# Even Faster Kernel Matrix Linear Algebra via Density Estimation

Rikhav Shah
MIT

Sandeep Silwal
UW-Madison

Haike Xu
MIT

October 1, 2025

**Abstract**

This paper studies the use of *kernel density estimation* (KDE) for linear algebraic tasks involving the *kernel matrix* of a collection of $n$ data points in $\mathbb{R}^d$. In particular, we improve upon existing algorithms for computing the following up to $(1+\varepsilon)$ relative error: matrix-vector products, matrix-matrix products, the spectral norm, and sum of all entries. The runtimes of our algorithms depend on the dimension $d$, the number of points $n$, and the target error $\varepsilon$. Importantly, the dependence on $n$ in each case is far lower when accessing the kernel matrix through KDE queries as opposed to reading individual entries.

Our improvements over existing best algorithms (particularly those of [BIMW21]) for these tasks reduce the polynomial dependence on $\varepsilon$, and additionally decreases the dependence on $n$ in the case of computing the sum of all entries of the kernel matrix.

We complement our upper bounds with several lower bounds for related problems, which provide (conditional) quadratic time hardness results and additionally hint at the limits of KDE based approaches for the problems we study.

## 1 Introduction

Kernel functions and their associated kernel matrices have been extremely influential in practice, for example in "classical" machine learning via the so called kernel methods [HSS08, Mur12, BIS17], as well as in "modern" machine learning, where they lie at the heart of the *attention mechanism* in transformers [VSP+17, ZHDK23, AS23, HJK+24, IKSW25].

One common thread across these two eras of algorithms is the *computational burden* of working with kernel matrices: given a dataset $X$ as input, initializing kernel matrices exactly requires $\Omega(n^2 d)$ time with the naive approach, and assuming standard complexity conjectures such as SETH, one cannot hope to do better than $\Omega(n^2)$ time in the exact or very precise regimes when $d = \omega(\log n)$ [BIS17]. The quadratic bottleneck is especially salient in the case of large $n$, which is common in practice. Fortunately, SETH-based lower bounds typically only rule out extremely accurate computation (e.g. with additive error $e^{-\omega(\log n)}$), so there is still hope of reasonable *approximation* algorithms for kernel matrices, which has lead to a long and fruitful line of work; see [CS17, BCIS18, BIW19, CKNS20, ACSS20, BIMW21, BIK+23, CKW24, IKSW25] and references therein.

Much of the above algorithmic work on kernel matrices has been driven by a remarkable datastructure for computing *kernel density estimation* (KDE): it can compute a sum of $n$ kernel evaluations in sublinear $o(n)$ time (after linear pre-processing). This functionality is captured via the following definition.

**Definition 1.1** (Fast KDE). *A kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ admits a 'Fast KDE' algorithm, if given a set of $n$ points $X = \{x_1, \cdots, x_n\} \subset \mathbb{R}^d$, $\varepsilon, \mu \in (0,1)$, there exists a $p \in (0,1)$ (depending on $k$) such that we can process $X$ in $\widetilde{O}(dn/\mu^p)$ time and return a datastructure $\mathcal{D}$ such that for any query point $y \in \mathbb{R}^d$, with probability $1 - 1/\operatorname{poly}(n)$, we have*

$$\frac{1}{n}\sum_{i=1}^{n} k(y, x_i) \leq \mathcal{D}_X(y) \leq (1+\varepsilon) \cdot \frac{1}{n}\sum_{i=1}^{n} k(y, x_i) + \mu. \tag{1}$$

*The query time of $\mathcal{D}_X$ is $O(\frac{d \log n}{\varepsilon^2 \mu^p})$. In other words, we can estimate the sum of $n$ kernel evaluations up to a $1+\varepsilon$ multiplicative factor and $\mu$ additive error.*

As a baseline, it can be easily checked that random sampling can guarantee the above definition for $p = 1$ and a substantial body of research has been done to develop more sophisticated tools to obtain $p < 1$ for many popular kernel functions; this is summarized in Table 3.

A KDE query is intimately tied to the kernel matrix $K$ whose entries are given by $K_{ij} = k(x_i, x_j)$. Querying the KDE with $y = x_i$ approximates the $i$th row sum of $K$. Thus, many algorithmic works have used these queries to simulate downstream linear algebra algorithms for kernel matrices, without paying the naive $\Omega(n^2)$ time to construct them (e.g. see many of the aforementioned works).

The goals of our paper are to exploit the power of KDE datastructures to develop much faster algorithms for approximating fundamental quantities related to kernel matrices, which are of interest to both theory and practice, including matrix-vector product, kernel sums, top eigenvector computation, and more (see Section 2).

Our algorithms take a kernel function $k$ and a dataset $X$, and the kernel matrix $K$ is defined *implicitly*. Our only assumption is that $K$ is a symmetric positive semi-definite matrix with 1s along the diagonal and entries in $[0, 1]$. We do not specialize to any particular kernel function[1], nor do we make any other assumptions on the matrix $K$ such as having entries bounded away from 0 [BIK+23] or having the largest $n$ entries dominate the rest [IKSW25]. Our algorithms indirectly access the kernel matrix through KDE queries. This results in our running times depending on $p$, the exponent of the KDE datastructure in Definition 1.1, and thus they are valid for any kernel admitting such KDE data structures. Our work also has the advantage that any independent progress on KDE datastructures automatically implies faster algorithms.

## 2 Our Results and Comparison to Prior Works

Our results can be summarized as follows. We devise a faster algorithm for approximately computing non-negative kernel matrix vector products (Section 4). We then study the application of this algorithm to two fundamental problems: multiplication of kernel matrices and computation of the spectral norm (top eigenvalue) of kernel matrices (with a witnessing vector). For matrix multiplication, we obtain an improved error estimate over previous approaches (Section 4.2). For spectral norm estimation, we strengthen the analysis of "noisy" power iteration appearing in [BIMW21], resulting in the optimal asymptotic relationship between the error of the top eigenvalue desired and the error of the approximate matrix-vector products required (Section 5).

[1]The reader can have the function $k(x, y) = \exp(-\|x - y\|_2^2)$ in mind for simplicity, but our arguments generalize to any function in Table 3

This approximate matrix-vector product oracle can also be used to compute $\mathbf{1}^\top K\mathbf{1}$, i.e. the sum of the entries of the kernel matrix, but the additional structure of this problem can be exploited to give a distinct $o(n)$-time algorithm. We give an improved algorithm for this task (Section 6.1). Lastly, we present lower bounds, based on the SETH hypothesis, which show quadratic time hardness results for problems related to our upper bounds (Section 8). We also give evidence for why our non-negative matrix vector product and kernel sum algorithms could be (near) the limit of KDE based algorithms (Section 7).

Our upper and lower bound results are summarized in Table 1 and Table 2 respectively, and details follow.

| **Problem** | **Our Upper Bounds** | **Prior Work** |
|---|---|---|
| Non-negative Matrix-Vector Product:<br>Compute $Ky + e$ where $\|e\|_2 \leq \varepsilon\|Ky\|_2$ | $\widetilde{O}\left(\frac{n^{1+p}}{\varepsilon^{2+2p}}\right)$<br>Theorem 2.1 | $\widetilde{O}\left(\frac{n^{1+p}}{\varepsilon^{3+2p}}\right)$<br>[BIMW21] |
| Top Eigenvalue:<br>Output unit vector $y$ with $y^\top Ky \geq (1-\varepsilon)\lambda_1(K)$ | $\widetilde{O}\left(\frac{n^{1+p}}{\varepsilon^{3+2p}}\right)$<br>Theorem 2.2 | $\widetilde{O}\left(\frac{n^{1+p}}{\varepsilon^{7+4p}}\right)$<br>$\Omega(n)$<br>[BIMW21] |
| Non-negative Vector-Matrix-Vector Product:<br>Compute a $1+\varepsilon$ approximation to $y^\top Ky$ | $\widetilde{O}\left(\frac{n^{1+p}}{\varepsilon^{2+2p}}\right)$<br>Theorem 2.3 | - |
| Kernel Sum:<br>Compute a $1+\varepsilon$ approximation to $\mathbf{1}^\top K\mathbf{1}$. | $\widetilde{O}\left(\frac{n^{\frac{1}{2}+\frac{p}{2}}}{\varepsilon^4}\right)$<br>Theorem 2.4 | $\widetilde{O}\left(\frac{n^{\frac{2+5p}{4+2p}}}{\varepsilon^{\frac{8+6p}{2+p}}}\right)$<br>$\Omega(\sqrt{n})$<br>[BIMW21] |

Table 1: Summary of our upper bound results. All upper bounds have a dimension $d$ dependence (including those of [BIMW21]) which we omit for simplicity. We refer to the specific theorems for details.

## 2.1 Approximate Matrix-Vector Products

The first algorithmic problem we consider is the following: given a kernel matrix $K$ and a vector $y$, output an approximation of the matrix-vector product $Ky$. We specialize to the case where $y$ is an entry-wise non-negative vector, considered in prior work [BIMW21] (later in the discussion of our lower bounds in Section 2.4, we discuss the case where $y$ may have both positive and negative entries). The following defines our notion of approximation.

**Definition 2.1** ($\varepsilon$-Non-negative Matrix-Vector Product, [BIMW21])**.** *Given a non-negative vector $y$ and precision $\varepsilon$, return $x$ such that $x = Ky + e$ where $\|e\|_2 \leq \varepsilon\|Ky\|_2$ and $e$ is entry-wise non-negative.*

| Problem | Our Lower Bound |
|---|---|
| Number of row/column samples required to estimate $s(K)$ up to $1+\varepsilon$ factor | $\Omega\left(\frac{\sqrt{n}}{\varepsilon^2}\right)$ Theorem 2.5 |
| Compute $v^\top K w$ for $v, w$ non-negative vectors up to $n^{O(1)}$ factor | $\Omega(n^{2-\alpha})$ for any $\alpha > 0$ (SETH) Theorem 2.7 |
| Compute $s(K) = \mathbf{1}^\top K \mathbf{1}$ for "asymmetric" kernel matrix $K$ where rows and columns are indexed by different points | $\Omega(n^{2-\alpha})$ for any $\alpha > 0$ (SETH) Corollary 8.1 |
| Compute a $n^{O(1)}$ approximation to $\sigma_1(K)$, the top singular value of an "asymmetric" kernel matrix $K$ | $\Omega(n^{2-\alpha})$ for any $\alpha > 0$ (SETH) Corollary 8.2 |
| Compute a $n^{O(1)}$ approximation to a non-negative matrix vector product of an "asymmetric" kernel matrix $K$ | $\Omega(n^{2-\alpha})$ for any $\alpha > 0$ (SETH) Corollary 8.3 |
| Given an arbitrary $y \in \mathbb{R}^{n+1}$, compute $Ky + e$ with $\|e\|_2 \leq n^{-0.002}\|Ky\|_2$ where rows of $K$ are indexed by $X$ and columns by $X \cup \{0\}$ | $\Omega(n^{2-\alpha})$ for any $\alpha > 0$ (SETH) Theorem 2.6 |
| Given an arbitrary $y \in \mathbb{R}^{n+1}$ and $x \in \mathbb{R}^n$ which shares $n$ coordinates with $y$, compute $x^\top K y$ up to $n^{O(1)}$ factor, where the rows of $K$ are indexed by $X$ and columns by $X \cup \{0\}$ | $\Omega(n^{2-\alpha})$ for any $\alpha > 0$ (SETH) Theorem 7.2 |

Table 2: Summary of our lower bound results. We refer to the specific theorems for details.

The algorithm of [BIMW21] returns such a $x$ in Definition 2.1 in $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{3+2p}}\right)$ time[2]. Our first result obtains a faster algorithm for approximating non-negative matrix vector products for kernel matrices, removing a $1/\varepsilon$ factor in the bounds of [BIMW21].

**Theorem 2.1.** *Let $k$ be a kernel admitting a KDE datastructure of Definition 1.1. Let $K \in \mathbb{R}^{n \times n}$ be the associated kernel matrix for $n$ points in $d$ dimensions. There is an $\varepsilon$-non-negative approximate matrix-vector product algorithm (Algorithm 1) satisfying Definition 2.1 for $K$ running in time $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$.*

Our Theorem 2.1 actually achieves a stronger guarantee: it can approximate every coordinate of $Ky$ up to a multiplicative $1+\varepsilon$ factor and additive $\varepsilon/\sqrt{n}$ term. See Section 4 for a technical

[2]throughout the paper, $\widetilde{O}(\cdot)$ suppresses factors of the form $\log^{O(1)}(n/\varepsilon)$.

overview of our result. In Section 4.1 we additionally give an argument, suggesting that among a "natural" class of algorithms which achieve our per-coordinate error, the running time of $n^{1+p}$ maybe necessary.

A direct application of our non-negative matrix vector product is faster approximate matrix-multiplication algorithm between a kernel matrix $K$ and any entry-wise non-negative matrix $A$ (e.g. another kernel matrix). In $\widetilde{O}\left(\frac{dn^{2+p}}{\varepsilon^{2+2p}}\right)$, it outputs an approximation $B$ to $KA$ whose frobenius norm error $\|KA - B\|_F$ is bounded by $\varepsilon\|KA\|_F$, rather than $\varepsilon\|K\|_F\|A\|_F$, as is typical of standard approximate matrix multiplication algorithms. We refer to Section 4.2 for details.

## 2.2 Approximating the Top Eigenvalue

The second algorithmic problem we consider is the following: given a kernel matrix $K$, output a unit vector $u$ such that $u^\top Ku$ approximates the top eigenvalue $\lambda_1(K)$. We call $u$ the "witness" to the norm of $K$.[3] Many prior works have studied sublinear approximation algorithms in the more general setting, such as when $K$ is a symmetric or PSD matrix with bounded entries [RST10, MM15, BCJ20, SW23, BDD+24], without access to any kernel structure. The most relevant work in this line is [SW25], which shows how to achieve an *additive error* estimate of $\lambda_1(K)$. Namely, they sample a $O(1/\varepsilon) \times O(1/\varepsilon)$ principal sub-matrix of $K$ and use only those entries to compute a unit vector $u$ which satisfies

$$u^\top Ku \geq \lambda_1(K) - \varepsilon n. \tag{2}$$

in time $\widetilde{O}\left(1/\varepsilon^2\right)$. Without access to the underlying kernel structure, this is the best one can hope for [SW25]. In particular, a *relative error* guarantee is ruled out among sublinear algorithms: sampling $o(n^2)$ entries cannot distinguish between $K$ is the identity matrix and $K$ is the identity matrix plus a symmetric pair of off-diagonal 1s, which have norms 1 and $\sqrt{2}$ respectively.

In our setting, with access to the kernel structure, relative errors are possible. That is, one demands the vector $u$ returned by the algorithm satisfy[4]

$$u^\top Ku \geq (1-\varepsilon)\lambda_1(K). \tag{3}$$

This guarantee was achieved by [BIMW21] in the same setting as this paper. Their idea is to first exploit the kernel structure via KDE queries to devise a fast algorithm for approximating matrix-vector products in the sense of Definition 2.1. Then, they implement the power method using an approximate matrix-vector product algorithm in place of exact matrix-vector products.

It's important to note that the desired accuracy level of the computation of $\lambda_1(K)$ and the accuracy to which the underlying approximate matrix-vector products are performed may differ. To be more precise, even though we desire a $1-\varepsilon$ multiplicative error in computing $\lambda_1(K)$, we may potentially need to use a *different* level of error $\delta$ in the approximate non-negative matrix-vector computation of Definition 2.1 when simulating the power method (e.g. we could require the error vector $e$ satisfies $\|e\|_2 \leq \delta\|Ky\|_2$ on an input $y$).

The algorithm of [BIMW21] sets $\delta = O(\varepsilon^2)$, i.e. matrix vector products are computed *much* more accurately than the desired accuracy of the top eigenvalue estimate. Since a smaller $\delta$ requires a more expensive running time, meaning the choice of $\delta$ has a large impact in the final running time of the top eigenvector computation.

[3] Sometimes $u$ is referred to as an approximate top eigenvector, but we wish to avoid the suggestion that $u$ is at all close to the actual top eigenvector of $K$.

[4] This is strictly stronger when the entries of $K$ are bounded by 1 which is our case, as $\lambda_1(K) \leq n$

Carefully adjusting some expressions in their argument shows that $\delta = O(\varepsilon^{1.5})$ suffices, though this is the limit of their argument (see Example 1). Altogether, the final runtime of their algorithm is $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{7+4p}}\right)$, in particular, the $\varepsilon$ dependence is $1/\varepsilon^{7+4p}$.

We give an entirely different analysis of the power method, departing from the "classic" way it is understood, and show that $\delta = O(\varepsilon)$ both suffices and is necessary (see Section 5 for a technical overview of this result). When combined with our improvement to matrix-vector products, it removes a $1/\varepsilon^{4+2p}$ factor from their running time. Our final result is the following.

**Theorem 2.2.** *Let $\varepsilon \in (0,1)$ be sufficiently small. There exists an algorithm (Algorithm 2) which returns a scalar $\lambda$ and unit vector $u$ satisfying*

$$\lambda_1(K) \geq u^\top K u \geq (1 - (5/8)\varepsilon)\lambda_1(K).$$

$$(1 + \varepsilon/8)\lambda_1(K) \geq \lambda \geq (1 - \varepsilon/2)\lambda_1(K).$$

*The overall runtime of the algorithm is $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{3+2p}}\right)$.*

Ignoring log factors, our running time can be understood as running $1/\varepsilon$ iterations of noisy-power iteration using $\delta = O(\varepsilon)$ precision for a non-negative matrix-vector product, each of which takes $\approx dn^{1+p}/\varepsilon^{2+p}$ time. In Proposition 5.3, we give a lower bound, showing that $\Omega\left(\frac{\log n}{\varepsilon}\right)$ iterations are necessary for *any* algorithm that works by post-processing the data collected during the noisy power method (noisy matrix vector products starting from a fixed vector). We leave the question of whether one can prove this lower bound for other, possibly adaptive, sequences of matrix-vector queries as an interesting open problem.

### 2.3 Vector-Matrix-Vector Products and Kernel Sum

Another type of structured matrix queries that are interesting are vector-matrix-vector queries [RWZ20]: given a vector $v$, output an approximation to $v^\top K v$. We consider the case where $v$ is entry-wise non-negative (the case of having both positive and negative entries will be discussed in Section 2.4). A simple reduction shows that computing $Kv$ up to the precision of Definition 2.1 and then taking the inner product of the noisy output with $v$ gives the following result, with running time dominated by the noisy matrix vector computation.

**Theorem 2.3.** *Let $k$ be a kernel admitting a KDE datastructure of Definition 1.1. Let $K \in \mathbb{R}^{n \times n}$ be the associated kernel matrix for $n$ points in $d$ dimensions and let $v \in \mathbb{R}^n$ be an entry-wise non-negative vector. We can compute $v^\top K v$ up to a $1+\varepsilon$ multiplicative factor in time $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$.*

Prior to our work, one could obtain an identical approximation result but with an increase of a $1/\varepsilon$ factor in the running time via the noisy matrix-vector product algorithm of [BIMW21].

However, substantially different algorithms are needed special case of vector-matrix-vector products where we wish to compute $\mathbf{1}^\top K \mathbf{1} = s(K)$, the sum of all the entries in $K$. This special case has been highlighted in practice in applications such as kernel alignment (a popular measure of similarity between kernel matrices) [CSTEK01] and in some statistics applications [ABB+23]. In this special case, there is hope of going beyond the $\Omega(n)$ lower bound needed to read general $v$. Indeed, [BIMW21] gave a much faster algorithm which approximates $s(K)$ up to a $1+\varepsilon$ factor in $\widetilde{O}\left(\frac{n^{\frac{2+5p}{4+2p}}}{\varepsilon^{\frac{8+6p}{2+p}}}\right)$ time. Our improved result is the following.

**Theorem 2.4.** *Let $k$ be a kernel function admitting a KDE datastructure of Definition 1.1. Let $K \in \mathbb{R}^{n \times n}$ be the associated kernel matrix for $n$ points in $d$ dimensions, and $\varepsilon \in (0,1)$. In time $\widetilde{O}\left(\frac{n^{\frac{1}{2}+\frac{p}{2}}}{\varepsilon^4}\right)$, Algorithm 3 the sum of the entries of $K$ up to a $1+\varepsilon$ factor with probability $\geq 0.99$.*

It can be verified that our exponent in $n$ and $\varepsilon$ are smaller than the bound of [BIMW21] (their exponent is $\approx 1/2 + p$). For example in the Gaussian kernel case, using $p = 0.173 + o(1)$ (Table 3), we obtain a running time of $n^{0.5865+o(1)}/\varepsilon^4$, whereas the algorithm of [BIMW21] takes time at least $n^{0.659}/\varepsilon^{4.159}$. Similar improvements are also achievable for other kernels (see Table 3). Indeed, Figure 1 shows that we improve over [BIMW21] across all ranges of $p$. We refer to Section 6.1 for a technical discussion.

Our algorithm of Theorem 2.4 (Algorithm 3) samples $\Theta(\sqrt{n}/\varepsilon^2)$ points, which we show is optimal in the following theorem. This improves upon a result in [BIMW21] which showed that $\Omega(\sqrt{n})$ points must be sampled.

**Theorem 2.5.** *Suppose $n \geq \Omega(1/\varepsilon^2)$. Consider an algorithm which specifies a subset $T \subseteq [n]$, receives the set of points $\{x_i, i \in T\}$ and returns a $1+\varepsilon/100$ approximation to $s(K)$, where $K$ is the underlying kernel matrix of the full set of $n$ points, with probability at least 99%. Then we must have $|T| \geq \Omega(\sqrt{n}/\varepsilon^2)$.*

Lastly in Section 6.2.1, we give a heuristic argument suggesting why our upper bound could be the limit of a certain natural class of KDE based algorithms for computing $s(K)$.

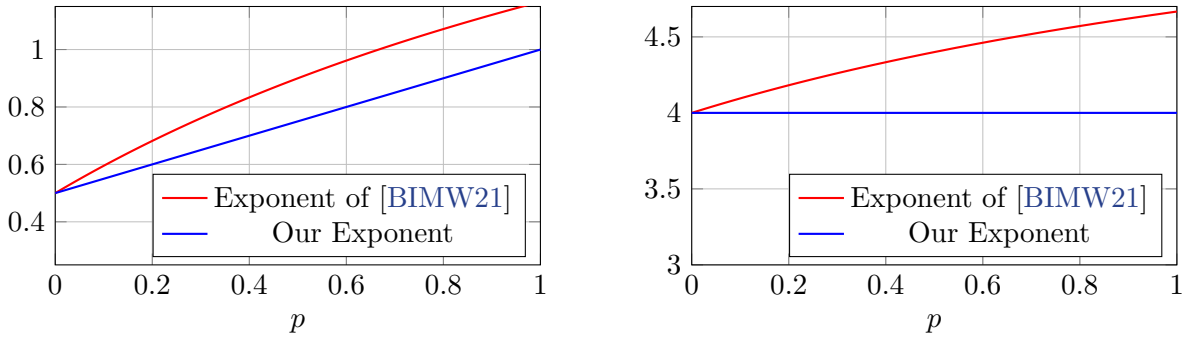


Figure 1: Exponent of $n$ (left plot) and exponent of $\varepsilon$ (right plot) for the problem of computing $s(K)$ up to a $1+\varepsilon$ factor. For the left plot, note that there is never a reason to spend more than $\omega(n)$ time since a simple Chernoff bound calculation shows that sampling $\tilde{O}(n/\varepsilon^2)$ uniformly random entries also suffices to $1+\varepsilon$ approximate $s(K)$. Thus, we may always take the minimum of the exponent of [BIMW21] and 1.

## 2.4 Improved Lower Bounds

We complement our upper bounds with lower bounds, demonstrating the limits of algorithms for computing on kernel matrices on problems similar to those we have introduced earlier. At a high level, all of our lower bounds encode the orthogonal vectors (OV) problem (Definition 3.1) in kernel computations, which implies they must take nearly quadratic time, assuming SETH.

First we discuss the case of computing approximate matrix vector products. Clearly, there remains a big gap in our understanding of the complexity of matrix-vector products for kernel

matrices: We have near linear upper bounds for the non-negative case with strong relative-error grantees (e.g. from [BIMW21] and our Theorem 2.1). On the other hand, virtually no non-trivial upper or lower bounds are known for the general case where the input vector can have both positive and negative entries, a question raised in prior works [BIMW21, IKSW25].

Our aim is to bridge this gap and provide a stronger evidence for the hardness of computing relative error estimates of $Kx$ for general $x$. This task seems out of reach of our techniques, but we show that a related intermediate problem requires $\Omega(n^2)$ time, assuming SETH. The intermediate problem is a slight generalization of computing $Kx$, where the kernel matrix $K$ can be "asymmetric" up to *one* vector. More formally, we consider the problem of computing matrix-vector products for kernel matrices $K$, where the rows correspond to a dataset $X = \{x_1, \cdots, x_n\}$ and the columns correspond to the same set $X$ *plus the origin* 0. Thus in the theorem below, $K \in \mathbb{R}^{n \times n+1}$ where $K_{ij} = k(x_i, x_j)$ for $1 \le i \le n, 1 \le j \le n$ and $K_{ij} = k(x_i, 0)$ for $j = n+1$ and $1 \le i \le n$.

**Theorem 2.6.** *Let $X = \{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ be input points with $d = \omega(\log n)$. Define $K \in \mathbb{R}^{n \times n+1}$ to be the Gaussian kernel matrix where the rows are indexed by $X$ and the columns are indexed by $X \cup \{0\}$. Any algorithm which takes as input $X$ and a vector $x \in \mathbb{R}^{n+1}$, and returns a $y \in \mathbb{R}^n$ such that $y = Kx + e$ with $\|e\|_2 \le n^{-0.002} \cdot \|Kx\|_2$ requires almost quadratic time, assuming SETH.*

Note that if the input vector is entry-wise non-negative, then it is easy to achieve the above guarantee: we simply call our non-negative matrix vector product algorithm of Theorem 2.1 on the first $n$ coordinates of $x$ and then compute the contribution of the last column of a $K \in \mathbb{R}^{n \times n+1}$ exactly in linear time. The theorem also generalizes to the case of computing a vector-matrix-vector product for a vector with possibly both positive and negative entries; see Theorem 7.2. Also see Section 7 for a detailed discussion of Theorem 2.6.

Our next lower bound studies a natural generalization of our vector matrix vector guarantee: can we compute $v^\top K w$ for a kernel matrix $K \in \mathbb{R}^{n \times n}$ where $v$ may not necessarily be equal to $w$? We show that the problem is hard, even in the easier case where $v$ and $w$ are assumed to be entry-wise non-negative.

**Theorem 2.7.** *For any $d = \omega(\log n)$, computing $v^\top K w$ for non-negative $v, w$ for the Gaussian kernel matrix $K \in \mathbb{R}^{n \times n}$ up to polynomial relative error requires almost quadratic time, assuming SETH.*

The underlying hard construction of the above theorem has broader applications in showing hardness for computations on *asymmetric* (but square) kernel matrices: kernel matrices where the rows and columns may not be indexed by the same point set. Such asymmetric kernel matrices are also common in practice, and we show that essentially none of our prior upper bounds are possible for the asymmetric case. In particular, we show that in the asymmetric case, computing any of the following quantities, even up to a polynomial approximation factor, solves the OV problem and thus requires nearly quadratic time:

- $\mathbf{1}^\top K \mathbf{1}$ (Corollary 8.1),
- top singular value (Corollary 8.2),
- computing a non-negative matrix vector product (Corollary 8.3).

## 3 Preliminaries

$X = \{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ are our input data points. To avoid confusion, we let $x(i)$ denote the $i$th coordinate of a vector $x$. All of our kernels satisfy $k(x, x) = 1$, $k(x, y) = k(y, x)$, and $k(x, y) \in [0, 1]$. $K \in \mathbb{R}^{n \times n}$ denotes the positive semi-definite kernel matrix where $K_{ij} = k(x_i, x_j)$. Throughout the paper, $\widetilde{O}(\cdot)$ hides $\text{poly}(\log(n/\varepsilon))$ terms.

For a kernel matrix $K$, we let $s(K)$ denote the sum of its entries. For a subset $S \subseteq [n]$, we let $K_S$ denote the principal submatrix where we keep the rows and columns whose indices are in $S$. For a (possibly rectangular) matrix $A \in \mathbb{R}^{m \times n}$ with $m \leq n$, we refer to the "diagonal" of $A$ as the diagonal of the $m \times m$ submatrix of $A$ formed by the first $m$ columns. We note the following prior works on kernels which satisfy Definition 1.1.

| Kernel | $k(x, y)$ | $p$ | Reference |
|---|---|---|---|
| Gaussian | $e^{-\|x-y\|_2^2}$ | $0.173 + o(1)$ | [CKNS20] |
| Exponential | $e^{-\|x-y\|_2}$ | $0.1 + o(1)$ | [CKNS20] |
| Laplacian | $e^{-\|x-y\|_1}$ | 0.5 | [BIW19] |
| Rational Quadratic | $\frac{1}{(1+\|x-y\|_2^2)^\beta}$ | 0 | [BCIS18] |

Table 3: Known results for kernel density estimation queries. The stated result in the last row assumes $\beta = O(1)$ for the rational quadratic kernel. We note that there is an alternate data structure for the rational quadratic kernel given in [CKW24] which achieves query time $O(\frac{\text{poly}(\log(n/\varepsilon) \cdot d)}{\varepsilon})$, i.e. it has a $\varepsilon^{-1}$ dependency at the cost of a larger polynomial dependence on $d$.

**Remark 3.1** (Remark on KDE guarantees)**.** We note that usually, the KDE guarantees are written in a slightly alternate form [CKNS20, BIMW21]: Given a lower bound $\mu$, the datastructure returns a $1 + \varepsilon$ approximation to the sum $\frac{1}{n}\sum_{i=1}^n k(y, x_i)$ assuming it is at least $\mu$. The datastructure also reports the case where $\mathcal{D}_X(y) < \mu$. By always adding $+\mu$ to the output in all cases, we can also obtain the weaker guarantee stated in (1), which will be convenient in our analysis.

**Definition 3.1** (Orthogonal Vectors (OV))**.** *Given two sets of vectors $A = \{a_1, \ldots, a_n\} \subseteq \{0, 1\}^d$ and $B = \{b_1, \ldots, b_n\} \subseteq \{0, 1\}^d$ of $n$ binary vectors, decide if there exists a pair $a \in A, b \in B$ such that $\langle a, b\rangle = 0$.*

It is known that OV requires almost quadratic time (i.e., $n^{2-o(1)}$) for any $d = \omega(\log n)$, assuming the Strong Exponential Time Hypothesis (SETH) [Wil05, AWW14, ABW15]. OV is the only lower bound assumption we use in the paper.

## 4 Improved approximate Kernel Matrix Vector Multiplication

The main algorithm in this paper is Algorithm 1, which we show admits the following guarantee.

**Theorem 2.1.** *Let $k$ be a kernel admitting a KDE datastructure of Definition 1.1. Let $K \in \mathbb{R}^{n \times n}$ be the associated kernel matrix for $n$ points in $d$ dimensions. There is an $\varepsilon$-non-negative approximate*

*matrix-vector product algorithm (Algorithm 1) satisfying Definition 2.1 for $K$ running in time* $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$.

The subsequent sections Section 4.2 and Section 5 analyze applications of this algorithm.

**Intuition Behind the Algorithm and Proof.** Before presenting the formal details, we give a high-level, intuitive overview of our algorithm, starting with a discussion on the prior algorithm of [BIMW21]. Given an input vector $y$, recall the goal is to satisfy Definition 2.1. Since we know that $\|Ky\|_2^2 \geq 1$ using the non-negativity of $y$, it suffices to obtain a $1+\varepsilon$ multiplicative approximation and $\varepsilon/\sqrt{n}$ additive error per coordinate of $Ky$ (so that the total additive error is at most $\varepsilon \leq \varepsilon\|Ky\|_2$). Thus, let's focus on the first coordinate. For the first coordinate, we want to estimate the sum

$$(Ky)_1 = \sum_{j=1}^{n} y_j k(x_1, x_j).$$

The sum above looks almost like a standard KDE query of Definition 1.1. However, it is weighted by the coefficients of the vector $y$. While some KDE datastructures can be made to handle positive weights in a while-box manner [CS17], this may not be true for all KDE datastructures, so [BIMW21] give a black-box reduction for computing the above sum given any KDE datastructure satisfying Definition 1.1.

Their analysis proceeds as follows. First, note that can ignore all coordinates $y_j \leq O(\varepsilon/n^{1.5})$ since these values can only contribute value $\leq O(\varepsilon/\sqrt{n})$ error to the above sum. [BIMW21] then proceed by bucketing the rest of the coordinates, which are all in the range $[\varepsilon/n^{1.5}, 1]$, into geometrically increasing values by a factor of $1+\varepsilon$. Ignoring logarithmic and $d$ factors for the sake of this discussion, this results in $O(1/\varepsilon)$ total buckets.

Inside every bucket, the coordinates are of all of approximately the same value (up to a multiplicative $1+\varepsilon$ factor), so one can effectively 'factor out' the coordinates $y_j$ in the desired sum for the first coordinate of $Ky$, and use a standard KDE query, calling on Definition 1.1. The runtime is dominated by the error $\mu$ that is set. [BIMW21] show that setting a *universal* value of $\mu = O(\varepsilon/n)$ suffices for every bucket, leading to a running time of $\widetilde{O}(n^p/\varepsilon^{2+p})$ per bucket and a total running time of $\widetilde{O}(n^p/\varepsilon^{3+p})$ per coordinate, and hence an overall $\varepsilon$-precision non-negative matrix vector product in $\widetilde{O}(n^{1+p}/\varepsilon^{3+p})$.

This discussion makes it clear that the extra undesired $1/\varepsilon$ factor is from the bucketing procedure. Rather than using one KDE data-structure per bucket (with a fixed $\mu$), our main idea is thus to use an *adaptive* choice of $\mu$, depending on the approximate value of the bucket. We have to handle two regimes: the first is when the bucket has mass more than $1/\sqrt{n}$, i.e., consider coordinates with value $\Theta(2^t/\sqrt{n})$ for some parameter $t \geq 0$. Such buckets can only have at most $O(n/2^{2t})$ many coordinates, since the total $\ell_2$ squared sum of the input $y$ is bounded by 1. For this bucket, if we set some additive error $\mu_t$, then the total additive error we get for the contribution of coordinates in this bucket is $\Theta(2^t/\sqrt{n})$ coming from factoring out the coordinates $y_j$ in the KDE sum, times $O(n/2^{2t})$, coming from the size of the bucket since the KDE query gives a scaled sum, times $\mu_t$. Thus, the additive error per bucket can be approximately bounded by $\approx \frac{2^t}{\sqrt{n}} \times \frac{n}{2^{2t}} \times \mu_t$. The right choice is to set $\mu \approx \frac{2^t\varepsilon^2}{n}$, bounding the overall additive error per bucket by $\widetilde{O}(\varepsilon^2/\sqrt{n})$, which allows us to control the total error on across $1/\varepsilon$ buckets.

For buckets that are smaller than $1/\sqrt{n}$, say $\Theta(\frac{1}{2^t\sqrt{n}})$ for $t \geq 1$, we use a different strategy. For these buckets, we cannot control their size since $\Theta(n)$ coordinates could be of approximately this

value. However, then the $\approx \frac{1}{2^t\sqrt{n}}$ value that we factor out in the KDE sum helps us, and a similar calculation as above shows that we can again pick $\mu \approx \frac{2^t\varepsilon^2}{n}$. Ignoring the $1/\varepsilon^2$ overhead of KDE queries, the overall query time then is approximately

$$\sum_{t\geq 0} \frac{1}{\mu_t^p} = \sum_{t\geq 0} \left(\frac{n}{2^t\varepsilon^2}\right)^p = O\left(\frac{n^p}{\varepsilon^{2p}}\right)$$

across *all* buckets for a fixed coordinate, meaning we don't have to pay the $1/\varepsilon$ overhead of the total number of buckets. One important point to note is that we have to be careful and bound the overall running time of creating all the different KDE datastructures as well. However, the KDE datastructures can be shared across all $n$ coordinates of $Ky$ that we compute, since the bucketing depends only on the values of $y$ and not the entries of $K$. Thus, the overall datastructure construction time can also be bounded using a similar sum as the query time. This ends the intuitive explanation and the formal algorithm and proof follow.

**Algorithm 1** Non-negative Matrix Vector Product

**procedure** NON-NEG-MVP($y \in \mathbb{R}^n, \|y\|_2 = 1, \varepsilon \in (0,1)$)
  $b \leftarrow C\log(n/\varepsilon)/\varepsilon$ for a sufficiently large constant $C > 1$
  **for** $i = 1$ to $b$ **do**
    $\mathcal{Y}_i \subseteq [n] \leftarrow$ subset of coordinates of $y$ consisting of values in the range $[(1-\frac{\varepsilon}{2})^i, (1-\frac{\varepsilon}{2})^{i-1}]$
    $\mathbb{R}^n \ni y^i \leftarrow$ coordinates of $y$ in $\mathcal{Y}_i$, rest 0 ▷ *We ignore entries of* $y < \frac{\varepsilon}{(b+1)n^{3/2}}$
    $\tilde{y}^i \leftarrow$ every non-zero coordinate of $y^i$ rounded to $(1-\frac{\varepsilon}{2})^{i-1}$
    **if** $(1-\frac{\varepsilon}{2})^{i-1} \geq \frac{1}{\sqrt{n}}$ **then**
      Let $t_i \geq 0$ be such that $(1-\frac{\varepsilon}{2})^{i-1} \in [\frac{2^{t_i}}{\sqrt{n}}, \frac{2^{t_i+1}}{\sqrt{n}})$
    **else**
      Let $t_i \geq 0$ be such that $(1-\frac{\varepsilon}{2})^{i-1} \in (\frac{1}{2^{t_i+1}\sqrt{n}}, \frac{1}{2^{t_i}\sqrt{n}}]$
    **end if**
    $\mu_i \leftarrow \frac{2^{t_i}\varepsilon'}{Cn}$ for a large constant $C > 1$ where $\varepsilon' = \frac{\varepsilon}{b}$
    Initialize data structure $\mathcal{D}_i$ of Definition 1.1 on points $\{x_j \in X \mid j \in \mathcal{Y}_i\}$, using additive error $\mu_i$ and relative error $1+\varepsilon$
    $z^i \leftarrow 0$ ▷ $z^i$ *is our approximation to* $K\tilde{y}^i$
    **for** $j = 1$ to $n$ **do**
      $z^i(j) \leftarrow (1-\frac{\varepsilon}{2})^{i-1}|\mathcal{Y}_i| \cdot \mathcal{D}_i(x_j)$ ▷ $x_1, \cdots, x_n$ *are the original data points*
    **end for**
  **end for**
  **Return** $z := \sum_i z^i$ as the approximation to $Ky$
**end procedure**

In our analysis below, we assume that every query to the KDE data structure satisfies the guarantees of (1). This happens with high probability. The first two lemmas show that removing all sufficiently small coordinates of the input $y$ and then rounding its values has a tolerable effect on the error of the final output.

**Lemma 4.1.** *Let* $y' = \sum_i y^i$. *We have* $\|Ky - Ky'\|_2 \leq \varepsilon\|Ky\|_2$.

*Proof.* Note that

$$\|Ky\|_2^2 \geq \|y\|_2^2 = 1 \tag{4}$$

since the diagonal of $K$ is all ones and the entries of $K - I$ and $y$ are all non-negative. Thus $y - y'$ just contains all the entries of $y$ that are smaller than $\frac{\varepsilon}{(b+1)n^{3/2}}$. Each coordinate of $K(y-y')$ is thus bounded by $\varepsilon/\sqrt{n}$ (since every entry of $K$ is at most 1) and so $\|K(y - y')\|_2 \leq \varepsilon \leq \varepsilon\|Ky\|_2$. □

**Lemma 4.2.** *Let $\tilde{y} = \sum_i \tilde{y}^i$. We have $\|K\tilde{y} - Ky'\|_2 \leq O(\varepsilon)\|Ky\|_2$.*

*Proof.* Note that both $\tilde{y}$ and $y'$ are non-negative and we have rounded every entry of $y'$ to it's closest multiple of $1 - \varepsilon/2$. For every $j \in [n]$, this means every $j$th entry of $\tilde{y} - y'$ is bounded between 0 and $O(\varepsilon)$ times the $j$th entry of $y'$. Again using the fact that the entries of $K$ are non-negative and so are the entries of $\tilde{y} - y'$, we have $0 \leq [K(\tilde{y} - y')](j) \leq O(\varepsilon)[Ky'](j)$, so

$$\|K\tilde{y} - Ky'\|_2 \leq O(\varepsilon)\|Ky'\|_2 = O(\varepsilon)\|Ky' - Ky + Ky\|_2 \leq O(\varepsilon)\|Ky\|_2,$$

as desired. □

The following is our main lemma proving our approximation bound. It executes the high-level strategy discussed in the intuition above.

**Lemma 4.3.** *We have $\|Ky - z\|_2 \leq O(\varepsilon) \cdot \|Ky\|_2$.*

*Proof.* Consider a fixed entry $j \in [n]$. We have

$$(K\tilde{y}^i)(j) = \left(1 - \frac{\varepsilon}{2}\right)^{i-1} \sum_{\ell \in \mathcal{Y}_i} K(x_j, x_\ell) := \Delta$$

In the $j$ case of loop 15 in Algorithm 1, the query $\mathcal{D}_i(x_j)$ satisfies (via Equation (1))

$$\left| \mathcal{D}_i(x_j) - \frac{1}{|\mathcal{Y}_i|} \sum_{\ell \in \mathcal{Y}_i} K(x_j, x_\ell) \right| \leq \frac{\varepsilon}{|\mathcal{Y}_i|} \sum_{\ell \in \mathcal{Y}_i} K(x_j, x_\ell) + \mu_i.$$

Multiplying through by $(1 - \frac{\varepsilon}{2})^{i-1}|\mathcal{Y}_i|$ gives us

$$\left| \left(1 - \frac{\varepsilon}{2}\right)^{i-1} |\mathcal{Y}_i| \cdot \mathcal{D}_i(x_j) - \Delta \right| \leq \varepsilon\Delta + \left(1 - \frac{\varepsilon}{2}\right)^{i-1} |\mathcal{Y}_i| \cdot \mu_i.$$

We need to bound the additive error in the RHS above. Note that $\mathcal{Y}_i$ corresponds to a set of coordinates in $y$ with values at least $\left(1 - \frac{\varepsilon}{2}\right)^i$. We now two cases. First suppose that $i$ triggered the if condition on line 7. Then this means $\mathcal{Y}_i$ corresponds to the coordinates in $y$ with entries at least $\Omega(2^{t_i}/\sqrt{n})$, so $|\mathcal{Y}_i| \leq n/2^{2t_i}$ (since the total $\ell_2$ norm of $y$ is 1). This gives us

$$\left(1 - \frac{\varepsilon}{2}\right)^{i-1} |\mathcal{Y}_i| \cdot \mu_i \leq O\left(\frac{2^{t_i}}{\sqrt{n}} \cdot \frac{n}{2^{2t_i}} \cdot \frac{2^{t_i}\varepsilon'}{n}\right) = O\left(\frac{\varepsilon'}{\sqrt{n}}\right).$$

Otherwise, we are in the else case of line 9 of Algorithm 1. Here we can only guarantee $|\mathcal{Y}_i| \leq n$, but we know $(1 - \varepsilon/2)^{i-1}$ is sufficiently small. We have

$$\left(1 - \frac{\varepsilon}{2}\right)^{i-1} |\mathcal{Y}_i| \cdot \mu_i \leq O\left(\frac{1}{2^{t_i}\sqrt{n}} \cdot n \cdot \frac{2^{t_i}\varepsilon'}{n}\right) = O\left(\frac{\varepsilon'}{\sqrt{n}}\right).$$

Thus for every $j \in [n]$, the $j$th entry of $K\tilde{y}^i - z^i$ is bounded in absolute value by $O(\varepsilon) \cdot [K\tilde{y}^i](j) + O\left(\frac{\varepsilon'}{\sqrt{n}}\right)$. Thus, by the triangle inequality, the $j$th entry of $\sum_i (K\tilde{y}^i - z^i)$ is bounded in absolute value by $O(\varepsilon) \cdot [K\sum_i \tilde{y}^i](j) + O\left(\frac{\varepsilon' b}{\sqrt{n}}\right) = O(\varepsilon) \cdot [K\tilde{y}](j) + O\left(\frac{\varepsilon' b}{\sqrt{n}}\right)$, meaning

$$\|K\tilde{y} - z\|_2 \leq O(\varepsilon)\|K\tilde{y}\|_2 + O(\varepsilon) \leq O(\varepsilon)\|Ky\|_2,$$

as desired. Finally, we note that we can always guarantee $e$ is entry-wise non-negative as well by scaling up all of our intermediate estimates in line 16 of Algorithm 1 by $1 + \varepsilon$. □

The next lemma bounds the overall running time, including the query and datastructure construction times.

**Lemma 4.4.** *Assuming a fast KDE data structure of Definition 1.1, the total runtime of Algorithm 1 is bounded by* $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$.

*Proof.* We bound the data structure construction and query times separately. First note that for the case of $p > 0$, the sum of $1/\mu^p$ overall $\mu$'s possibly considered in line 12 of Algorithm 1 can be bounded up to constant factors by

$$T = O\left(\sum_{t \geq 0} \left(\frac{n}{2^t \varepsilon'}\right)^p\right) = O\left(\frac{n^p}{\varepsilon'^p} \sum_{t \geq 0} \frac{1}{2^{kp}}\right) = O\left(\frac{n^p}{\varepsilon'^p}\right).$$

We quickly remark that in the case of $p = 0$, the above sum over $t$ can be truncated to only have $\text{poly}(\log n)$ terms, giving $T_1 = \widetilde{O}(1)$ instead.

Now since $|\mathcal{Y}_i| \leq n$, the total data structure construction times across all instances of $\mathcal{D}_i$ is bounded by

$$\frac{1}{\varepsilon^2} \cdot O\left(\sum_{i=1}^{b} \frac{d|\mathcal{Y}_i|}{\mu_i^p}\right) \leq O\left(\frac{ndT}{\varepsilon^2}\right) = O\left(\frac{n^{1+p}d}{\varepsilon^2 \cdot \varepsilon'^p}\right).$$

Now we bound the total query time. There is always an overhead of $n$ coming from the loop in line 15 of Algorithm 1 since we query each $\mathcal{D}_i$ $n$ times. The cost of a single query summed across all $\mathcal{D}_i$ is bounded by

$$\widetilde{O}\left(\sum_i \frac{d}{\varepsilon^2 \mu_i^2}\right) = \widetilde{O}\left(\frac{dn^p}{\varepsilon^{2+2p}}\right),$$

giving a total running time of $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$, as desired. □

Putting everything together proves our main theorem.

*Proof of Theorem 2.1.* Lemma 4.3 shows the desired $\varepsilon \cdot \|Ky\|_2$ error bound (after scaling $\varepsilon$ by a constant) and Lemma 4.4 proves the running time, which proves Theorem 2.1. □

**Remark 4.1.** Note that the output of Algorithm 1 satisfies a slightly stronger guarantee than stated in Theorem 2.1: Every entry of $z$, which is the approximation to $Ky$ for an input non-negative unit vector $y$, satisfies $(Ky)(i) \leq z(i) \leq (1 + \varepsilon)(Ky)(i) + O\left(\frac{\varepsilon}{\sqrt{n}}\right)$.

### 4.1 Limits of KDE-Based Algorithms for Non-Negative Matrix Vector Product?

We give some evidence that a running time of the form $n^{1+p}$ maybe necessary, if want the 'per coordinate' guarantee of our non-negative matrix vector product gives, namely that every coordinate of $Ky$, for a unit vector $y$, is approximated up to $1+\varepsilon$ multiplicative error and $\varepsilon/\sqrt{n}$ additive error (see Remark 4.1).

**Lemma 4.5.** *Suppose there is an algorithm which computes a non-negative matrix vector product on a dataset of size $n$ with the same guarantees as Remark 4.1 ($1+\varepsilon$ multiplicative error and $\varepsilon/\sqrt{n}$ additive error per coordinate) in $T(n,\varepsilon)$ time. Then there exists an algorithm which given a set $X, |X| = n$ and a set of $n$ queries $Z = \{z_1, \cdots, z_n\}$, answers all $n$ KDE queries $\frac{1}{n}\sum_{x\in X} k(z_i, x)$ with $1+\varepsilon$ multiplicative and $\varepsilon/n$ additive error in $T(O(n),\varepsilon)$ time. That is, the amortized time per KDE query to get a $1+\varepsilon$ multiplicative and $\varepsilon/n$ additive error is $T(O(n),\varepsilon)/n$ time.*

Before the proof, let's see how the lemma relates to our bound of $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$. Ignoring logarithmic factors and the $d$ term, our bound can be decomposed into

$$n \times \frac{n^p}{\varepsilon^{2+2p}}.$$

Thus the lemma implies that, based on the second term, we can get an amortized KDE query time of $n^p/\varepsilon^{2+2p}$ for multiplicative error $1+\varepsilon$ and relative error $\varepsilon/n$. Ignoring an extra $\varepsilon^p$ term, this is essentially the best KDE query time one can obtain for this desired level of precision (almost by definition from Definition 1.1).

Thus one cannot improve upon our algorithm for computing a non-negative matrix vector query, with the same coordinate wise approximation guarantees, unless there is a different family of KDE algorithms different than the form of Definition 1.1 (which we are not aware of), or amortization helps (which we are also not aware of). There is one more caveat we must mention: it may also possible to obtain an $\varepsilon$-non-negative matrix vector product without having the stronger coordinate wise guarantee. We leave these as interesting questions for future work.

*Proof of Lemma 4.5.* Consider the kernel matrix on the dataset $Z \cup X$ and the query vector $y = [0, \ldots, 0, 1/\sqrt{n}, \ldots, 1/\sqrt{n}]$ where there are $n$ copies of $1/\sqrt{n}$ and $n$ copies of 0. Then if we compute a non-negative matrix vector product using this query $v$, then the $i$th coordinate of the output is exactly a $1+\varepsilon$ multiplicative and $\varepsilon/n$ additive approximations to

$$\frac{1}{\sqrt{n}}\sum_{x\in X} k(y_i, x).$$

We need to divide by $1/\sqrt{n}$ to get the correct scaling for a KDE query, so we are left with a $1+\varepsilon$ multiplicative and $\varepsilon/n$ additive error approximation to each of the $n$ KDE queries of interest, as desired. □

### 4.2 Approximate Matrix Multiplication

Another application of our non-negative matrix-vector product result of Theorem 2.6 is a faster algorithm to (approximately) multiply a kernel matrix with another entry-wise non-negative matrix, for example another kernel matrix.

**Theorem 4.6.** *Let $k$ be a kernel admitting a KDE datastructure of Definition 1.1. Let $K \in \mathbb{R}^{n \times n}$ be the associated kernel matrix for $n$ points in $d$ dimensions. Let $A$ be any entry wise non-negative matrix (e.g. another kernel matrix). In time $\widetilde{O}\left(\frac{dn^{2+p}}{\varepsilon^{2+2p}}\right)$, we can compute a matrix $B$ such that*

$$\|KA - B\|_F \leq \varepsilon \|KA\|_F \leq \varepsilon \cdot \min(\|K\|_2 \cdot \|A\|_F, \|K\|_F \cdot \|A\|_2)$$

*with high probability.*

Note that standard techniques for approximate matrix multiplication multiplication, based on sketching or row sampling [M+11, W+14, CNW16], roughly say that by sampling $\widetilde{O}(1/\varepsilon^2)$ columns of $K$ and $\widetilde{O}(1/\varepsilon^2)$ rows of $A$, in time $\widetilde{O}(n^2/\varepsilon^2)$ we can compute a $B$ satisfying

$$\|KA - B\|_F \leq \varepsilon \|K\|_F \cdot \|A\|_F.$$

Our result is able to replace a frobenius norm factor with a spectral norm factor, e.g. $\|K\|_F$ can be converted to $\|K\|_2$, which can be up to $\Omega(\sqrt{n})$ times smaller.

*Proof.* We simply view the columns of the matrix $A$ as separate vectors and apply our non-negative matrix vector product guarantee. We have that the $i$th column of our output $B$ is the same as the $i$th column of $KA$, denoted as $(KA)_i$, up to error which has norm at most $\varepsilon \|(KA)_i\|_2$. Then

$$\|B\|_F^2 \leq \varepsilon^2 \|(KA)_i\|_2^2 = \varepsilon^2 \|KA\|_F^2,$$

as desired. The last step follows from a well known fact, which we quickly sketch here. Letting $A_i$ denote the $i$th column of $A$,

$$\|K(A_i)\|_2 \leq \|K\|_2 \|A_i\|_2 \implies \sum_i \|K(A_i)\|_2^2 \leq \|K\|_2^2 \sum_i \|A_i\|_2^2 = \|K\|_2^2 \|A\|_F^2,$$

as desired. The running time follows from $n$ invocation of Theorem 2.1. □

# 5 Faster Sub-quadratic Time Top Eigenvalue

We now analyze the noisy power method for approximating the top eigenvalue of a given kernel matrix. In this method, exact matrix-vector products in the usual power method are replaced by a approximate matrix-vector products with error satisfying Definition 2.1. This method, stated precisely in Algorithm 2, produces a sequence of unit vectors $z_0, z_1, z_2, \ldots$ satisfying the following definition.

**Definition 5.1** ($\delta$-approximate power iteration)**.** *Let $z_t, \widetilde{z}_t$ be the sequence of normalized and unnormalized unit vectors produced by $\delta$-approximate power iteration starting with $\langle v_1, z_0 \rangle \geq 1/\sqrt{n}$. That is,*

$$\widetilde{z}_{t+1} = Kz_t + \|Kz_t\| \, \delta u_t \quad \& \quad z_{t+1} = \frac{\widetilde{z}_{t+1}}{\|\widetilde{z}_{t+1}\|} \tag{5}$$

*for some unit vectors $u_t$ satisfying $\langle v_1, u_t \rangle \geq 0$.*

In this definition, $\delta$ denotes the error to which each matrix-vector product is performed. When $\delta = 0$, this is the exact power method and $z_t$ approaches span $\{v_j : \lambda_j = \lambda_1\}$. For $\delta > 0$, the best available analysis in the literature is due Backurs et al [BIMW21]. They show that if $\delta = O(\varepsilon^2)$, then the power method produces a unit vector $u$ satisfying the relative error guarantee (3),

$$u^\top K u \geq (1-\varepsilon)\lambda_1, \tag{3}$$

in $O(\log(n/\varepsilon)/\varepsilon)$ iterations. The smaller $\delta$ is, the more expensive the approximate matrix-vector products become. It's therefore desirable to improve upon the condition $\delta = O(\varepsilon^2)$. The classical analysis of the power method as well as the analysis of [BIMW21] expresses each iterate $z_t$ in the eigenbasis of $K$ and considers the mass this vector places on the top eigenvector and the eigenvalues above versus below the threshold $(1-\varepsilon)\lambda_1$. The analysis then shows that the mass placed below $(1-\varepsilon)\lambda_1$ is driven to 0. By adjusting some of the expressions in [BIMW21], one can improve $\delta = O(\varepsilon^2)$ to $\delta = O(\varepsilon^{1.5})$, but the argument fundamentally breaks down for $\delta = \Omega(\varepsilon^{1.5})$ as demonstrated by Example 1.

Rather than track the mass that each iterate $z_t$ places on eigenvalues above and below $(1-\varepsilon)\lambda_1$, our analysis tracks the mass placed on just the top eigenvector $v_1$, that is, $\langle v_1, z_t\rangle$, and the norm of the vector $\|Kz_t\|$. This can be thought of as a kind of weighted sum of the components $\langle v_j, z_t\rangle$ in the eigenbasis where the components corresponding to small eigenvalues are weighted less. Notably, we do not make a hard cutoff between eigenvalues that are above or below $(1-\varepsilon)\lambda_1$. We directly show that $\|Kz_t\|$ is driven to $\lambda_1$ *without* appealing to an argument that shows the components in the direction of small eigenvectors become small.

**Example 1** (Limitation of previous argument for $\delta = \Omega(\varepsilon^{1.5})$)**.** Suppose one runs $\delta$-approximate power iteration for $\delta \geq (2\varepsilon)^{1.5}/(1-5\varepsilon^2)$ on a kernel matrix $K \in \mathbb{R}^{3\times 3}$ with the goal of achieving the relative error guarantee (3) in any number iterations. Suppose $K = VDV^\top$ has the eigenvalues

$$\lambda_1 > 0, \quad \lambda_2 = (1-2\varepsilon)\lambda_1, \quad \lambda_3 = 0.$$

Let $x_t = V^\top z_t$, $\widetilde{x}_t = V^\top \widetilde{z}_t$. This example shows that one *cannot* conclude convergence by only considering the quantities $x_t(1)$ and $\|x_t(2:3)\| = \sqrt{x_t(2)^2 + x_t(3)^2}$. In particular, the set

$$S = \left\{ x \in \mathbb{R}^3 : x_t(1) = \sqrt{2\varepsilon},\ \|x_t(2:3)\| = \sqrt{1-2\varepsilon} \right\}$$

both contains vectors that do not witness $\lambda_1$, and vectors that are fixed by the noisy power method. Consider

$$x_t = \begin{bmatrix} \sqrt{1-2\varepsilon} & \sqrt{2\varepsilon} & 0 \end{bmatrix}^\top \in S.$$

Then it is possible for the next iterate $\widetilde{x}_{t+1}$ to be

$$\begin{aligned} \widetilde{x}_{t+1} &= \begin{bmatrix} \lambda_1\sqrt{1-2\varepsilon} & \lambda_2\sqrt{2\varepsilon} + \delta\sqrt{\lambda_1^2(1-2\varepsilon) + \lambda_2^2\cdot 2\varepsilon} & 0 \end{bmatrix}^\top \\ &= \lambda_1 \begin{bmatrix} \sqrt{1-2\varepsilon} & \sqrt{2\varepsilon} - (2\varepsilon)^{1.5} + \delta\sqrt{1-8\varepsilon^2+8\varepsilon^3} & 0 \end{bmatrix}^\top. \end{aligned} \tag{6}$$

When $\delta \geq \frac{(2\varepsilon)^{1.5}}{\sqrt{1-8\varepsilon^2+8\varepsilon^3}}$, then it's possible for $x_{t+1} = x_t$, meaning that the power method makes no additional progress. On the other hand, consider

$$x_t = \begin{bmatrix} \sqrt{1-2\varepsilon} & 0 & \sqrt{2\varepsilon} \end{bmatrix}^\top \in S.$$

The quadratic form

$$z_t^\top K z_t = x_t(1)^2 \lambda_1 = (1-2\varepsilon)\lambda_1,$$

does not give an $\varepsilon$ relative error approximation of $\lambda_1$.

**Algorithm 2** Top Eigenpair Computation

**Input:** A kernel matrix $K$, a matrix-vector product method `Non-neg-MVP` satisfying Definition 2.1, an error parameter $\varepsilon \in (0,1)$.
**Output:** A scalar $\lambda$ and non-negative unit vector $u$.
**procedure** TOP-EIGENPAIR
  $T \leftarrow \lceil 10\log(n)/\varepsilon \rceil$ ▷ Number of iterations of power method
  Initialize $u = z_0 \leftarrow \frac{1}{\sqrt{n}}\mathbf{1} \in \mathbb{R}^n$
  $\lambda \leftarrow z_0^\top K z_0$
  **for** $t = 0$ to $T$ **do**
    $\widetilde{z}_{t+1} \leftarrow$ `Non-neg-MVP`$(K, z_t, \varepsilon/8)$ ▷ Algorithm 1 with precision $\varepsilon/8$
    **if** $\langle z_t, \widetilde{z}_{t+1}\rangle > \lambda$ **then**
      $u \leftarrow z_t$
      $\lambda \leftarrow \langle z_t, \widetilde{z}_{t+1}\rangle$
    **end if**
    $z_{t+1} \leftarrow \widetilde{z}_{t+1} / \left\|\widetilde{z}_{t+1}\right\|$
  **end for**
  **Return** $\lambda, u$
**end procedure**

The key lemma is that one of the $z_t$ produced by Algorithm 2 will satisfy a relative error guarantee.

**Lemma 5.1.** *Let $z_t$ be defined by Definition 5.1 for $\delta = \varepsilon/8 \le 0.01$. Let $T = 10\log(n)/\varepsilon$. Then there exists $t \in [0,T]$ for which $z_t^\top K z_t \ge (1-\varepsilon/2)\lambda_1(K)$.*

*Proof.* By Definition 5.1, we may express the component in the direction of the top eigenvector as

$$\langle v_1, z_{t+1}\rangle = \frac{\langle v_1, \widetilde{z}_{t+1}\rangle}{\left\|\widetilde{z}_{t+1}\right\|} = \frac{\langle v_1, Kz_t + \|Kz_t\|\,\delta u(t)\rangle}{\left\|\widetilde{z}_{t+1}\right\|}.$$

Now use that $v_1$ is a left-eigenvector of $K$ and that $\langle v_1, u(t)\rangle \ge 0$ to obtain

$$\langle v_1, z_{t+1}\rangle = \frac{\lambda_1 \langle v_1, z_t\rangle + \|Kz_t\|\langle v_1, u(t)\rangle}{\left\|\widetilde{z}_{t+1}\right\|\delta} \ge \frac{\lambda_1 \langle v_1, z_t\rangle}{\left\|\widetilde{z}_{t+1}\right\|}. \tag{7}$$

The denominator can be upper bounded by the triangle inequality and Hölder's inequality,

$$\left\|\widetilde{z}_{t+1}\right\| \le (1+\delta)\|Kz_t\| \le (1+\delta)\sqrt{z_t^\top K z_t \cdot \lambda_1}. \tag{8}$$

Suppose that that

$$z_t^\top K z_t \le (1-\varepsilon/2)\lambda_1$$

for all $t \le T$. Combining this with (7) and (8), we arrive at

$$\langle v_1, z_{t+1}\rangle \ge \frac{\langle v_1, z_t\rangle}{(1+\delta)\sqrt{1-\varepsilon/2}}.$$

Since $\langle v_1, z_0\rangle \ge n^{-1/2}$, we have

$$\langle v_1, z_T\rangle \ge \frac{\langle v_1, z_0\rangle}{\left((1+\delta)\sqrt{1-\varepsilon/2}\right)^T} \ge \frac{n^{-1/2}}{\left((1+\delta)\sqrt{1-\varepsilon/2}\right)^T}.$$

Since $\delta = \varepsilon/8 \le 0.01$, when $T = 10\log(n)/\varepsilon$, we have $\langle v_1, z_T\rangle > 1$ which is a contradiction. □

The previous lemma shows that there exist a "good" vector $z_t$. The remainder of the proof of our main theorem is that the algorithm will pick up on this property and return it or a similarly good vector.

**Theorem 2.2.** *Let $\varepsilon \in (0,1)$ be sufficiently small. There exists an algorithm (Algorithm 2) which returns a scalar $\lambda$ and unit vector $u$ satisfying*

$$\lambda_1(K) \ge u^\top K u \ge (1-(5/8)\varepsilon)\lambda_1(K).$$
$$(1+\varepsilon/8)\lambda_1(K) \ge \lambda \ge (1-\varepsilon/2)\lambda_1(K).$$

*The overall runtime of the algorithm is $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{3+2p}}\right)$.*

*Proof.* Note that by the guarantee of Definition 2.1, the sequence $z_t$ produced by Algorithm 2 satisfies Definition 5.1. Algorithm 2 returns $\lambda$ and $z_r$ where

$$r = \arg\max_{t\in[0,T]} \langle z_t, \widetilde{z}_{t+1}\rangle, \quad \lambda = \langle z_r, \widetilde{z}_{r+1}\rangle.$$

Note that by Definition 5.1,

$$z_t^\top K z_t \le \langle z_t, \widetilde{z}_{t+1}\rangle = z_t^\top K z_t + \delta \|Kz_t\| \langle z_t, u_t\rangle \le z_t^\top K z_t + \delta\lambda_1.$$

Let $\alpha = \max_{t\in[0,T]} z_t^\top K z_t$ so that

$$\alpha \le \max_{t\in[0,T]} \langle z_t, \widetilde{z}_{t+1}\rangle = \lambda \le \alpha + \delta\lambda_1$$

and

$$\alpha - \delta\lambda_1 \le z_r^\top K z_r.$$

But Lemma 5.1 ensures that

$$(1-\varepsilon/2)\lambda_1 \le \alpha \le \lambda_1$$

which gives the desired approximation guarantee for $\delta = \varepsilon/8$. The running time of our algorithm can be decomposed into the following:

$$\underbrace{O\left(\frac{\log(n/\varepsilon)}{\varepsilon}\right)}_{\#\text{ of iterations}} \times \underbrace{T(n,\varepsilon/8)}_{\text{Running time for a single } \varepsilon/8\text{-precision MVP}}.$$

We have

$$T(n,\varepsilon/8) = \widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+p}}\right),$$

from Theorem 2.1, resulting in the overall running time of $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{3+p}}\right)$, as desired. □

### 5.1 Lower bounds

In this section, we argue that Theorem 2.2 is optimal in a couple different senses. The first is that the precision requirement $\delta$ for each matrix-vector product cannot be much larger than is set by Algorithm 2. The second is that *any* algorithm produces an output based on the $z_t$ requires $\Omega(\log(n)/\varepsilon)$ iterations. Notably, this is *not* the case when $\delta = 0$. When $\delta = 0$, the span of the $z_t$ forms a Krylov space, and so by computing the norm of the compression of $A$ onto this space, one recovers the norm of $A$ in just $\widetilde{O}(1/\sqrt{\varepsilon})$ iterations.

**Proposition 5.2** (Optimality of precision)**.** *If $\delta > \varepsilon/(1-\varepsilon)$, then $\delta$-approximate power iteration does not always result in a unit vector $x$ with $x^\top Kx \geq (1-\varepsilon)\lambda_1(K)$, even for infinitely many iterations.*

*Proof.* We give an example of a $K$ for which power iteration with adversarial noise immediately stagnates, that is, the adversary may cause $z_t = z_0$ for all $t$. Set

$$\lambda = 1 - \frac{n(\varepsilon + \mathrm{d}\varepsilon)}{n-1}$$

where $\mathrm{d}\varepsilon$ is infinitesimal and consider the matrix $n \times n$

$$K = \begin{bmatrix} 1 \end{bmatrix} \oplus \lambda I.$$

Note that the starting vector $z_0 = \frac{1}{\sqrt{n}}\mathbf{1}$ does not produce the desired approximation as $z_0^\top K z_0 = 1 - (\varepsilon + \mathrm{d}\varepsilon) < 1 - \varepsilon$. Furthermore,

$$z_0 = Kz_0 + e$$

when $e = \frac{1-\lambda}{\sqrt{n}} \begin{bmatrix} 0 & 1 & \cdots & 1 \end{bmatrix}^\top$. If $\|e\| \leq \delta \|Kz_0\|$, then $z_1 = z_0$ is a possible choice for the adversary. To this end, we compute

$$\|e\| = (1-\lambda) \cdot \sqrt{\frac{n-1}{n}} = (\varepsilon + \mathrm{d}\varepsilon)(1 + O(1/n))$$

and

$$\|Kz_0\| = \sqrt{\frac{1}{n} + \frac{n-1}{n}\lambda^2} = (1-\varepsilon)(1+O(1/n)).$$

Thus, for $n$ sufficiently large, the ratio $\|e\| / \|Kz_0\|$ drops below every value larger than $\varepsilon/(1-\varepsilon)$, including $\delta$. □

**Proposition 5.3** (Optimality of iteration count)**.** *Any algorithm which estimates the largest eigenvalue of $K$ using the sequence*

$$z_{t+1} = \texttt{Non-neg-MVP}(K, z_t, \delta)$$

*for $z_0 \in \mathrm{span}(\mathbf{1})$ must use*

$$\frac{\log(\frac{\delta}{2\varepsilon}\sqrt{n})}{\log(1+2\varepsilon)} = \Omega\left(\frac{\log((\delta/\varepsilon)^2 n)}{\varepsilon}\right)$$

*iterations.*

*Proof.* Let $K = I$ and $K' = I + 2\varepsilon e_1 e_1^\top$. Let $z_t, z'_t$ be defined by $z_0 = z'_0 = \mathbf{1}$ and

$$z_{t+1} = \texttt{NN-MVP}(K, z_t) \quad \& \quad z'_{t+1} = \texttt{NN-MVP}(K, z'_t).$$

Then the adversary may make

$$z_t = z'_t \quad \forall t \leq \log(\sqrt{n}/2)/\log(1+2\varepsilon).$$

We prove this by induction. Note the adversary can make $z'_{t+1} = K' z'_t$ exactly. Suppose $z_t = z'_t$. Then

$$\begin{aligned} z_{t+1} &= K z'_t + e \\ &= (K' - 2\varepsilon e_1 e_1^\top) z'_t + e \\ &= z'_{t+1} - 2\varepsilon e_1 e_1^\top z'_t + e. \end{aligned} \tag{9}$$

Now the adversary may set $e = 2\varepsilon e_1 e_1^\top z'_t = 2\varepsilon(1+2\varepsilon)^t e_1$ whenever $2\varepsilon(1+2\varepsilon)^t \leq \delta\sqrt{n}$, which is equivalent to $t \leq \log(\frac{\delta}{2\varepsilon}\sqrt{n})/\log(1+2\varepsilon)$. Thus, the algorithm must give the same output on input $K$ and $K'$, but the range of acceptable outputs for those inputs are disjoint. □

**The non-negativity property.** In our analysis, we crucially used the fact that the top eigenvector $v_1$ has non-negative entries and that the error vector $e$ can be made to have non-negative entries. This implies that all times $t$, $e(t)^\top v_1 \geq 0$. Intuitively, this means that the adversary can only 'cancel out' the contribution of the top eigenvalue by boosting coordinates associated with other eigenvalues; it cannot directly reduce the coordinate associated with the top eigenvalue. This fact is crucially used throughout our analysis. Here we give a simple example showing why without this property, power method cannot work in our (adversarial) noise setting, unless the precision $\delta$ is extremely small.

**Example 2.**

$$K = \begin{pmatrix} 1 & & & \\ & 1/2 & & \\ & & \ddots & \\ & & & 1/2 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

and let the starting vector in the power method be $\begin{pmatrix} \frac{1}{\sqrt{n}} & \cdots & \frac{1}{\sqrt{n}} \end{pmatrix}$. Then

$$Kx = \begin{pmatrix} \frac{1}{\sqrt{n}} & \frac{1}{2\sqrt{n}} & \cdots & \frac{1}{2\sqrt{n}} \end{pmatrix}.$$

Now suppose we no longer have the property that $e(t)^\top v_1 \geq 0$. Then if $\delta = \omega(1/\sqrt{n})$, then in the *very next* iteration, the noisy power method can instead return the vector

$$\begin{pmatrix} 0 & \frac{1}{2\sqrt{n}} & \cdots & \frac{1}{2\sqrt{n}} \end{pmatrix},$$

which satisfies the requirement that the norm of the error is bounded by $\delta\|Kx\|_2 = O(\delta)$. However, now the resulting vector lies completely in the orthogonal complement of the top eigenvector, so no additional iterations of the power method, even without any noise, can give a $1-\varepsilon$ approximation of the top eigenvalue.

# 6 Improved Vector-Marix-Vector Product and Kernel Sum

An immediate application of our faster non-negative matrix-vector product is a fast algorithm for estimating a vector-matrix-vector product $v^\top Kv$ for an entry-wise non-negative vector $v$. Note that we must read all the entries of the vector $v$ (otherwise we cannot detect the all zeros vector versus say a basis vector), meaning that the problem has a $\Omega(n)$ lower bound.

**Theorem 2.3.** *Let $k$ be a kernel admitting a KDE datastructure of Definition 1.1. Let $K \in \mathbb{R}^{n\times n}$ be the associated kernel matrix for $n$ points in $d$ dimensions and let $v \in \mathbb{R}^n$ be an entry-wise non-negative vector. We can compute $v^\top Kv$ up to a $1+\varepsilon$ multiplicative factor in time $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$.*

*Proof.* Without loss of generality, assume $v$ is a unit vector. First note that

$$v^\top Kv = v^\top (K-I)v + \|v\|_2^2 \geq 1,$$

since all the entries of $K-I$ are non-negative. We simply compute an approximation $y$ to the matrix vector product $Kv$ using the algorithm of Theorem 2.1, where $y = Kv + e$ where $e$ is entry-wise non-negative and $\|e\|_2 \leq \varepsilon\|Kv\|_2$. Then we compute $v^\top Ky$. The running time is dominated by the matrix-vector product, which requires $\widetilde{O}\left(\frac{dn^{1+p}}{\varepsilon^{2+2p}}\right)$ time from Theorem 2.1.

For the approximation guarantee, note that $v^\top Ky = v^\top Kv + v^\top e$, where $v^\top e \geq 0$. We additionally have (see Remark 4.1),

$$v^\top e = \sum_{i=1}^n v(i)e(i) \leq \sum_{i=1}^n v(i)\left(\varepsilon v(i) + O\left(\frac{\varepsilon}{\sqrt{n}}\right)\right) \leq \varepsilon + O\left(\frac{\varepsilon}{\sqrt{n}}\right)\sum_{i=1}^n v(i) \leq O(\varepsilon) \leq O(\varepsilon)\cdot v^\top Kv,$$

implying that $v^\top Ky \in (1 \pm O(\varepsilon))v^\top Kv$, as desired.

□

## 6.1 Improved Kernel Matrix Sum

In this section, we give a more involved algorithm for computing $\mathbf{1}^\top K\mathbf{1}$, which is the sum of the entries of the kernel matrix. Recall our notation that $s(K) = \mathbf{1}^\top K\mathbf{1}$ and $s_o(K)$ denotes the off-diagonal sum. Note that the almost trivial $\Omega(n)$ lower bound of computing a general $v^\top Kv$ does not apply here since we already know what all the entries of $\mathbf{1}$ are. Indeed, in this special but practically relevant case (e.g. it can be used to compute the *kernel alignment* problem, a similarity measure between kernel matrices themselves in practice [CSTEK01], and it is a commonly used U-statistic [ABB+23]), much faster algorithms are possible, as discussed in Section 2. Our main theorem is the following:

**Theorem 2.4.** *Let $k$ be a kernel function admitting a KDE datastructure of Definition 1.1. Let $K \in \mathbb{R}^{n\times n}$ be the associated kernel matrix for $n$ points in $d$ dimensions, and $\varepsilon \in (0,1)$. In time $\widetilde{O}\left(\frac{n^{\frac{1}{2}+\frac{p}{2}}}{\varepsilon^4}\right)$, Algorithm 3 the sum of the entries of $K$ up to a $1+\varepsilon$ factor with probability $\geq 0.99$.*

**Intuition Behind the Algorithm and Proof.** Before presenting formal details, we give a high-level intuitive overview of what is happening under the hood of our algorithm, Algorithm 3. The proof proceeds in three steps.

- **Step 1:** In the first step, we show that a random $m \times m$ principal submatrix $K_A$ for $m = \Theta(\sqrt{n}/\varepsilon^2)$ preserves the kernel sum. More precisely, we show via a concentration bound calculation in Lemma 6.1 that it suffices to compute the sum of the *off-diagonal* entries of $K_A$ up to additive $\text{poly}(1/\varepsilon)$ to get a $1+\varepsilon$ multiplicative approximation of $\mathbf{1}^\top K \mathbf{1}$. We prove this is the optimal sampling complexity to use; see Theorem 2.5. This step extends the idea of [BIMW21] who only sampled a $\approx \sqrt{n} \times \sqrt{n}$ submatrix (and repeated their overall procedure $1/\varepsilon^2$ times). While this strategy is entirely valid, it can be lossy in $1/\varepsilon$ factors in the final bound, and sampling $\Theta(\sqrt{n}/\varepsilon^2)$ rows/columns from the very beginning allows us to better tune parameters of the KDE datastructures that are ultimately used.

- **Step 2:** Now the task is to approximate the off-diagonal sum of $K_A$. A natural strategy is to just initiate $m$ KDE queries with an appropriate additive error. (One caveat is that the KDE sum should not contain a diagonal term, but this can be easily dealt with; see 6.1). This strategy was already explored in [BIMW21], but it turns out that one needs to set the additive error $\mu$ quite low, leading to an overall $n^{1/2+p}$ running time of the final algorithm (note that we are aiming for $n^{1/2+p/2}$). Instead, we follow the high-level idea of [BIMW21] in obtaining their final bound: we filter out all "heavy" rows of $K_A$ using KDE queries with a relatively large $\mu$ (which is faster than the prior strategy). Then for the remaining "light" rows, we know their individual contribution to the off-diagonal sum is bounded by $\approx \mu\sqrt{n}$. This means that instead of computing their contribution exactly or via KDE queries, one can further *subsample* light rows to determine their total contribution to the off-diagonal sum of $K_A$. Optimizing this, [BIMW21] prove an overall running time of $n^{\frac{2+5p}{4+2p}}$ where $\frac{2+5p}{4+2p} = \frac{1}{2} + \frac{2p}{p+2}$. Since $p$ is a relatively small constant, e.g. $p \approx 0.173$ in the Gaussian kernel case (see Table 3), the improvement over $1/2 + p$ is relatively minor.

- **Step 3:** This is the main technical difference between our algorithm and that of [BIMW21], which gives us the improved running time. Once we subsample light rows, we have no choice but to either compute their sum *exactly* (which is what [BIMW21] opt to do). We could use KDE queries since the sample size can be shown to be $\ll m$, which actually would lead to our improved running time. However, the issue here is that *creating* the KDE datastructure is too costly now since there are still $m$ vectors in the KDE sum (corresponding to the columns since only the rows have been sub-sampled). Intuitively, what is happening is that KDE queries don't really outweigh the benefit of exact sum unless one makes $\Omega(m)$ queries, where $m$ is the total number of vectors in the KDE datastructure. This does not appear to be the case here since there are relatively very few light rows, meaning few queries, (after sub-sampling) compared to the number of columns.

  Thus, we employ a different sampling strategy. Once we filter out the heavy rows, we note that this also filters out the heavy *columns* since the matrix $K_A$ is symmetric. Then we sample a square *principal submatrix* of $K_A$, rather than just sampling rows as done in [BIMW21]. Thankfully, the same variance valuations we did in Lemma 6.1 for Step 1 can be used to analyze this sampling as well. The advantage of our sampling is now the number of rows/columns of the sub-sampled matrix, which recall we sub-sampled from only light rows and columns, is *balanced*, meaning this is exactly the regime where KDE queries are useful (the number of KDE queries will be approximately equal to the total umber of vectors in the datastructure)! Thus, we initialize an appropriate KDE datastructure on the columns of this sampled square matrix and query all the rows to compute a sufficiently fine-grained contribution of the light

rows to $s_o(K_A)$. By carefully balancing parameters across the three steps, we are able to prove a final running time of $n^{1/2+p/2}$, which we additionally argue below is the "natural" limit of all KDE based algorithms.

### 6.1.1 Main Sampling Lemma

In this subsection, we prove our main sampling workhorse lemma. For a subset $A \subseteq [n]$, let $K_A$ be the principal submatrix of the kernel matrix where we keep the rows and columns indexed by $A$. The following lemma generalizes Lemma 6 in [BIMW21].

**Lemma 6.1.** *Let $K$ be a symmetric (not necessarily kernel) matrix with all entries in $[0, \alpha]$ for a parameter $0 \leq \alpha \leq 1$. Further suppose that all the row and column sums of $K$, ignoring the diagonal, are bounded by a parameter $\beta \geq 0$. Let $A \subseteq [n]$ be a random set where we pick every element of $[n]$ to be in $A$ with probability $q$ independently. Let $Z = \operatorname{diag}(K) + s_o(K_A)/q^2$, where $s_o(K_A)$ denotes the sum of the off diagonal entries of $K_A$, and $\operatorname{diag}(K)$ denotes the sum of the diagonal of $K$. Then*

- $\mathbb{E}[Z] = s(K)$,
- $\operatorname{Var}[Z] \leq O(\alpha q^{-2} + \beta q^{-1}) \cdot s(K)$.

*Proof.* The expectation is clear so we just have to bound the variance. Since variance is preserved under additive shifts, we instead consider $\operatorname{Var}[Z - \operatorname{diag}(K)] = \operatorname{Var}[s_o(K_A)/q^2]$. The expectation of the square is:

$$\mathbb{E}\left[\left(\frac{s_o(K_A)}{q^2}\right)^2\right] = \frac{1}{q^4}\left(q^2 \sum_{i \neq j} K_{i,j}^2 + 2q^3 \sum_{|\{i,j,j'\}|=3} K_{i,j}K_{i,j'} + q^4 \sum_{|\{i,j,i',j'\}|=4} K_{i,j}K_{i',j'}\right). \tag{10}$$

We bound each term separately. The first term satisfies

$$q^{-2} \sum_{i \neq j} K_{i,j}^2 \leq q^{-2}\alpha \sum_{i,j} K_{i,j} = q^{-2}\alpha s(K).$$

For the third term, we have

$$\sum_{|\{i,j,i',j'\}|=4} K_{i,j}K_{i',j'} \leq \sum_{i,j,i',j',i\neq j,i'\neq j'} K_{i,j}K_{i',j'} \leq (s(K) - \operatorname{diag}(K))^2.$$

Since

$$\mathbb{E}[s_o(K_A)/q^2]^2 = (s(K) - \operatorname{diag}(K))^2,$$

this term directly cancels with the third term in Equation (10) in the variance calculation. Thus, it remains to bound the second term in Equation (10) We have

$$\sum_{|\{i,j,j'\}|=3} K_{i,j}K_{i,j'} \leq O\left(\sum_i s_{o,i}(K)^2\right) \leq O(\max_i s_{o,i}(K) \cdot s(K)),$$

where $s_{o,i}(K)$ denotes the sum of the $i$th row of $K$, ignoring the diagonal. This is because every entry in $K$ is only multiplied by other entries sharing the same row and column, and $K$ is symmetric. This implies that the second term in Equation (10) can be bounded, up to constant factors, by

$$q^{-1} \cdot \max_i s_{o,i}(K) \cdot s(K) \leq q^{-1}\beta s(K),$$

by our definition of $\beta$. The lemma follows from putting together our calculations. □

A useful corollary is the case of the previous lemma applied to a kernel matrix $K$.

**Corollary 6.2.** *Consider the setting of Lemma 6.1 and let $K$ be be a kernel matrix (as defined in Section 3) with $q = C(\varepsilon^2\sqrt{n})^{-1}$ for a large constant $C \geq 1$. Consider the random variable $Z = n + s_o(K_A)/q^2$ from Lemma 6.1. We have*

- $\mathbb{E}[Z] = s(K)$,
- $\mathrm{Var}[Z] \leq 0.001 \cdot \varepsilon^2 \cdot s(K)^2$.

*Proof.* We set $\alpha = 1$ in Lemma 6.1. Now Lemma 5 in [BIMW21] gives

$$\max_i s_{o,i}(K) \leq O(\sqrt{n} + \sqrt{s_o(K)}),$$

where $s_o(K) = \sum_i s_{o,i}(K)$ denotes the sum of all the off diagonal entries of $K$. Denote this quantity by $\beta$. We have

$$\alpha q^{-2} + \beta q^{-1} \leq O(\varepsilon^4 n + \varepsilon^2\sqrt{n} \cdot (\sqrt{n} + \sqrt{s_o(K)})) = O(\varepsilon^4 n + \varepsilon^2 n + \varepsilon^2\sqrt{n s(K)}).$$

Using the fact $s(K) \geq n$ gives us that the above quantity is bounded by $O(\varepsilon^2 s(K))$, so $(\alpha q^{-2} + \beta q^{-1}) \cdot s(K) \leq O(\varepsilon^2 s(K)^2)$. Increasing the constant $C$ in the definition of $q$ as necessary completes the proof. □

### 6.2 A Faster Algorithm for the Kernel Sum

Now we present our faster algorithm for computing $s(K)$, the sum of all the entries of the kernel matrix, up to $1 + \varepsilon$ multiplicative factor.

**Algorithm 3** Faster algorithm for Kernel Sum

1: **procedure** FAST-KERNELSUM($X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$)
2: $q_1 \leftarrow \min\left(\frac{C}{\varepsilon^2\sqrt{n}}, 1\right)$ for a large constant $C \geq 1$
3: Sample every $i \in [n]$ with probability $q_1$ to form subset $A = \{a_1, \ldots, a_m\} \subseteq \{x_1, \ldots, x_n\}$. ▷ *Defines a principal submatrix $K_A$ of $K$.*
4: Let $B = \{a_i \in A \mid \mathcal{D}_{A\setminus a_i}(a_i) \geq \tau\}$, where we set $\tau = m\varepsilon^3/C$ and additive error $\mu = \varepsilon\tau/C$ in the construction of the KDE datastructure ▷ *Note the query $\mathcal{D}_{A\setminus a_i}(a_i)$* ***excludes*** *$a_i$. Remark 6.1 details how this can be achieved with a logarithmic overhead via a standard trick.*
5: $\hat{S}_1 \leftarrow \sum_{b\in B} \mathcal{D}_{B\setminus b}(b)$ where we set additive error $\mu$ ▷ `Estimate of` $s_o(K_B)$
6: $\hat{S}_2 \leftarrow \sum_{b\in B} \mathcal{D}_{A\setminus b}(b)$ where we set additive error $\mu$ ▷ `Estimate of` $s_o(K_{B\times A})$
7: $\hat{S}_3 \leftarrow 2\hat{S}_2 - \hat{S}_1$
8: $q_2 \leftarrow \min(C\varepsilon^{1.5}\sqrt{m\tau}, 1)$ for $\tau$ defined line Line 4
9: Sample every row and column in $K_{A\setminus B}$ with probability $q_2$ to form subset $B' \subseteq A \setminus B$
10: $\hat{S}_4 \leftarrow \sum_{b'\in B'} \mathcal{D}_{B'\setminus b'}(b')$ with KDE additive error $\mu' = \widetilde{O}\left(\frac{\sqrt{\tau}}{\varepsilon^{1.5}\sqrt{m}}\right)$ ▷ `Estimate of` $s_o(K_{B'})$
11: **Return** $n + q_1^{-2} \cdot (\hat{S}_3 + q_2^{-2} \cdot \hat{S}_4)$ as the approximation to $s(K)$, the sum of entries of $K$
12: **end procedure**

*Proof of Theorem 2.4.* First we show the approximation guarantee. The proof will be broken down in three steps which we outline before presenting the full details. The first step samples a $\approx \frac{\sqrt{n}}{\varepsilon^2} \times \frac{\sqrt{n}}{\varepsilon^2}$ principal submatrix $K_A$ of $K$, and we show that it suffices to approximate the off-diagonal sum $s_o(K_A)$ up to $1+\varepsilon$ multiplicative and poly$(1/\varepsilon)$ additive error. To approximate $s_o(K_A)$, we divide the rows and columns of $K_A$ into "heavy" and "light" groups, where heavy means their off-diagonal sum is at least $m\tau$. Thus, the second step removes all the heavy rows and columns of $K_A$, detected using appropriately-tuned KDE queries, and computes a $1+\varepsilon$ multiplicative approximation of their contribution to $s_o(K_A)$. Finally, we need to deal with the light principal submatrix left. Here, we have the guarantee that their row and column sums are bounded by $m\tau$. So in the third step, we perform another sampling to substantially reduce the size of the light submatrix, which is only possible due to the bounded row/column sum guarantee. Crucially, the sub-sampled matrix remains square, which allows us to take full advantage of KDE queries again.

**Step 1.** Define the random variable $Z = n + s_o(K_A)/q_1^2$, where $K_A$ is the $m \times m$ matrix with rows and columns indexed by the random set $A = \{a_1, \ldots, a_m\}$ sampled in Line 3 of Algorithm 3, and $s_o(\cdot)$ denotes the off-diagonal sum. Corollary 6.2, along with Chebyshev's inequality, implies that

$$|Z - \mathbb{E}[Z]| = |Z - s(K)| \leq \varepsilon s(K),$$

with probability $\geq 99\%$. We condition on this event for the rest of the proof. Since $q_1^2 = O(\varepsilon^4 n)$, it suffices to estimate $s_o(K_A)$ up to a multiplicative $1+\varepsilon$ factor and additive error $O(1/\varepsilon^3)$. The latter is because this will imply our additive error in computing $Z$ is bounded by $O(q_1^2 \cdot 1/\varepsilon^3) \leq O(\varepsilon n) \leq O(\varepsilon s(K))$, noting that $s(K) \geq n$ due to the diagonal.

Thus, for the rest of the proof, we will show that the algorithm estimates $s_o(K_A)$ up to $1+\varepsilon$ multiplicative factor and additive error $O(1/\varepsilon^3)$ (by increasing the constant $C$ in $q_1$, we can make the additive error any small constant multiple of $1/\varepsilon^3$).

**Step 2.** Towards this end, we have

$$s_o(K_A) = s_o(K_B) + s_o(K_{A\setminus B}) + 2s,$$

where $s_o(K_B)$ is the sum of the off-diagonal entries of the submatrix $K_B$ and similarly $s_o(K_{A\setminus B})$ is the sum of the off-diagonal entries of the submatrix $K_{A\setminus B}$. The $2s$ term comes from the two rectangles left over from carving out $K_B$ and $K_{A\setminus B}$ from $K_A$ (see Figure 2).

Note that $B$ contains all the rows of $K_A$ where the off-diagonal sum is at least $m\tau/2$, for $\tau$ defined in Line 4 of Algorithm 3, since the additive error in the KDE estimate is $O(\varepsilon\tau)$ (and we multiply by $m$ since a KDE query returns an approximation to the scaled sum). Thus, every row of $K_A$ in the set $A \setminus B$ must also have its off-diagonal sum bounded by $m\tau/2$. In particular, if we consider the submatrix $K_{A\setminus B}$, every off-diagonal row and column sum of this matrix is bounded by $m\tau/2$.

Now we show how to obtain a $1+\varepsilon$ approximation to $s_o(K_B)$ and $2s$. Note that $\sum_{a\in B} \mathcal{D}_{A\setminus a}(a)$, where the additive error in the KDE queries are all set to $\mu$ as defined in Line 4 of Algorithm 3, is a $1 + O(\varepsilon)$ approximation to $s_o(K_B) + s$. This is because the additive error $\mu$ is at most $O(\varepsilon)$ times the sum of the off-diagonal values of the rows in $B$, since $B$ consists of all the $a_i \in A$ where $\mathcal{D}_{A\setminus a_i}(a_i) \geq \tau$, meaning the additive error can be absorbed into the multiplicative one. Furthermore, the sum $\sum_{b\in B} \mathcal{D}_{B\setminus b}(b)$, where again we use the same additive error $\mu$, is a multiplicative $1+\varepsilon$ and additive $\mu|B|$ approximation to $s_o(K_B)$, and thus

$$2 \cdot \left( \sum_{a\in B} \mathcal{D}_{A\setminus a}(a) \right) - \sum_{b\in B} \mathcal{D}_{B\setminus b}(b)$$

is a $1 + O(\varepsilon)$ multiplicative approximation to $s_o(K_B) + 2s$, since the additive error $O(\mu|B|m) = O(\varepsilon\tau|B|m)$ is at most $\varepsilon$ fraction of $\Omega(\tau|B|m) \leq s_o(K_B) + s \leq s_o(K_B) + 2s$.

**Step 3.** It remains to deal with the term $s_o(K_{A\setminus B})$. Towards that end, note that all the rows and columns of $K_{A\setminus B}$ have their entries bounded by $O(m\tau)$ (since we removed all "large" rows and columns in $B$). Then Lemma 6.1 implies that if we sample the rows and columns of $K_{A\setminus B}$ with probability $q_2$ (as defined in Line 6 of Algorithm 3) and compute the (scaled) sum of off-diagonal entries of the resulting matrix $K'$, then

$$\mathrm{Var}\left[\frac{s_o(K')}{q_2^2}\right] \leq O(m\tau q_2^{-2} + m\tau q_2^{-1}) \cdot s_o(K_{A\setminus B}),$$

where we set both $\alpha = \beta = O(m\tau)$ in the lemma statement and note that we can ignore the diagonal contributions since we are effectively setting the diagonal as zeros by only considering the off-diagonal sum. We have

$$O(m\tau q_2^{-2} + m\tau q_2^{-1}) \cdot s_o(K_{A\setminus B}) = O(m\tau q_2^{-2}) \cdot s_o(K_{A\setminus B}) \leq O\left(\frac{s_o(K_{A\setminus B})}{\varepsilon^3}\right).$$

By Chebyshev's inequality, we have

$$\begin{aligned}\Pr\left(\left|\frac{s_o(K')}{q_2^2} - s_o(K_{A\setminus B})\right| \geq \varepsilon s_o(K_{A\setminus B}) + \frac{1}{\varepsilon^3}\right) &\leq \frac{\mathrm{Var}\left[\frac{s_o(K')}{q_2^2}\right]}{\left(s_o(K_{A\setminus B}) + \frac{1}{\varepsilon^3}\right)^2} \\ &\leq \frac{\varepsilon^3 \mathrm{Var}\left[\frac{s_o(K')}{q_2^2}\right]}{2s_o(K_{A\setminus B})},\end{aligned}$$

where the last step is due to

$$\left(s_o(K_{A\setminus B}) + \frac{1}{\varepsilon^3}\right)^2 \geq \frac{2s_o(K_{A\setminus B})}{\varepsilon^3}.$$

Plugging in

$$\mathrm{Var}\left[\frac{s_o(K')}{q_2^2}\right] \leq O\left(\frac{s_o(K_{A\setminus B})}{\varepsilon^3}\right),$$

and adjusting the constant $C$ in the definition of $q_2$ to be sufficiently large, shows that

$$\left|\frac{s_o(K')}{q_2^2} - s_o(K_{A\setminus B})\right| \leq \varepsilon s_o(K_{A\setminus B}) + \frac{1}{\varepsilon^3}$$

with probability at least 0.999.

Thus, it suffices to compute the value $s_o(K')$. We will do this with additive error $\mu' = \widetilde{O}\left(\frac{\sqrt{\tau}}{\varepsilon^{1.5}\sqrt{m}}\right)$. With this setting, the overall additive error in computing $s_o(K')$ is $\mu' m'$ and we have

$$\frac{\mu' m'}{q_2^2} \leq \widetilde{O}\left(\frac{\sqrt{\tau}}{\varepsilon^{1.5}\sqrt{m}} \cdot \frac{m'}{\varepsilon^3 m\tau}\right) = \widetilde{O}\left(\frac{1}{\varepsilon^3} \cdot \frac{m'}{\varepsilon^{1.5} m\sqrt{m\tau}}\right),$$

and we exactly have $m' = \widetilde{O}(\varepsilon^{1.5} m\sqrt{m\tau})$ with high probability. Thus by decreasing the value of $\mu'$ by logarithmic factors, we have

$$\frac{\mu' m'}{q_2^2} \leq \frac{1}{\varepsilon^3},$$

so this value of $\mu'$ suffices for an additive $1/\varepsilon^3$ error (which contributes to a $1/\varepsilon^3$ additive error for our estimate of $s_o(K_A)$ which we can tolerate).

**Putting Everything Together.** We know $s_o(K_A) = s_o(K_B) + s_o(K_{A\setminus B}) + 2s$, and all three terms are non-negative. We obtained a $1+\varepsilon$ multiplicative approximation to $s_o(K_B) + 2s$ in Step 2 and a $1+\varepsilon$ multiplicative and $1/\varepsilon^3$ additive approximation to $s_o(K_{A\setminus B})$ in Step 3, which gives us a $1+\varepsilon$ multiplicative and $1/\varepsilon^3$ additive approximation to $s_o(K_A)$, as desired. This proves the approximation guarantee.

**Proving the Running Time.** For the running time, we know Step 2 takes $\widetilde{O}\left(\frac{m}{\varepsilon^2(\varepsilon\tau)^p}\right)$ time. For Step 3, $K'$ is a $m' \times m'$ matrix with $m' = \widetilde{O}(\varepsilon^{1.5} m\sqrt{m\tau})$ with high probability. We compute the sum of the entries of $K'$ using $m'$ KDE queries with additive error $\mu' = \widetilde{O}\left(\frac{\sqrt{\tau}}{\varepsilon^{1.5}\sqrt{m}}\right)$, giving a runtime of

$$\widetilde{O}\left(\frac{m'}{\varepsilon^2(\mu')^p}\right) = \widetilde{O}\left(m^{p/2+3/2}\tau^{1/2-p/2}\varepsilon^{3p/2-1/2}\right).$$

To balance, if we set the expressions

$$m^{p/2+3/2}\tau^{1/2-p/2}\varepsilon^{3p/2-1/2} = \frac{m}{\varepsilon^2(\varepsilon\tau)^p},$$

where the first term corresponds to the running time of computing $s_o(K_{A\setminus B})$ and the second term corresponds to the running time of computing $s_o(K_B)$ (ignoring logarithmic factors), this would solve to

$$\frac{1}{\tau} = \left(m^{1/2+p/2}\varepsilon^{3p/2+3/2}\right)^{\frac{2}{p+1}} = m\varepsilon^3,$$

which is (asymptotically) our choice of $\tau$. Thus, neither term dominates, meaning that the runtime can be bounded by

$$\widetilde{O}\left(\frac{m}{\varepsilon^{2+p}} \cdot \left(m^{1/2+p/2}\varepsilon^{3p/2+3/2}\right)^{\frac{2p}{p+1}}\right) = \widetilde{O}\left(\frac{m^{1+p}}{\varepsilon^{2-2p}}\right).$$

Noting that $m = \Theta(\sqrt{n}/\varepsilon^2)$ with high probability, we have that the total runtime is bounded by

$$\widetilde{O}\left(\frac{n^{\frac{1}{2}+\frac{p}{2}}}{\varepsilon^4}\right).$$

Finally, note that by using the standard trick of repeating $O(\log n)$ times and taking the median, we can boost the success probability of Algorithm 3 to $1 - 1/n^{\Theta(1)}$.

□

**Remark 6.1.** We briefly mention how to construct a KDE datastructure $\mathcal{D}'$ over a dataset $X = \{x_1, \ldots, x_n\}$ (with some desired relative error $1+\varepsilon$ and additive error $\mu$), capable of answering queries $\mathcal{D}_{X\setminus x_i}(y)$ for a given query $y$ and $x_i \in X$. In other words, the sum excludes any desired vector $x_i$. Note that this same construction appears in many prior works, including [BIMW21] and [BIK+23], but we briefly mention it here for completeness.

Assume without loss of generality that $n$ is a power of 2. Let $\mu' = \mu/(100 \log n)$. We build a data structure for points $x_1, \ldots, x_{n/2}$ and another for $x_{n/2+1}, \ldots, x_n$, both with additive error $\mu'$. We additionally build 4 data structures for sets $\{x_1, \ldots, x_{n/4}\}, \{x_{n/4+1}, \ldots, x_{n/2}\}, \{x_{n/2+1}, \ldots, x_{3n/4}\}$, and $\{x_{3n/4+1}, \ldots, x_n\}$. We continue this pattern for $O(\log n)$ levels which naturally corresponds to a binary tree with the interval $\{1, \cdots, n\}$ at the top and every interval has two children representing its first and second half. Then to estimate the sum $\frac{1}{n}\sum_{j=1, j\neq i}^{n} k(y, x_j)$ for a given query $y$ and index $i$ to exclude, we represent $\{1, \ldots, i-1\} \cup \{i+1, \ldots, n\}$ as the disjoint union of $O(\log n)$ intervals represented in the binary tree. All these values are positive and the total additive error is $O(\mu' \cdot \log n) \leq \mu$. It can be easily computed that the total construction and query times only blow up by $\mathrm{poly}(\log n)$ factors.

### 6.2.1 Limits of KDE-Based Algorithms for the Kernel Sum?

We briefly argue why $\Omega(n^{1/2+p/2}/\varepsilon^{4-p})$ is a natural limit of our ideas. Theorem 2.5 shows that we must sample $m = \Omega(\sqrt{n}/\varepsilon^2)$ points in the input set $X$; otherwise we have no hopes of obtaining a $1+\varepsilon$ multiplicative approximation. However, this is only a sample complexity lower bound. We now argue how the extra term $n^{p/2}/\varepsilon^{2-p}$ is a natural limitation for processing these $m$ sampled points.

After we sample, obtaining a $1+\varepsilon$ multiplicative approximation is equivalent to obtaining a $1/\varepsilon^3$ additive error of the off diagonal entries of the sampled set, since we multiply the estimator by $\varepsilon^4 n$ and the original kernel sum is at least $n$. Thus, we must detect every row with off-diagonal sum at least $1/\varepsilon^3$, otherwise our additive error is already off by too much. The only reasonable way to do this seems to be test every row with a KDE query, which requires setting $\mu = 1/(m\varepsilon^3) \leq 1/(\sqrt{n}\varepsilon)$, giving a running time of at least $n^{p/2}\varepsilon^p/\varepsilon^2$ per row, (using our definition of a KDE query in Definition 1.1) and an overall running time of at least $n^{1/2+p/2}/\varepsilon^{4-p}$. Thus, to improve upon our result by more than an $\varepsilon^p$ factor, one would need a substantially different algorithm for detecting all rows with sum larger than $1/\varepsilon^3$, besides computing $m$ KDE queries with the appropriate additive error.

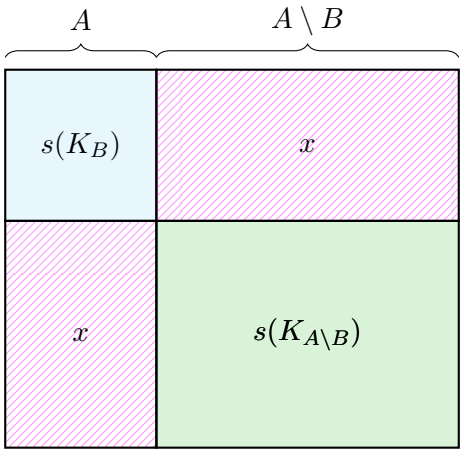


Figure 2: In Algorithm 3, we need to compute the sum $s_o(K_A)$, which can be broken down as $s_o(K_B) + s_o(K) + 2x$, where $x$ represents one of the pink rectangles.

### 6.3 Optimal Sampling Lower Bound

Our algorithm for computing $\mathbf{1}^\top K\mathbf{1}$ (Algorithm 3) samples $\Theta(\sqrt{n}/\varepsilon^2)$ data points to compute the sum, which we show is optimal. This is a strengthening of a similar result in [BIMW21] which proved a $\Omega(\sqrt{n})$ lower bound.

We consider the following distribution $\mathcal{D}(p)$ over vectors in $\mathbb{R}^n$ parameterized by $p \in [0,1]$:

- With probability $1-p$, return $n^{100}$ times a uniformly random basis vector.
- With probability $p$, return the origin 0.

**Lemma 6.3.** *Let $p_1 = (1+\varepsilon)/\sqrt{n}, p_2 = 1/\sqrt{n}$ and assume $n$ is sufficiently large. Consider the kernel matrices $K_1$ and $K_2$ corresponding to $n$ points drawn from $\mathcal{D}(p_1)$ and $\mathcal{D}(p_2)$ respectively. Let $s(\cdot)$ denote the sum of the entries. With probability at least $1 - \exp(-\Omega(\varepsilon^2\sqrt{n}))$, we have:*

1. $\mathbb{E}[s(K_1)] \geq (3+\varepsilon/2)n$.
2. $\mathbb{E}[s(K_2)] \leq (3+\varepsilon/4)n$.
3. $\Pr(|s(K_i) - \mathbb{E}[s(K_i)]| \geq \varepsilon n/100) \leq 10^{-3}$.

*Proof.* Among the $n$ points (either from $\mathcal{D}(p_1)$ or $\mathcal{D}(p_2)$, for $x \in \{0, e_1, \ldots, e_n\}$, let $c_x$ denote the number of points in the $n$ points that are copies of $x$. Then the kernel sum is equal to

$$\sum_{x\in\{0,e_1,\ldots,e_n\}} c_x^2 + o(e^{-n}).$$

Thus,

$$\begin{aligned}
\mathbb{E}[s(K_i)] &= n\,\mathbb{E}[c_{e_1}^2] + \mathbb{E}[c_o^2] + o(e^{-n}) \\
&= n^2(n-1)\left(\frac{1-p_i}{n}\right)^2 + n(1-p_i) + n(n-1)p_i^2 + np_i + o(e^{-n}) \\
&= 2n + n^2p_i^2 + o(n),
\end{aligned}$$

and the first two claims follow.

We now show concentration of the random variable $\sum_{x\in\{0,e_1,\ldots,e_n\}} c_x^2$. For $x \in \{e_1, \ldots, e_n\}$, write

$$c_x^2 = c_x^2 \cdot \mathbf{1}\{c_x \le 100\log n\} + c_x^2 \cdot \mathbf{1}\{c_x > 100\log n\}.$$

Then since each fixed $c_x$ is a binomial random variable, the Chernoff bound implies that in either case ($i = 0$ or $i = 1$),

$$\Pr(\exists c_x > 100\log n) \le n\Pr(c_1 > 100\log n) \le n \cdot e^{-50\sqrt{n}\cdot\frac{1}{\sqrt{n}}} < \frac{1}{n^{10}}.$$

Thus,

$$\mathbb{E}\left[\sum_x c_x^2 \cdot \mathbf{1}\{c_x > 100\log n\}\right] = o(1),$$

and the probability that $\sum_x c_x^2 \cdot \mathbf{1}\{c_x > 100\log n\}$ deviates by more than $\varepsilon n/200$ from its expected value is at most $1/n$ by Markov's inequality.

To handle $c_x^2 \cdot \mathbf{1}\{c_x \le 100\log n\}$, we note that changing any one of the sampled points can only change $\sum_x c_x^2 \cdot \mathbf{1}\{c_x \le 100\log n\}$ by $\mathrm{poly}(\log n)$ factors, so McDiarmid's Inequality inequality (applied to the appropriate Doob martingale) implies

$$\Pr\left(\left|\sum_x c_x^2 \cdot \mathbf{1}\{c_x \le 100\log n\} - \mathbb{E}\left[\sum_x c_x^2 \cdot \mathbf{1}\{c_x \le 100\log n\}\right]\right| \ge \frac{\varepsilon n}{200}\right) \le e^{-\Omega\left(\frac{\varepsilon^2 n^2}{n\cdot\mathrm{poly}(\log n)}\right)} \le \frac{1}{n}$$

for sufficiently large $n$. Combining our two tail bounds with the triangle inequality and the union bound proves claim (3). □

**Theorem 2.5.** *Suppose $n \ge \Omega(1/\varepsilon^2)$. Consider an algorithm which specifies a subset $T \subseteq [n]$, receives the set of points $\{x_i, i \in T\}$ and returns a $1+\varepsilon/100$ approximation to $s(K)$, where $K$ is the underlying kernel matrix of the full set of $n$ points, with probability at least 99%. Then we must have $|T| \ge \Omega(\sqrt{n}/\varepsilon^2)$.*

*Proof.* Suppose we flip an unbiased coin and either give the algorithm points drawn from $\mathcal{D}(p_1)$ or $\mathcal{D}(p_2)$. If $n$ is sufficiently large, then from Lemma 6.3, we know that with probability at least 99% that the kernel sum under case 1 is at least a $1+\varepsilon/2$ factor larger than the kernel sum under case 2. Thus, an algorithm which returns a $1+\varepsilon/100$ approximation can determine, with probability at least 99%, which distribution we are drawing from.

Let $x_1$ and $x_2$ be a sample from $\mathcal{D}(p_1)$ and $\mathcal{D}(p_2)$ respectively. Then the Hellinger distance, $d_H$, between $x_1$ and $x_2$ is bounded up to constant factors by

$$\sqrt{n}\cdot(\sqrt{1-p_1} - \sqrt{1-p_2})^2 + (\sqrt{p_1} - \sqrt{p_2})^2.$$

The first term can be bounded by

$$(\sqrt{\sqrt{n} - (1+\varepsilon)} - \sqrt{\sqrt{n}-1})^2 = \left(\frac{(\sqrt{n} - (1+\varepsilon)) - (\sqrt{n}-1)}{\sqrt{(\sqrt{n}-(1+\varepsilon))(\sqrt{n}-1)}}\right)^2 = O\left(\frac{\varepsilon^2}{n}\right).$$

The second term is bounded by

$$\left(\sqrt{\frac{1+\varepsilon}{\sqrt{n}}} - \sqrt{\frac{1}{\sqrt{n}}}\right)^2 = O\left(\frac{\varepsilon^2}{\sqrt{n}}\right),$$

so

$$d_H(x_1, x_2) \leq O\left(\frac{\varepsilon^2}{\sqrt{n}}\right).$$

Then by a standard relationship between total variation and Hellinger distance, we have that the total variation distance between $|T|$ samples from $\mathcal{D}(p_1)$ and $\mathcal{D}(p_2)$ is bounded by $O\left(\sqrt{|T|} \cdot \frac{\varepsilon}{n^{1/4}}\right)$. Thus if $|T| \ll \sqrt{n}/\varepsilon^2$, then the total varaition distance is bounded by 0.001, contradicting the fact that we can distinguish the two distributions with probability $> 99\%$.

□

# 7 Towards a Lower Bound for Matrix-Vector-Product for Negative Vectors

The guarantees of Theorem 2.1 naturally lead us to ask if we can obtain such a guarantee for *arbitrary* vectors (with both positive and negative coordinates). Prior works which also studied matrix-vector products for kernel matrices ([CS17, BIMW21, IKSW25]) also require the input vector to be entry-wise non-negative to obtain a relative-error guarantee. Furthermore, it has been previously conjectured that the arbitrary vector case may require nearly quadratic time. Indeed, [IKSW25] state that for this task "in general it may require computing $Kx$ exactly, which takes $\Omega(n^2)$ time". Unfortunately, the evidence given in [IKSW25] is not formal[5]. More subtly, the evidence of [IKSW25] actually *does not* require $\Omega(n^2)$ time, nor does it require computing $Kx$ using an $x$ with negative entries. In more detail, the argument of [IKSW25] proceeds in two steps.

1. Having error $\varepsilon\|Kx\|_2$ actually implies 0 error if it is the case that $Kx = 0$.
2. In general, computing $Kx$ exactly requires $\Omega(n^2)$ time.

There are issues with both steps of the reasoning. For (1), it may not always be the case that such an $x$ exists. Furthermore, finding such an $x$ could be computationally difficult itself. And lastly, if it is known that $Kx = 0$ beforehand, then computing $Kx$ is very easy, we can simply return 0!

For step (2), the hard instance used in [IKSW25] is a matrix of all zeros with a hidden 1. This cannot happen in Kernel matrices since the diagonals are all 1's. Furthermore, detecting a 1 is easy to do in linear time since it implies two vectors are identical (due to the fact that in all the kernels we study and all the natural kernels that we are aware of, $k(x, y) = 0 \iff x = y$). It *is* true that computing $Kx$ exactly in general should require $\Omega(n^2)$ time, but this also holds for non-negative $x$. Indeed, [BIS17] show that computing the sum of entries of $K$ *exactly* requires $\Omega(n^2)$ time assuming SETH, and the sum can be easily deduced if we can compute $K\mathbf{1}$ exactly.

Thus, there remains a big gap in our understanding of the complexity of matrix-vector products for kernel matrices. On one hand, we have near linear upper bounds for the non-negative case with

[5] note that the authors are not claiming a formal statement, rather giving a heuristic derivation, so the following discussion does not contradict any of their formal theorems in the paper

strong relative-error grantees. On the other hand, virtually no non-trivial upper or lower bounds are known for the general case.

Our aim is to bridge this gap and provide a stronger evidence for the hardness of computing relative error estimates of $Kx$ for general $x$. This task seems out of reach of our techniques, but we show that a related intermediate problem requires $\Omega(n^2)$ time, assuming SETH. The intermediate problem is a slight generalization of computing $Kx$ for general $x$, where the kernel matrix $K$ can be "asymmetric" up to *one* vector.

In more detail, we consider computing matrix-vector products for kernel matrices $K$ where the rows correspond to a point set $X = \{x_1, \cdots, x_n\}$ as usual and the columns correspond to the same point set $X$ plus the origin 0. Thus in this subsection, we assume $K \in \mathbb{R}^{n \times n+1}$ where $K_{ij} = k(x_i, x_j)$ for $1 \leq i \leq n, 1 \leq j \leq n$ and $K_{ij} = k(x_i, 0)$ for $j = n+1$ and $1 \leq i \leq n$.

Our main theorem is the following:

**Theorem 2.6.** *Let $X = \{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ be input points with $d = \omega(\log n)$. Define $K \in \mathbb{R}^{n \times n+1}$ to be the Gaussian kernel matrix where the rows are indexed by $X$ and the columns are indexed by $X \cup \{0\}$. Any algorithm which takes as input $X$ and a vector $x \in \mathbb{R}^{n+1}$, and returns a $y \in \mathbb{R}^n$ such that $y = Kx + e$ with $\|e\|_2 \leq n^{-0.002} \cdot \|Kx\|_2$ requires almost quadratic time, assuming SETH.*

**Remark 7.1.** Note that if the vector $x$ is entry-wise non-negative, then computing an approximation $y$ to $Kx$ satisfying $y = Kx + e$ with $\|e\|_2 \leq n^{-0.002} \cdot \|Kx\|_2$ is possible in subquadratic in $n$ time, even if $K \in \mathbb{R}^{n \times n+1}$ has the form specified in Theorem 2.6. This is because we can first compute an approximation to $K'x'$, where $K'$ is $K$ with it's last column removed and $x'$ is $x$ with it's last coordinate removed, using our algorithm from Theorem 2.1 (which takes subquadratic in $n$ time). Then we can simply add the vector which is the last column of $K$ times the last coordinate of $x$ to the output exactly in $O(n)$ time. Because of non-negativity, the total error can be bounded by $\leq n^{-0.002} \cdot \|K'x'\|_2$ (coming from Theorem 2.1), which is $\text{len}^{-0.002} \cdot \|Kx\|_2$.

To prove the theorem, we use reduction from the Orthogonal Vectors (OV) problem, showing that the OV problem can be solved if such a hypothetical algorithm of Theorem 2.6 exists. First, the following lemma helps transform the input of OV (Definition 3.1) into a more convenient form.

**Proposition 7.1.** *Let $(A, B)$ be the input to the Orthogonal Vectors problem. By increasing the dimension $d$ by a constant multiplicative factor, we can ensure the following:*

- *$\forall i, j \in [n], \langle a_i, a_j \rangle \neq 0$ and $\langle b_i, b_j \rangle \neq 0$; that is, none of the pairs within $A$ and within $B$ are orthogonal.*
- *$\forall i, j \in [n], \|a_i\|_1 = \|b_j\|_1$, that is, all vectors have the same number of 1's.*

*Proof.* We pad the vectors with $2d+2$ extra coordinates as follows: for the first extra coordinate, all vectors in $A$ will get a 1 and all vectors in $B$ will get a 0. Similarly in the second extra coordinate, all vectors in $B$ will get a 1 and all vectors in $A$ will get a 0. For the next block of $d$ coordinates, all vectors in $B$ get all zeros. For vectors in $a \in A$, we give them $d - \|a\|_1$ many 1's (say consecutive), and the rest zeros (where we use the original number of 1s). We do the symmetric operation on the second block of $d$ coordinates for vectors in $B$. □

We now show that a sub-quadratic algorithm for computing matrix vector products for arbitrary vectors allows us to solve OV.

*Proof of Theorem 2.6.* Let $X = A \cup -B$, with the transformation specified in Proposition 7.1 applied to $A$ and $B$. First, we can assume without loss of generality that there are at most $n^{0.001}$ pairs $a \in A, b \in B$ that satisfy $\langle a, b\rangle = 0$, since otherwise by random sampling, we can find such a pair in $n^{1.99999} = o(n^2)$ time with high probability.

Let $t$ be the number of $\pm 1$ coordinates in the vectors in $A \cup -B$ (this quantity is the same for all vectors in $X$ due to Proposition 7.1). We now design the kernel matrix $K$ in a similar fashion as in the proof of Theorem 2.7 by considering the Gaussian kernel $k(x, y) = e^{-C\log(n)\|x-y\|_2^2}$ for a sufficiently large constant $C \geq 1$. We have:

1. For $a \in A$ and $b \in B$, if $\langle a, b\rangle = 0$, then the corresponding entry in $K$ has value (note the plus)
$$e^{-C\log(n)\|a+b\|_2^2} = e^{-2C\log(n)t} = n^{-2Ct}.$$
2. On the other hand, if $\langle a, b\rangle \geq 1$, the corresponding entry in $K$ has value
$$e^{-C\log(n)\|a+b\|_2^2} \leq e^{-C\log(n)(2t+2)} \leq n^{-(2t+2)C}.$$
3. Furthermore, the $n+1$th column of $K$ is a constant vector since $\|x_i - 0\|_2^2 = \|x_i\|_2^2 = \|x\|_1$ is the same value for all $x_i \in X$ due to Proposition 7.1. Let $\Delta = e^{-C\log(n)t}$ be the constant value in the $n+1$th column.

Consider the vector
$$x = (1, \cdots, 1, -1/\Delta) \in \mathbb{R}^{n+1}. \tag{11}$$
For $i \in [n]$, the $i$th entry of $Kx$ has value
$$\sum_{j=1}^{n} e^{-C\log(n)\|x_i - x_j\|_2^2} + \Delta \cdot \frac{-1}{\Delta} = \sum_{j=1, j\neq i}^{n} e^{-C\log(n)\|x_i - x_j\|_2^2}.$$
In other words, the $i$th entry of $Kx$ has value equal to the sum of the $i$th row of $K$, ignoring the diagonal (note: removing the diagonal is the main hurdle and the extra coordinate allows us to do this).

Now in the case that an orthogonal pair exists, item (1) above implies that some entry of $Kx$ has value at least $n^{-2Ct}$. In the case where no such pair exists, item (2) implies that every entry of $Kx$ is bounded by $n^{-(2t+1)C}$. Furthermore, by our assumption that there are at most $n^{0.001}$ orthogonal pairs in the yes case, we know that in *both cases*, we can upper bound $\|Kx\|_2^2$ by
$$\|Kx\|_2^2 \leq n^{0.001} \cdot (n^{0.001} n^{-2Ct})^2 + n \cdot (n^{-(2t+2)C})^2 \leq O(n^{-4Ct+0.003}).$$
This implies that *every* entry of $e$ is upper bounded by $\varepsilon\|Kx\|_2 = O(\varepsilon \cdot n^{-2Ct+0.0015})$. Thus if $\varepsilon \leq n^{-0.0015}/C'$ for a sufficiently large constant $C' \geq 1$, then in the yes case, some coordinate of $Kx+e$ has value at least $n^{-2Ct}/100$. Conversely in the no case, all coordinates of $Kx+e$ are smaller than $n^{-(2t+1)C} + \varepsilon\|Kx\|_2 \leq n^{-2Ct}/1000$. Thus, computing $y = Kx + e$ satisfying $\|e\|_2 \leq \varepsilon\|Kx\|_2$ allows us to solve OV, and the conclusion follows. □

### 7.1 Extension to Vector-Matrix-Vector Products

We extend Theorem 2.6 to the case of computing $y^\top Kx$ for a $K \in \mathbb{R}^{n \times n+1}$ Gaussian kernel matrix, where $K$ is only asymmetric by one column.

**Theorem 7.2.** *Let $X = \{x_1, \cdots, x_n\} \subset \mathbb{R}^d$ be input points with $d = \omega(\log n)$. Define $K \in \mathbb{R}^{n \times n+1}$ to be the Gaussian kernel matrix where the rows are indexed by $X$ and the columns are indexed by $X \cup \{0\}$. Any algorithm which takes as input $X$, vectors $x \in \mathbb{R}^{n+1}$ and $y \in \mathbb{R}^n$ such that the first $n$ coordinates of $x$ and $y$ agree, and returns a $n^{O(1)}$ multiplicative approximation to $y^\top Kx$ requires almost quadratic time, assuming SETH.*

**Remark 7.2.** The same remark as in Remark 7.1 holds: if $y$ and $x$ are entry-wise non-negative, we can compute even a constant factor multiplicative approximation to $y^\top Kx$ in subquadratic in $n$ time.

*Proof.* We use a similar reduction from OV as in the proof of Theorem 2.6. Recall from the proof of Theorem 2.6 the definition of vector $x$ (Equation (11)) and the construction of the kernel matrix $K \in \mathbb{R}^{n \times n+1}$: Given the input OV instance $A, B$, we let $X = A \cup -B$ and define the kernel matrix $K$ based on the Gaussian kernel $k(x, y) = e^{-C \log(n) \|x-y\|_2^2}$ for a sufficiently large constant $C \geq 1$.

From the proof of Theorem 2.6, we know that the $i$th entry of $Kx$ is equal to the sum of the $i$th row of $K$, ignoring the diagonal. Thus, $y^\top Kx$ denotes the sum of the off diagonal entries of $K$. That is,

$$y^\top Kx = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} e^{-C \log(n) \|x_i - x_j\|_2^2}.$$

Now continuing as in the proof of Theorem 2.6, item (1) there implies that some entry of $Kx$ has value at least $n^{-2Ct}$ if an orthogonal pair exists, where $t$ is the (fixed) number of 1 coordinates in $A \cup B$. In the case where no such pair exists, item (2) there implies that every entry of $Kx$ is bounded by $n^{-(2t+1)C}$. In other words, if an orthogonal pair exists, then $y^\top Kx$ is at least $n^{-2Ct}$, where as if no such pair exists, $y^\top Kx \leq n^{-(2t+1)C+1} = n^{-2tC} \cdot n^{-C+1} \ll n^{-2Ct}$ by picking a large constant $C \geq 1$. Thus, any polynomial approximation to $y^\top Kx$ allows us to solve OV, proving the lower bound. □

## 8 Lower Bounds for Kernel Sums

In this section, we prove lower bounds for calculating arbitrary vector-matrix-vector products, as well as computing on *asymmetric* kernel matrices, where the rows and columns need not be indexed by the same point set.

**Theorem 2.7.** *For any $d = \omega(\log n)$, computing $v^\top Kw$ for non-negative $v, w$ for the Gaussian kernel matrix $K \in \mathbb{R}^{n \times n}$ up to polynomial relative error requires almost quadratic time, assuming SETH.*

*Proof.* Let $A, B$ be the input to OV. Note that without loss of generality, we can assume every vector in $A$ and $B$ have the same number of 1's (say $t$). This is because we can pad $2d$ extra entries to the vectors, set the first $d$ new coordinates to all zeros for all vectors $A$ and the second set of $d$ to all zeros for all vectors in $B$. Then for vectors in $A$, we set the appropriate number of the second set of $d$ coordinates to 1's and vice-versa for $B$.

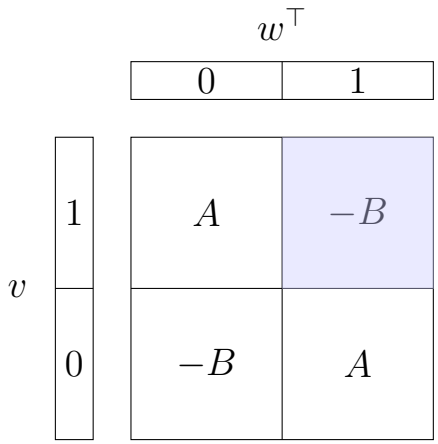


Figure 3: Representation of $v^\top K w$.

Now let $X = A \cup -B$ be the underlying dataset for the kernel matrix $K$ with kernel function $k(x, y) = e^{-C \log(n) \|x-y\|_2^2}$, let $v$ be the vector with all 1's in the first half and zeros in the second half, and $w$ be the vector with all 1's in the second half and zeros in the first half.

For $a \in A$ and $b \in B$, if $\langle a, b \rangle = 0$, then the corresponding entry in $K$ has value

$$e^{-C \log(n) \|a+b\|_2^2} = e^{-2C \log(n) t}.$$

On the other hand, if $\langle a, b \rangle \geq 1$, the corresponding entry in $K$ has value

$$e^{-C \log(n) \|a+b\|_2^2} \leq e^{-C \log(n)(2t+2)}.$$

Thus, by our choice of $v$ and $w$ (see Figure 3), we have that $v^\top K w$ is exactly the sum of all the entries

$$S := \sum_{a \in A, b \in B} e^{-C \log(n) \|a+b\|_2^2} = e^{-C \log(n) \cdot 2t} \sum_{a \in A, b \in B} e^{-2C \log(n) \langle a, b \rangle}.$$

If there is no orthogonal pair, then this sum is at most $e^{-C \log(n) \cdot (2t+2)} \cdot n^2$ and if there is an orthogonal pair, then this sum is at least $e^{-C \log(n) \cdot (2t+2)} \cdot (n^2 - 1) + e^{-2C \log(n) t}$. The ratio of these values is at least

$$\frac{e^{-C \log(n) \cdot (2t+2)} \cdot (n^2 - 1) + e^{-2C \log(n) t}}{e^{-C \log(n) \cdot (2t+2)} \cdot n^2} = \frac{n^{-2C+2} - n^{-2C} + 1}{n^{-2C+2}} \geq \Omega(n^{2C-2}).$$

By setting $C$ to be a sufficiently large constant, we have that any $o(n^{2C-2})$ approximation to $v^\top K w$ decides if we have an orthogonal pair in $A, B$, solving the OV problem.

□

Ideas of the above lower bound can be extended to show hardness for computations on *asymmetric* kernel matrices. Note that in the introduction, we defined a kernel matrix to be symmetric where the rows and columns are indexed by the same point set $X$. In practice, asymmetric kernel matrices are also popular, where an asymmetric kernel matrix $K \in \mathbb{R}^{n \times n}$ has its rows and columns indexed by possibly different point sets $X$ and $Y$: $K_{i,j} = k(x_i, y_j)$.

**Corollary 8.1.** *For any $d = \omega(\log n)$, computing $\mathbf{1}^\top K \mathbf{1}$ for an asymmetric kernel matrix $K \in \mathbb{R}^{n \times n}$ up to polynomial relative error requires almost quadratic time, assuming SETH.*

*Proof.* The proof uses the same construction as in Theorem 2.7: let $A, B$ be the input to OV and consider the 'asymmetric' kernel matrix $K_{A,-B}$. If we let the kernel function be $k(x, y) = e^{-C\log(n)\|x-y\|_2^2}$ then the same calculation as in Theorem 2.7 implies that the sum of the entries of $K$ is

$$\sum_{a\in A, b\in B} e^{-C\log(n)\|a+b\|_2^2} = e^{-C\log(n)\cdot 2t} \sum_{a\in A, b\in B} e^{-2C\log(n)\langle a,b\rangle}.$$

Thus to repeat the argument, if there is no orthogonal pair, then this sum is at most $e^{-C\log(n)\cdot(2t+2)} \cdot n^2$ and if there is an orthogonal pair, then this sum is at least $e^{-C\log(n)\cdot(2t+2)} \cdot (n^2-1) + e^{-2C\log(n)t}$. The ratio of these values is at least $\Omega(n^{2C-2})$ as before, and by setting $C$ to be a sufficiently large constant, we have that any $o(n^{2C-2})$ approximation to $\mathbf{1}^\top K \mathbf{1}$ decides if we have an orthogonal pair in $A, B$, solving the OV problem. □

Theorem 2.7 also implies that approximating $\lambda_1$ for an asymmetric kernel matrix requires almost quadratic time, assuming SETH.

**Corollary 8.2.** *For any $d = \omega(\log n)$, computing the top singular value of $K$ for an asymmetric kernel matrix $K$ up to polynomial relative error requires almost quadratic time, assuming SETH.*

*Proof.* We use the same construction as in Theorem 2.7. Note that the frobenius norm squared of the matrix $K$ is

$$\sum_{a\in A, b\in B} e^{-2C\log(n)\|a+b\|_2^2} = e^{-2C\log(n)\cdot 2t} \sum_{a\in A, b\in B} e^{-4C\log(n)\langle a,b\rangle}.$$

If there is no orthogonal pair, then the sum is at most $n^2 e^{-4C\log(n)\cdot(t+1)}$, meaning the top singular value is at most $n \cdot e^{-2C\log(n)\cdot(t+1)} \leq n^{-2Ct-2C+1}$. On the other hand, an orthogonal pair implies an entry with value at least $e^{-2C\log(n)\cdot t} = n^{-2Ct}$, which lower bounds the top singular value, e.g. by considering the basis vector corresponding to the column that this entry is in. Thus, any algorithm outputting a fixed polynomial factor approximation solves OV by appropriately increasing the value of the constant $C$. □

**Corollary 8.3.** *For any $d = \omega(\log n)$, computing $K\mathbf{1}$ up to polynomial relative error requires almost quadratic time, assuming SETH.*

*Proof.* Let $v$ be the computed value of $K\mathbf{1}$, which we suppose for the sake of contradiction can be expressed as $v = K\mathbf{1} + n^\alpha \|K\mathbf{1}\| u$ for some non-negative unit vector $u$. The proof uses the same construction as in Theorem 2.7. Note that the $i$th entry of $K\mathbf{1}$ is

$$\sum_{b\in B} e^{-2C\log(n)\|a_i+b\|_2^2} = e^{-2C\log(n)2t} \sum_{b\in B} e^{-4C\log(n)\langle a_i,b\rangle}.$$

If there are no orthogonal pairs, then

$$\|K\mathbf{1}\| \leq \sqrt{n} e^{-2C\log(n)(2t+2)}$$

so

$$\|v\| \leq (1+n^\alpha)\|K\mathbf{1}\| \leq e^{-2C\log(n)(2t+2)+(\alpha+\frac{1}{2})\log(n)+1}.$$

On the other hand, if there is an orthogonal pair (say $a_i$ and $b_j$), then

$$(K\mathbf{1})_i = e^{-2C\log(n)2t} \sum_{b \in B} e^{-4C\log(n)\langle a_i, b\rangle} \geq e^{-2C\log(n)2t}$$

so

$$\|v\| \geq e^{-2C\log(n)2t}.$$

When $C$ is large enough, this allows one to distinguish between the two cases. □

# 9 Conclusion and Open Questions

Our work makes significant progress on a number of natural algorithmic problems on kernel matrices. We believe the following are the most interesting questions left open by our work.

**Question 9.1.** *What is the smallest $\varepsilon \in (0,1)$ such that $s(K)$ can be computed up to relative error $1 \pm \varepsilon$ in subquadratic time? In particular, is $\varepsilon < \frac{1}{n}$ possible in subquadratic time?*

We conjecture the answer to the above question to be no. Note that it is possible to prove SETH based lower bounds for extremely high precision (computing $s(K)$ up to $1+e^{-\omega(\log n)}$ error) [BIS17]), but it seems new lower bound assumptions maybe needed to prove hardness for "moderately large" $\varepsilon$.

The second question is a natural follow up question to our non-negative matrix vector product upper bound of Theorem 2.1.

**Question 9.2.** *Can we prove a quadratic time hardness for computing a matrix vector product $Kx$ for a general $x$ (i.e. not necessarily non-negative) with guarantees similar to Theorem 2.1?*

Again, we conjecture the answer is no based on the hardness of our intermediate problem given in Theorem 2.6. A related question concerns approximating the eigenvalues of $K$ beyond the top eigenvector. Algorithmically, a bottleneck in obtaining an upper bound by using something subspace iteration or Arnoldi iteration is those algorithms require general matrix-vector queries. This was also an open question given by [BIMW21].

**Question 9.3.** *What is the complexity of computing $\lambda_2(K)$? In particular, can we compute a factor of 2 approximation of $\lambda_2(K)$ in sub-quadratic time?*

What specifically makes this problem difficult is that we do not have control over the inner product between the error vector of computing matrix-vector products and the second eigenvector. Our Theorem 2.2 strongly used that both the error vector and top eigenvector had a positive dot product; Example 2 showed that the method fails without this assumption. Our last question concerns the effectiveness of the power method for just the top eigenvalue.

**Question 9.4.** *How many $\delta$-approximate non-negative matrix-vector queries are required to estimate $\lambda_1(K)$ up to $\varepsilon$ relative error?*

Our lower bound Proposition 5.3 only applies to the sequence of queries made by noisy power iteration, and only says something non-trivial when $(\delta/\varepsilon)\sqrt{n} \gg 1$. What about other queries? In particular, the two matrices described in Proposition 5.3 that are not distinguished by the power method *are* distinguished by the single query $K\mathbf{1} + (\text{noise}) - \mathbf{1}$. When $\delta = 0$, it's clear one can use Krylov subspace / Chebyshev polynomial ideas to reach $\widetilde{O}(1/\sqrt{\varepsilon})$ iterations. Can one obtain suitable control over the error when $\delta > 0$ to extrapolate this bound?

## References


[ABB+23] Andreas Anastasiou, Alessandro Barp, François-Xavier Briol, Bruno Ebner, Robert E Gaunt, Fatemeh Ghaderinezhad, Jackson Gorham, Arthur Gretton, Christophe Ley, Qiang Liu, et al. Stein's method meets computational statistics: A review of some recent developments. *Statistical Science*, 38(1):120–139, 2023.

[ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for lcs and other sequence similarity measures. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 59–78. IEEE, 2015.

[ACSS20] Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 541–552. IEEE, 2020.

[AS23] Josh Alman and Zhao Song. Fast attention requires bounded entries. *Advances in Neural Information Processing Systems*, 36:63117–63135, 2023.

[AWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I 41*, pages 39–51. Springer, 2014.

[BCIS18] Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. Efficient density evaluation for smooth kernels. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 615–626. IEEE, 2018.

[BCJ20] Ainesh Bakshi, Nadiia Chepurko, and Rajesh Jayaram. Testing positive semi-definiteness via random submatrices. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1191–1202. IEEE, 2020.

[BDD+24] Rajarshi Bhattacharjee, Gregory Dexter, Petros Drineas, Cameron Musco, and Archan Ray. Sublinear time eigenvalue approximation via random sampling. *Algorithmica*, 86(6):1764–1829, 2024.

[BIK+23] Ainesh Bakshi, Piotr Indyk, Praneeth Kacham, Sandeep Silwal, and Samson Zhou. Subquadratic algorithms for kernel matrices via kernel density estimation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[BIMW21] Arturs Backurs, Piotr Indyk, Cameron Musco, and Tal Wagner. Faster kernel matrix algebra via density estimation. In *International Conference on Machine Learning*, pages 500–510. PMLR, 2021.

[BIS17] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[BIW19] Arturs Backurs, Piotr Indyk, and Tal Wagner. Space and time efficient kernel density estimation in high dimensions. *Advances in neural information processing systems*, 32, 2019.

[CKNS20] Moses Charikar, Michael Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 172–183. IEEE, 2020.

[CKW24] Moses Charikar, Michael Kapralov, and Erik Waingarten. A quasi-monte carlo data structure for smooth kernel evaluations. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5118–5144. SIAM, 2024.

[CNW16] Michael B. Cohen, Jelani Nelson, and David P. Woodruff. Optimal approximate matrix product in terms of stable rank. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[CS17] Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1032–1043. IEEE, 2017.

[CSTEK01] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

[HJK+24] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. In *The Twelfth International Conference on Learning Representations*, 2024.

[HSS08] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171 – 1220, 2008.

[IKSW25] Piotr Indyk, Michael Kapralov, Kshiteej Sheth, and Tal Wagner. Improved algorithms for kernel matrix-vector multiplication under sparsity assumptions. In *The Thirteenth International Conference on Learning Representations*, 2025.

[M+11] Michael W Mahoney et al. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[MM15] Cameron Musco and Christopher Musco. Randomized block krylov methods for stronger and faster approximate singular value decomposition. *Advances in neural information processing systems*, 28, 2015.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[RST10] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2010.

[RWZ20] Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 26:1–26:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[SW23] William Swartworth and David P Woodruff. Optimal eigenvalue approximation via sketching. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 145–155, 2023.

[SW25] William Swartworth and David P Woodruff. Tight sampling bounds for eigenvalue approximation. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 489–516. SIAM, 2025.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[W+14] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[Wil05] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2-3):357–365, 2005.

[ZHDK23] Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. In *International Conference on Machine Learning*, pages 40605–40623. PMLR, 2023.