the proposed data-driven methods with respect to the size of the data.

**Test 6. Linear complexity with respect to data size.** We consider approximating an $n$-by-$n$ kernel matrix $K_{XY}$ with increasing matrix size $n$ and dimension $d$. We consider dimensions $d = 3, 10, 50, 100$ and generate synthetic data $X, Y$ in $\mathbb{R}^d$. $X$ and $Y$ are randomly sampled from the uniform distribution over $[0,1]^d$ and $[2,3]^d$, respectively. The kernel function is chosen as $\log|x-y|$. We use the one-sided factorization in Algorithm 2 based on farthest point sampling. In Figure 13, we report the peak memory use and timing for our method as $n$ increases. The CPU time is computed as an average over ten repeated runs. For all cases in Figure 13, the relative low-rank approximation error is around $2 \times 10^{-4}$.

It is easily seen from Figure 13 that, for each dimension $d$, the peak memory and timing both increase approximately linearly as a function of $n$, that is, the number of points in $X$ or $Y$.

## 4.4 | Kernel test

To show that the proposed data-driven approach can be applied to various kinds of kernel functions defined in high dimensions, we consider six different kernel functions in Table 1 and a dataset in 561 dimensions. The kernel $\kappa_5(x,y)$ in Table 1 is non-symmetric. We demonstrate the generality and accuracy of the data-driven approach by comparing to ACA.

**Test 7. Approximating various types of kernel functions.** We use a high dimensional dataset from the UCI Machine Learning Repository: *Smartphone Dataset for Human Activity Recognition in Ambient Assisted Living*[†]. The training data contains $n = 4252$ instances with $d = 561$ attributes. We choose $X$ to be the standardized data, and define $Y = \frac{2R}{\sqrt{d}} + X$, where $R = \max_{x\in X} \text{dist}(x, 0)$. By construction, $X$ and $Y$ do not overlap and kernel functions in Table 1 are all well-defined over $X \times Y$.

The kernel functions $\kappa_1(x,y), \ldots, \kappa_6(x,y)$ in the experiment are given in Table 1.

In Table 2, we report the relative approximation error $\|K - UV\|/\|K\|$ with respect to the approximation rank $r$, which is equal to the number of columns of $U$. Three methods are compared: ACA, the data-driven compression in Algorithm 2 with farthest point sampling ("DD-FPS") or anchor net method ("DD-ANC").

We see from Table 2 that DD-ANC achieves the best result for all kernels and ranks tested. For the same approximation rank $r$, the accuracy of DD-ANC is noticeably higher than that of DD-FPS and ACA. DD-FPS outperforms ACA

**TABLE 1** Kernels used for experiment in Table 2.

| $\kappa_1(x,y)$ | $\kappa_2(x,y)$ | $\kappa_3(x,y)$ | $\kappa_4(x,y)$ | $\kappa_5(x,y)$ | $\kappa_6(x,y)$ |
|---|---|---|---|---|---|
| $\lvert x-y\rvert$ | $\log\lvert x-y\rvert$ | $\left(1+\left\lvert\frac{x-y}{R}\right\rvert^2\right)^{-1}$ | $\exp\left(-\frac{1}{1-c\lvert x-y\rvert^2}\right)$ | $\frac{x_1}{\lvert x-y\rvert}$ | $x\cdot y+(x\cdot y)^2+(x\cdot y)^3$ |

*Note*: In $\kappa_4$, the constant $c = \frac{0.8}{\max_{x\in X, y\in Y}\lvert x-y\rvert^2}$, and in $\kappa_5$, $x_1$ denotes the first entry of the vector $x \in \mathbb{R}^d$.

**TABLE 2** Rank-$r$ approximation accuracy of the data-driven factorizations (DD-ANC and DD-FPS) and ACA on the kernel matrices defined by the smartphone dataset ($d = 561$) and six kernel functions shown in Table 1.

| $r$ | | 10 | 50 | 90 | 130 | 170 | 210 | 250 |
|---|---|---|---|---|---|---|---|---|
| $\kappa_1(x,y)$ | ACA | 2.4E-3 | 4.8E-4 | 1.2E-4 | 8.2E-5 | 3.1E-5 | 1.7E-5 | 8.5E-6 |
| | DD-FPS | 1.3E-3 | 2.4E-4 | 1.4E-4 | 4.2E-5 | 3.5E-5 | 1.4E-5 | 4.7E-6 |
| | DD-ANC | 3.5E-4 | 9.0E-5 | 3.8E-5 | 1.9E-5 | 9.9E-6 | 4.6E-6 | 2.7E-6 |
| $\kappa_2(x,y)$ | ACA | 2.2E-4 | 6.1E-5 | 4.8E-5 | 2.3E-5 | 6.7E-6 | 3.6E-6 | 2.7E-6 |
| | DD-FPS | 1.7E-4 | 4.5E-5 | 2.7E-5 | 7.5E-6 | 5.1E-6 | 2.4E-6 | 1.1E-6 |
| | DD-ANC | 5.9E-5 | 1.6E-5 | 6.4E-6 | 3.6E-5 | 1.9E-6 | 9.0E-7 | 5.5E-7 |
| $\kappa_3(x,y)$ | ACA | 2.5E-3 | 9.3E-4 | 3.2E-4 | 1.8E-4 | 6.3E-5 | 4.3E-5 | 2.9E-5 |
| | DD-FPS | 5.2E-3 | 9.8E-4 | 2.5E-4 | 1.2E-4 | 6.8E-5 | 3.8E-5 | 2.0E-5 |
| | DD-ANC | 8.3E-4 | 1.3E-4 | 6.4E-5 | 3.9E-5 | 1.8E-5 | 9.9E-6 | 6.0E-6 |
| $\kappa_4(x,y)$ | ACA | 1.2E-2 | 8.7E-4 | 3.3E-4 | 1.4E-4 | 7.1E-5 | 4.9E-5 | 4.8E-5 |
| | DD-FPS | 3.6E-2 | 2.5E-3 | 3.9E-4 | 1.7E-4 | 1.0E-4 | 4.6E-5 | 1.8E-5 |
| | DD-ANC | 1.8E-3 | 2.8E-4 | 1.2E-4 | 6.2E-5 | 3.5E-5 | 1.7E-5 | 9.0E-6 |
| $\kappa_5(x,y)$ | ACA | 9.1E-4 | 1.7E-4 | 7.7E-5 | 5.7E-5 | 2.2E-5 | 1.1E-5 | 7.1E-6 |
| | DD-FPS | 4.0E-4 | 1.1E-4 | 4.9E-5 | 2.1E-5 | 1.1E-5 | 4.1E-6 | 2.1E-6 |
| | DD-ANC | 2.9E-4 | 5.7E-5 | 3.2E-5 | 1.3E-5 | 5.4E-6 | 2.0E-6 | 1.2E-6 |
| $\kappa_6(x,y)$ | ACA | 1.3E-1 | 8.9E-3 | 5.1E-3 | 2.8E-3 | 2.2E-3 | 1.8E-3 | 1.7E-3 |
| | DD-FPS | 1.8E-2 | 3.7E-3 | 2.0E-3 | 1.2E-3 | 9.8E-4 | 7.6E-4 | 6.3E-4 |
| | DD-ANC | 1.2E-2 | 2.7E-3 | 1.1E-3 | 7.5E-4 | 6.8E-4 | 5.4E-4 | 4.5E-4 |

for almost all cases except $\kappa_4(x, y)$ when $r = 10, \ldots, 170$. Together with Test 4 in Section 4.2, the results show that the proposed fast data-driven approach is not only more robust, but also more accurate for the high dimensional dataset with general kernels. Compared to existing methods, one advantage of the data-driven method is that, for the same dataset and fixed rank $r$, the geometric selection is performed only *once* and can be used for different kernel functions or kernel function parameters. This can hardly be achieved by methods that require kernel function evaluation as the first step of the compression. For example, for algebraic methods such as ICA (Incomplete Cross Approximation[60]) and ACA, if the kernel function changes, the pivots need to be computed anew. In Table 2, for each $r$, ACA computes pivots six times for six kernels, while DD-FPS and DD-ANC each only select one subset, which is used for all six kernel functions.

## 5 | CONCLUSION

For compressing low-rank kernel matrices where sets of points $X$ and $Y$ are available, it appears appealing to use subsets of $X$ and $Y$ that capture the geometry of $X$ and $Y$. This paper presented theoretical justification and numerical tests that argue for choosing points such that no original point in $X$ (or $Y$) is very far from a point chosen for the subset. If these subsets can be selected in linear time, then the overall compression algorithm can be performed in linear time, which is optimal for kernel matrices. We demonstrated effective low-rank compression for both low and high dimensional datasets using geometric selection based on farthest point sampling and the anchor net method, which are both linear scaling. It is possible that even more sophisticated linear scaling schemes for selecting subsets can lead to even better approximation accuracy with the same number of selected points, especially in the high-dimensional case.

### ACKNOWLEDGMENTS

We would like to thank anonymous referees for their valuable suggestions. The research of Difeng Cai and Yuanzhe Xi is supported by NSF award OAC 2003720. The research of Edmond Chow is supported by NSF award OAC 2003683.

### CONFLICT OF INTEREST STATEMENT

This study does not have any conflicts to disclose.

### DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in UCI machine learning repository at https://archive.ics.uci.edu/ml/index.php.

### ENDNOTES

*https://archive.ics.uci.edu/ml/index.php
†https://archive.ics.uci.edu/ml/machine-learning-databases/00364/

### ORCID


*Yuanzhe Xi* https://orcid.org/0000-0003-0361-0931

### REFERENCES

1. Kress R. Linear integral equations. Applied mathematical sciences. New York: Springer; 2013.
2. Hsiao GC, Wendland WL. Boundary integral equations. Applied mathematical sciences. Berlin, Heidelberg: Springer; 2008.
3. Atkinson KE. The numerical solution of the eigenvalue problem for compact integral operators. Trans Am Math Soc. 1967;129(3):458–65.
4. Rokhlin V. Rapid solution of integral equations of classical potential theory. J Comput Phys. 1985;60(2):187–207.
5. Cai D, Vassilevski PS. Eigenvalue problems for exponential-type kernels. Comput Methods Appl Math. 2020;20(1):61–78.
6. Barnes J, Hut P. A hierarchical O(N log N) force-calculation algorithm. Nature. 1986;324:446–9.
7. Greengard L, Rokhlin V. A fast algorithm for particle simulations. J Comput Phys. 1987;73:325–48.
8. Bishop CM. Pattern recognition and machine learning. New York: Springer; 2006.
9. Chow E, Saad Y. Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions. SIAM J Sci Comput. 2014;36(2):A588–608.
10. Vapnik V. The nature of statistical learning theory. Berlin, Germany: Springer; 2013.
11. Hackbusch W. Hierarchical matrices: algorithms and analysis. Springer Series in Computational Mathematics. Berlin Heidelberg: Springer; 2015.
12. Henderson J, He H, Malin BA, Denny JC, Kho AN, Ghosh J, et al. Phenotyping through semi-supervised tensor factorization (PSST). AMIA annual symposium proceedings. Volume 2018. American Medical Informatics Association; 2018. p. 564.

13. He H, Henderson J, Ho JC. Distributed tensor decomposition for large scale health analytics. The World Wide Web Conference. 2019 659–69.
14. He H, Xi Y, Ho JC. Fast and accurate tensor decomposition without a high performance computing machine. Paper presented at: 2020 IEEE international conference on big data (big data). IEEE. 2020 163–170.
15. Tyrtyshnikov E. Mosaic-skeleton approximations. Calcolo. 1996;33(1):47–57.
16. Goreinov SA, Tyrtyshnikov EE. The maximal-volume concept in approximation by low-rank matrices. Contemporary Math. 2001;280:47–52.
17. Goreinov SA, Tyrtyshnikov EE. Quasioptimality of skeleton approximation of a matrix in the Chebyshev norm. Doklady mathematics. Springer; 2011;83:374–5.
18. Cortinovis A, Kressner D. Low-rank approximation in the Frobenius norm by column and row subset selection. SIAM J Matrix Anal Appl. 2020;41(4):1651–73.
19. Mahoney MW, Drineas P. CUR matrix decompositions for improved data analysis. Proc Natl Acad Sci. 2009;106(3):697–702.
20. Anderson D, Du S, Mahoney M, Melgaard C, Wu K, Gu M. Spectral gap error bounds for improving CUR matrix decomposition and the Nyström method. Artificial intelligence and statistics; 2015. p. 19–27.
21. Hackbusch W, Nowak ZP. On the fast matrix multiplication in the boundary element method by panel clustering. Numer Math. 1989;54(4):463–91.
22. Anderson CR. An implementation of the fast multipole method without multipoles. SIAM J Sci Stat Comput. 1992;13(4):923–47.
23. Sun X, Pitsianis NP. A matrix version of the fast multipole method. SIAM Rev. 2001;43(2):289–300.
24. Börm S, Grasedyck L, Hackbusch W. Introduction to hierarchical matrices with applications. Eng Anal Bound Elem. 2003;27(5):405–22.
25. Cai D, Chow E, Erlandson L, Saad Y, Xi Y. SMASH: structured matrix approximation by separation and hierarchy. Numer Linear Algebra Appl. 2018;25(6):e2204.
26. Gillman A, Young PM, Martinsson PG. A direct solver with O(N) complexity for integral equations on one-dimensional domains. Front Math China. 2012;7(2):217–47.
27. Xing X, Chow E. Interpolative decomposition via proxy points for kernel matrices. SIAM J Matrix Anal Appl. 2020;41(1):221–43.
28. Bebendorf M. Approximation of boundary element matrices. Numer Math. 2000;86(4):565–89.
29. Bebendorf M, Rjasanow S. Adaptive low-rank approximation of collocation matrices. Comput Secur. 2003;70(1):1–24.
30. Börm S, Grasedyck L. Hybrid cross approximation of integral operators. Numer Math. 2005;101(2):221–49.
31. Cambier L, Darve E. Fast low-rank kernel matrix factorization using skeletonized interpolation. SIAM J Sci Comput. 2019;41(3):A1652–80.
32. Williams CK, Seeger M. Using the Nyström method to speed up kernel machines. Advances in Neural Information Processing Systems; 2001. p. 682–8.
33. Cheng H, Gimbutas Z, Martinsson PG, Rokhlin V. On the compression of low rank matrices. SIAM J Sci Comput. 2005;26(4):1389–404.
34. Gu M, Eisenstat SC. Efficient algorithms for computing a strong rank-revealing QR factorization. SIAM J Sci Comput. 1996;17(4):848–69.
35. Smola A, Bartlett P. Sparse greedy Gaussian process regression. Adv Neural Inf Process Syst. 2000;13:598–604.
36. Lawrence N, Seeger M, Herbrich R. Fast sparse Gaussian process methods: the informative vector machine. Adv Neural Inf Process Syst. 2002;15:625–32.
37. Snelson E, Ghahramani Z. Sparse Gaussian processes using pseudo-inputs. Adv Neural Inf Process Syst. 2005;18:1257–64.
38. Xu Z, Cambier L, Rouet FH, L'Eplatennier P, Huang Y, Ashcraft C, et al. Low-rank kernel matrix approximation using skeletonized interpolation with endo-or exo-vertices. arXiv preprint arXiv:180704787. 2018.
39. Eldar Y, Lindenbaum M, Porat M, Zeevi YY. The farthest point strategy for progressive image sampling. IEEE Trans Image Process. 1997;6(9):1305–15.
40. Peyré G, Cohen LD. Geodesic remeshing using front propagation. Int J Comput Vision. 2006;69(1):145–56.
41. Schlömer T, Heck D, Deussen O. Farthest-point optimized point sets with maximized minimum distance. Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics. 2011 135–42.
42. Cai D, Nagy J, Xi Y. Fast deterministic approximation of symmetric indefinite kernel matrices with high dimensional datasets. SIAM J Matrix Anal Appl. 2022;43(2):1003–28.
43. Hackbusch W, Khoromskij BN, Sauter SA. On $\mathscr{H}^2$-matrices. In: Bungartz HJ, RHW H, Zenger C, editors. Lectures on Applied Mathematics. Berlin: Springer; 2000. p. 9–29.
44. Erlandson L, Cai D, Xi Y, Chow E. Accelerating parallel hierarchical matrix-vector products via data-driven sampling. Paper presented at: 2020 IEEE international parallel and distributed processing symposium (IPDPS). 2020 749–758.
45. Bauer M, Bebendorf M, Feist B. Kernel-independent adaptive construction of H 2-matrix approximations. Numer Math. 2022;150(1):1–32.
46. Madych W, Nelson S. Bounds on multivariate polynomials and exponential error estimates for multiquadric interpolation. J Approx Theory. 1992;70(1):94–114.
47. Zienkiewicz OC, Zhu JZ. A simple error estimator and adaptive procedure for practical engineerng analysis. Int J Numer Methods Eng. 1987;24(2):337–57.
48. Verfürth R. A posteriori error estimation and adaptive mesh-refinement techniques. J Comput Appl Math. 1994;50(1):67–83.
49. Verfürth R. Robust a posteriori error estimates for stationary convection-diffusion equations. SIAM J Numer Anal. 2005;43(4):1766–82.
50. Braess D, Schöberl J. Equilibrated residual error estimator for edge elements. Math Comput. 2008;77(262):651–72.
51. Cai D, Cai Z. A hybrid a posteriori error estimator for conforming finite element approximations. Comput Methods Appl Mech Eng. 2018;339:320–40.

52. Cai D, Cai Z, Zhang S. Robust equilibrated a posteriori error estimator for higher order finite element approximations to diffusion problems. Numer Math. 2020;144(1):1–21.
53. Ainsworth M, Vejchodský T. Fully computable robust a posteriori error bounds for singularly perturbed reaction–diffusion problems. Numer Math. 2011;119(2):219–43.
54. Martinsson PG, Tropp JA. Randomized numerical linear algebra: foundations and algorithms. Acta Numer. 2020;29:403–572.
55. Nakatsukasa Y. Fast and stable randomized low-rank matrix approximation. arXiv preprint arXiv:200911392. 2020.
56. Niederreiter H. Random number generation and quasi-Monte Carlo methods. Philadelphia: SIAM; 1992.
57. Kuipers L, Niederreiter H. Uniform distribution of sequences. New York: Courier Corporation; 2012.
58. Cai D. Physics-informed distribution transformers via molecular dynamics and deep neural networks. J Comput Phys. 2022;468:111511.
59. Cai D, Ji Y, He H, Ye Q. AUTM flow: atomic unrestricted time machine for monotonic normalizing flows. Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence. Vol 180. PMLR; Xi Y. 2022 266–74.
60. Tyrtyshnikov E. Incomplete cross approximation in the mosaic-skeleton method. Comput Secur. 2000;64(4):367–80.

**How to cite this article:** Cai D, Chow E, Xi Y. Data-driven linear complexity low-rank approximation of general kernel matrices: A geometric approach. Numer Linear Algebra Appl. 2023;e2519. https://doi.org/10.1002/nla.2519