Sketching Meets Differential Privacy: Fast Algorithm for Dynamic Kronecker Projection Maintenance

Zhao Song [1] Xin Yang [2] Yuanyuan Yang [2] Lichen Zhang [3]

Abstract

Projection maintenance is one of the core data structure tasks. Efficient data structures for projection maintenance have led to recent breakthroughs in many convex programming algorithms. In this work, we further extend this framework to the Kronecker product structure. Given a constraint matrix A and a positive semi-definite matrix $W \in \mathbb{R}^{n \times n}$ with a sparse eigenbasis, we consider the task of maintaining the projection in the form of $\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B}$, where $\mathsf{B} = \mathsf{A}(W \otimes I)$ or $\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2})$. At each iteration, the weight matrix W receives a low rank change and we receive a new vector h. The goal is to maintain the projection matrix and answer the query $\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B}h$ with good approximation guarantees. We design a fast dynamic data structure for this task and it is robust against an adaptive adversary. Following the beautiful and pioneering work of [Beimel, Kaplan, Mansour, Nissim, Saranurak and Stemmer, STOC'22], we use tools from differential privacy to reduce the randomness required by the data structure and further improve the running time.

1. Introduction

Projection maintenance is one of the most important data structure problems in recent years. Many convex optimization algorithms that give the state-of-the-art running time heavily rely on an efficient and robust projection maintenance data structure (Cohen et al., 2019; Lee et al., 2019; Jiang et al., 2020b;a; Huang et al., 2022b). Let $B \in \mathbb{R}^{m \times n}$, consider the projection matrix $P = B(B^\top B)^{-1}B^\top$. The projection maintenance task aims for the design of a data structure with the following guarantees: it can preprocess and compute an initial projection. At each iteration, B receives a low rank or sparse change, and the data structure needs to update B to reflect these changes. It will then be asked to approximately compute the matrix-vector product, between the updated P and an online vector h. For example, in linear programming, one sets $B = \sqrt{W}A$, where $A \in \mathbb{R}^{m \times n}$ is the constraint matrix and W is the diagonal slack matrix. Each iteration, W receives relatively small perturbations. Then, the data structure needs to output an approximate vector to $\sqrt{W}A(A^\top W A)^{-1}A^\top\sqrt{W}h$, for an online vector $h \in \mathbb{R}^n$.

In this work, we consider a specific type of projection maintenance problem. Concretely, our matrix $B = \mathsf{A}(W \otimes I)$ or $B = \mathsf{A}(W^{1/2} \otimes W^{1/2})$, where $W \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix and $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ is a matrix whose i-th row is the vectorization of an $n \times n$ matrix. We call the problem for maintaining such kind of matrices, the *Dynamic Kronecker Product Projection Maintenance Problem*. Maintaining the Kronecker product projection matrix has important implications for solving semi-definite programs using interior point method (Jiang et al., 2020a; Huang et al., 2022b;a; Gu & Song, 2022). Specifically, one has m constraint matrices $A_1, \ldots, A_m \in \mathbb{R}^{n \times n}$, and A is constructed by vectorization each of the A_i's as its rows. The matrix W is typically the complementary slack of the program. In many cases, the constraint matrices $A_1, \ldots, A_m$ are simultaneously diagonalizable, meaning that there exists a matrix P such that $P^\top A_i P$ is diagonal. This leads to the matrix W also being simultaneously diagonalizable, which implies potential faster algorithms for this kind of SDP (Jiang & Li, 2016). We study this setting in a more abstract and general fashion: suppose $A_1, \ldots, A_m$ are fixed. The sequence of update matrices $W^{(0)}, W^{(1)}, \ldots, W^{(T)} \in \mathbb{R}^{n \times n}$ share the same eigenbasis.

The data structure design problem can be decomposed into 2 parts: 1). How to update the projection matrix fast, and 2). How to answer the query efficiently. For the update portion, we leverage the fact that the updates are relatively small. Hence, by updating the inverse part of projection in a lazy fashion, we can give a fast update algorithm.

[1] Adobe Research [2] The University of Washington [3] MIT. Correspondence to: Zhao Song <zsong@adobe.com>, Yuanyuan Yang <yyangh@cs.washington.edu>, Lichen Zhang <lichenz@mit.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

For the query portion, it is similar to the Online Matrix Vector Multiplication Problem (Henzinger et al., 2015; Larsen & Williams, 2017; Chakraborty et al., 2018), with a changing matrix-to-multiply. To speed up this process, prior works either use importance sampling to sparsify the vector $h^{(t)}$, or using sketching matrices to reduce the dimension. For the latter, the idea is to prepare multiple instances of sketching matrices beforehand, and batch-multiplying them with the initial projection matrix $P^{(0)}$. Let $\mathsf{R} \in \mathbb{R}^{n^2 \times Tb}$ denote the batched sketching matrices, where b is the sketching dimension for each matrix, one prepares the matrix $P\mathsf{R}$. At each query phase, one only needs to use one sketching matrix, $R^{(t)}$, and compute $(P(R^{(t)})^\top)R^{(t)}h$. By computing this product from right to left, we effectively reduce the running time from $O(n^4)$ to $O(n^2 b)$. One significant drawback of this approach is that, if the number of iterations T is large, then the preprocessing and update phase become less efficient due to the sheer number of sketching matrices.

We observe that the fundamental reason that applying one uniform sketch for all iterations will fail is due to the dependence between the current output and all previous inputs: When using the projection maintenance data structure in an iterative process, the new input query is typically formed by a combination of the previous outputs. This means that our data structure should be robust against an *adaptive adversary*. Such an adversary can infer the randomness from observing the output of the data structure and design new input to the data structure. Prior works combat this issue by using a uniformly-chosen sketching matrix that won't be used again.

To make our data structure both robust against an adaptive adversary and reduce the number of sketches to use, we adapt a differential privacy framework as in the fantastic work (Beimel et al., 2022). Given a data structure against an oblivious adversary that outputs a real number, the pioneering work (Beimel et al., 2022) proves that it is enough to use $\widetilde{O}(\sqrt{T})$ data structures instead of T for adaptive adversary, while the runtime is only blew up by a polylogarithmic factor. However, their result is not immediately useful for our applications, since we need an approximate vector with n^2 numbers. A direct consequence of strong composition gives rise to $\widetilde{O}(n\sqrt{T})$ data structures, which is only useful for $n < \sqrt{T}$ and less satisfactory for moderate choice of T. To circumvent this issue, we instead ask for a high-precision estimation to each entry of the vector. Using strong composition over all entries, we arrive at a data structure that only uses $\widetilde{O}(\sqrt{T})$ sketching matrices, and pays an extra $\widetilde{O}(k^{1.5})$ time per query, where k denotes the total number of coordinates-to-be-estimated. While one might want a tighter runtime by tightening the extra $\widetilde{O}(k^{1.5})$ to $\widetilde{O}(k)$, we conjecture our analytical framework is essentially tight due to the tightness of strong composition (Kairouz et al., 2015). If one wants to improve both the number of sketches and extra query time, it might be instrumental to develop faster algorithms for harder problem, such as efficient differentially private mean estimation.

An immediate consequence of our data structure is an improvement over the (diagonal) projection maintenance of (Cohen et al., 2019; Lee et al., 2019; Song & Yu, 2021). In this model, the matrix $B = \sqrt{W}A \in \mathbb{R}^{n\times n}$ where $W \in \mathbb{R}^{n\times n}$ is a non-negative diagonal matrix. Prior works use T independent sketching matrices with sketching dimension being $\widetilde{O}(\sqrt{n})$. Hence, the query time is $\widetilde{O}(n^{1.5})$, matches the differential-privacy-incurred extra $\widetilde{O}(n^{1.5})$ factor. Our preprocessing and update time is faster, since we only need $\widetilde{O}(\sqrt{T})$ sketches instead of T.

We start with defining notations to simplify further discussions.

Definition 1.1 (Time complexity for preprocessing, update and query)**.** Let $B^{(0)}, B^{(1)}, \ldots, B^{(T)} \in \mathbb{R}^{m\times n}$ be an online sequence of matrices and $h^{(1)}, \ldots, h^{(T)} \in \mathbb{R}^n$ be an online sequence of vectors.

- We define $\mathcal{T}_{\mathrm{prep}}(m, n, s, b)$ as the preprocessing time of a dynamic data structure with input matrix $B \in \mathbb{R}^{m\times n}$, sketching dimension b and the number of sketches s.
- We define $\mathcal{T}_{\mathrm{update}}(m, n, s, b, \varepsilon)$ as the update time of a dynamic data structure with update matrix $B^{(t)} \in \mathbb{R}^{m\times n}$, sketching dimension b and the number of sketches s. This operation should update the projection to be $(1 \pm \varepsilon)$ spectral approximation.
- We define $\mathcal{T}_{\mathrm{query}}(m, n, b, \varepsilon, \delta)$ as the query time of a dynamic data structure with query vector $h \in \mathbb{R}^n$, sketching dimension b and the number of sketches s. This operation should output a vector $\widetilde{h}^{(t)} \in \mathbb{R}^n$ such that for any $i \in [n]$, $|\widetilde{h}_i^{(t)} - (P^{(t)}h^{(t)})_i| \leq \varepsilon \|h^{(t)}\|_2$ with probability at least $1 - \delta$.

Theorem 1.2. *Let $B^{(0)}, B^{(1)}, \ldots, B^{(T)} \in \mathbb{R}^{m\times n}$ be an online sequence of matrices and $h^{(1)}, \ldots, h^{(T)} \in \mathbb{R}^n$ be an online sequence of vectors.*

There exists a dynamic data structure for the projection maintenance task, with

- *Preprocessing time $\mathcal{T}_{\mathrm{prep}}(m, n, \sqrt{T}, b)$.*
- *Update time $\mathcal{T}_{\mathrm{update}}(m, n, \sqrt{T}, b, \varepsilon)$.*
- *Query time $\mathcal{T}_{\mathrm{query}}(m, n, b, \varepsilon, \delta) + n^{1.5}$.*

Moreover, the data structure is robust against an adaptive adversary.

For the sake of comparison, we note the data structure in (Cohen et al., 2019; Lee et al., 2019; Song & Yu, 2021) has preprocessing time $\mathcal{T}_{\mathrm{prep}}(m, n, T, b)$, update time $\mathcal{T}_{\mathrm{update}}(m, n, T, b, \varepsilon)$ and query time $\mathcal{T}_{\mathrm{query}}(m, n, b, \varepsilon, \delta)$. In the linear program applications, they choose $b = \sqrt{n}$, yielding a query time of $\widetilde{O}(n^{1.5})$, which matches our query time.

Now, we turn to the harder problem of maintaining a Kronecker product projection matrix. While the update matrices are positive semi-definite instead of diagonal, we note that they still share the same eigenbasis. Essentially, the updates are a sequence of diagonal eigenvalues, and we can leverage techniques from prior works (Cohen et al., 2019; Lee et al., 2019; Song & Yu, 2021) with lazy updates.

Our data structure also relies on using sketches to speed up the query phase. We present an informal version of our result below.

Theorem 1.3. *Let* $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ *and* $\mathsf{B} = \mathsf{A}(W \otimes I)$ *or* $\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2})$. *Let* $\mathsf{R} \in \mathbb{R}^{sb \times n^2}$ *be a batch of* s *sketching matrices, each with dimension* b. *Given a sequence of online matrices* $W^{(1)}, \dots, W^{(T)} \in \mathbb{R}^{n \times n}$ *where* $W^{(t)} = U \Lambda^{(t)} U^\top$ *and online vectors* $h^{(1)}, \dots, h^{(T)} \in \mathbb{R}^{n^2}$. *The data structure has the following operations:*

- INIT*: The data structure preprocesses and generates an initial projection matrix in time*

$$mn^\omega + m^\omega + \mathcal{T}_{\mathrm{mat}}(m, m, n^2) + \mathcal{T}_{\mathrm{mat}}(n^2, n^2, sb).$$

- UPDATE(W)*: the data structure updates and maintains a projection* $\widetilde{P}$ *such that*

$$(1-\varepsilon) \cdot P \preceq \widetilde{P} \preceq (1+\varepsilon) \cdot P,$$

where P *is the projection matrix updated by* W. *Moreover, if*

$$\sum_{i=1}^{n} (\ln \lambda_i(W) - \ln \lambda_i(W^{\mathrm{old}}))^2 \le C^2,$$

the expected time per call of UPDATE(W) *is*

$$C/\varepsilon^2 \cdot (\max\{n^{f(a,c)+\omega-2.5} + n^{f(a,c)-a/2}, \\ n^{f(a,c)+\omega-4.5} sb + n^{f(a,c)-2-a/2} sb\}).$$

- QUERY(h)*: the data structure outputs* $\widetilde{P} R_l^\top R_l \cdot h$. *If* $\mathrm{nnz}(U) = O(n^{1.5+a/2})$, *then it takes time*

$$n^{3+a} + n^{2+b}.$$

Here, $a \in (0,1)$ *is a parameter that can be chosen and* $f(a,c) \in [4,5)$ *is a function defined as in Def. B.14.*

Combining with our differential privacy framework, it is sufficient to set $s = \widetilde{O}(\sqrt{T})$ instead of T to handle adaptive adversary. The extra $\widetilde{O}(n^3)$ cost for query is subsumed by the original query cost $\widetilde{O}(n^{3+a})$. For the sake of discussion, let us consider the parameter setting

$$m = n^2, b = \widetilde{O}(\varepsilon^{-2}), \omega = 2, f(a,c) = 4,$$

and $C/\varepsilon^2 = O(1)$. Then the running time simplifies to

- Preprocessing in $m^\omega + \mathcal{T}_{\mathrm{mat}}(m, m, \sqrt{T})$ time.
- Update in $m^{0.75}\sqrt{T} + m^{1-a/4}\sqrt{T}$ time.
- Query in $m^{1.5+a/2}$ time.

This again improves the standard approach that uses T sketching matrices. We want to point out that while reducing the number of sketches from T to $\widetilde{O}(\sqrt{T})$ does not directly improve the running time for linear programming and semidefinite programming (due to other operations subsuming the cost), our result provides an interesting perspective on designing adaptive robust data structures. Particularly, for optimization problems that require many iterations to converge (e.g, the solver for sum-of-squares optimization (Jiang et al., 2022)), our data structure provides substantial speedups.

Roadmap. In Section 2, we present our related work on sketching, differential privacy and projection maintenance. In Section 3, we provide the basic notations of this paper, preliminaries on Kronecker product calculation, and the formulation of the data structure design problem. In Section 4, we provide an overview on techniques we used in this paper. In Section 5, we provide the description of our algorithm , running time analysis and proof of correctness of Kronecker Product Maintenance Data Structure. In Section 6, we present the definition of the set query problem, the set query estimation data structure and its analysis of correctness and approximation guarantee.

2. Related Work

Sketching. Sketching is a fundamental tool and has many applications in machine learning and beyond, such as linear regression, low-rank approximation (Clarkson & Woodruff, 2013; Nelson & Nguyên, 2013; Meng & Mahoney, 2013; Boutsidis & Woodruff, 2014; Razenshteyn et al., 2016; Song et al., 2017; Andoni et al., 2018; Song et al., 2019), distributed problems (Woodruff & Zhong, 2016; Boutsidis et al., 2016), reinforcement learning (Wang et al., 2020), projected gradient descent (Xu et al., 2021), tensor decomposition (Song et al., 2019), clustering (Esfandiari et al., 2021), signal interpolation (Song et al., 2022a), distance oracles (Deng et al., 2022b), generative adversarial networks (Xiao et al., 2018), training neural networks (Lee et al., 2020; Brand et al., 2021; Song et al., 2021a;b; Hu et al., 2022), matrix completion (Gu et al., 2023), matrix sensing (Deng et al., 2023b; Qin et al., 2023b), attention scheme inspired regression (Li et al., 2023b; Deng et al., 2023a; Li et al., 2023a; Gao et al., 2023c;a), sparsification of attention matrix (Deng et al., 2023c), discrepancy minimization (Deng et al., 2022a), dynamic tensor regression problem (Reddy et al., 2022), John Ellipsoid computation (Song et al., 2022b), NLP tasks (Liang et al., 2020), total least square regression (Diao et al., 2019).

Differential Privacy. First introduced in (Dwork et al., 2006), differential privacy has been playing an important role in providing theoretical privacy guarantees for enormous algorithms (Dwork & Roth, 2014), for example, robust learning a mixture of Gaussians (Kamath et al., 2019), hypothesis selection (Bun et al., 2021), hyperparameter selection (Mohapatra et al., 2022), convex optimization (Kamath et al., 2022), first-order method (Subramani et al., 2021) and mean estimation (Kamath et al., 2020; Hopkins et al., 2022). Techniques from differential privacy are also widely studied and applied in machine learning (Chaudhuri & Monteleoni, 2008; Williams & McSherry, 2010; Jayaraman & Evans, 2019; Triastcyn & Faltings, 2020), deep neural networks (Abadi et al., 2016; Bagdasaryan et al., 2019), computer vision (Zhu et al., 2020; Luo et al., 2021; Torkzadehmahani et al., 2019), natural language processing (Yue et al., 2021; Weggenmann & Kerschbaum, 2018), large language models (Gao et al., 2023b; Yu et al., 2022), label protection in split learning (Yang et al., 2022), multiple data release (Wu et al., 2022), federated learning (Sun et al., 2022) and peer review (Ding et al., 2022). Recent works also show that robust statistical estimator implies differential privacy (Hopkins et al., 2023).

Projection Maintenance Data Structure. The design of efficient projection maintenance data structure is a core step that lies in many optimization problems, such as linear programming (Vaidya, 1989b; Cohen et al., 2019; Song, 2019; Lee et al., 2019; Brand, 2020; Song & Yu, 2021; Jiang et al., 2021; Brand et al., 2020; Ye, 2020; Dong et al., 2021; Gu & Song, 2022), cutting plane method (Vaidya, 1989a; Jiang et al., 2020b), integral minimization (Jiang et al., 2023a), empirical risk minimization (Lee et al., 2019; Qin et al., 2023a), semidefinite programming (Jiang et al., 2020b;a; Huang et al., 2022b;a; Gu & Song, 2022), dynamic least-square regression (Jiang et al., 2023b), large language models (Brand et al., 2023), and sum-of-squares optimization (Jiang et al., 2022).

3. Preliminaries & Problem Formulation

In Section 3.1, we introduce the basic notations that we will use in the remainder of the paper. In Section 3.2, we describe the data structure design problem of our paper.

3.1. Notations and Basic Definitions.

For any integer $n > 0$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. Let $\Pr[\cdot]$ denote probability and $\mathbb{E}[\cdot]$ denote expectation. We use $\|x\|_2$ to denote the ℓ_2 norm of a vector x. We use $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean μ and variance σ^2. We use $\widetilde{O}(f(n))$ to denote $O(f(n) \cdot \mathrm{poly}\log(f(n))$. We use $\mathcal{T}_{\mathrm{mat}}(m, n, k)$ to denote the time for matrix multiplication for matrix with dimension $m \times n$ and matrix with dimension $n \times k$. We denote $\omega \approx 2.38$ as the current matrix multiplication exponent, i.e., $\mathcal{T}_{\mathrm{mat}}(n, n, n) = n^\omega$. We denote $\alpha \approx 0.31$ as the dual exponent of matrix multiplication.

We use $\|A\|$ and $\|A\|_F$ to denote the spectral norm and the Frobenius norm of matrix A, respectively. We use $A^\top$ to denote the transpose of matrix A. We use I_m to denote the identity matrix of size $m \times m$. For α being a vector or matrix, we use $\|\alpha\|_0$ to denote the number of nonzero entries of α. Given a real square matrix A, we use $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ to denote its largest and smallest eigenvalue, respectively. Given a real matrix A, we use $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ to denote its largest and smallest singular value, respectively. We use A^{-1} to denote the matrix inverse for matrix A. For a square matrix A, we use $\mathrm{tr}[A]$ to denote the trace of A. We use b and n^b interchangeably to denote the sketching dimension, and $b \in [0, 1]$ when the sketching dimension is n^b.

Given an $n_1 \times d_1$ matrix A and an $n_2 \times d_2$ matrix B, we use $\otimes$ to denote their Kronecker product, i.e., $A \otimes B$ is a matrix where its $(i_1 + (i_2 - 1) \cdot n_1, j_1 + (j_2 - 1) \cdot d_1)$-th entry is $A_{i_1,j_1} \cdot B_{i_2,j_2}$. We denote $I_n \in \mathbb{R}^{n \times n}$ as the n dimensional identity matrix. For matrix $A \in \mathbb{R}^{n \times n}$, we denote (column vector) $\mathrm{vec}(A)$ as the vectorization of A. We use $\langle \cdot, \cdot \rangle$ to denote the inner product, when applied to two vectors, this denotes the standard dot product between two vectors, and when applied to two matrices, this means $\langle A, B \rangle = \mathrm{tr}[A^\top B]$, i.e., the trace of $A^\top B$.

3.2. Problem Formulation

In this section, we introduce our new online Kronecker projection matrix vector multiplication problem. Before that, we will review the standard online matrix vector multiplication problem:

Definition 3.1 (Online Matrix Vector Multiplication (OMV), (Henzinger et al., 2015; Larsen & Williams, 2017; Chakraborty et al., 2018))**.** Given a fixed matrix $A \in \mathbb{R}^{n \times n}$. The goal is to design a dynamic data structure that maintains matrix A and supports fast matrix-vector multiplication for $A \cdot h$ for any query h with the following operations:

- $\textsc{Init}(A \in \mathbb{R}^{n \times n})$: The data structure takes the matrix A as input, and does some preprocessing.
- $\textsc{Query}(h \in \mathbb{R}^n)$: The data structure receives a vector $h \in \mathbb{R}^n$, and the goal is to approximate the matrix vector product $A \cdot h$.

It is known that if we are given a list of h (e.g. $T = n$ different vectors $h^{(1)}, h^{(2)}, \cdots, h^{(T)}$) at the same time, this can be done in n^ω time. However, if we have to output the answer before we see the next one, it's unclear how to do it in truly sub-quadratic time per query.

Motivated by linear programming, a number of works (Cohen et al., 2019; Lee et al., 2019; Song & Yu, 2021; Jiang et al., 2021; Dong et al., 2021) have explicitly studied the following problem:

Definition 3.2 (Online Projection Matrix Vector Multiplication (OPMV), (Cohen et al., 2019))**.** Given a fixed matrix $A \in \mathbb{R}^{m \times n}$ and a diagonal matrix $W \in \mathbb{R}^{m \times m}$ with non-negative entries. The goal is to design a dynamic data structure that maintains $P(W) = \sqrt{W} A (A^\top W A)^{-1} A^\top \sqrt{W}$ and supports fast multiplication for $P(W) \cdot h$ for any future query h with the following operations:

- $\textsc{Init}(A \in \mathbb{R}^{m \times n}, W \in \mathbb{R}^{m \times m})$: The data structure takes the matrix A and the diagonal matrix W as input, and performs necessary preprocessing.
- $\textsc{Update}(W^{\mathrm{new}} \in \mathbb{R}^{m \times m})$: The data structure takes diagonal matrix W^{new} and updates W by $W + W^{\mathrm{new}}$.
- $\textsc{Query}(h \in \mathbb{R}^n)$: The data structure receives a vector $h \in \mathbb{R}^n$, and the goal is to approximate the matrix vector product $P(W) \cdot h$.

In this work, inspired by semidefinite programming (Huang et al., 2022b), we introduce the following novel data structure design problem:

Definition 3.3 (Online Kronecker Projection Matrix Vector Multiplication(OKPMV))**.** Suppose we have $A_1 \in \mathbb{R}^{n \times n}, \ldots, A_m \in \mathbb{R}^{n \times n}$, and let $\mathsf{A} = \begin{bmatrix} \mathrm{vec}(A_1) & \mathrm{vec}(A_2) & \cdots & \mathrm{vec}(A_m) \end{bmatrix}^\top \in \mathbb{R}^{m \times n^2}$. Let $W = U \Lambda U^\top \in \mathbb{R}^{n \times n}$ be positive semi-definite. Define $\mathsf{B} = \mathsf{A}(W \otimes I_n) \in \mathbb{R}^{m \times n^2}$ or $\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2}) \in \mathbb{R}^{m \times n^2}$. The goal is to design a dynamic data structure that maintains the projection $\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B}$ with the following procedures:

- $\textsc{Initialize}$: The data structure preprocesses B and forms $\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B}$.
- $\textsc{Update}$: The data structure receives a matrix $\Delta = U \widetilde{\Lambda} U^\top \in \mathbb{R}^{n \times n}$. The goal is to update the matrix B to $\mathsf{A}((W + \Delta) \otimes I_n)$ or $\mathsf{A}((W + \Delta)^{1/2} \otimes (W + \Delta)^{1/2})$ and the corresponding projection.
- $\textsc{Query}$: The data structure receives a vector $h \in \mathbb{R}^{n^2}$, and the goal is to approximate the matrix B and forms the matrix vector product $\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B} h$ quickly.

Remark 3.4. This problem can be viewed as a generalization to the data structure problem posed in (Cohen et al., 2019; Lee et al., 2019; Song & Yu, 2021). In their settings, the matrix B is not in the Kronecker product form and W is a full rank diagonal matrix.

4. Technical Overview

Our work consists of two relatively independent but robust results, and our final result is a combination of them both.

The first result considers designing an efficient projection maintenance data structure for Kronecker product in the form of $\mathsf{A}(W \otimes I)$ or $\mathsf{A}(W^{1/2} \otimes W^{1/2})$. Our main machinery consists of sketching and low rank update for amortization. More concretely, we explicitly maintain the quantity $\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B} \mathsf{R}^\top$, where R is a batch of sketching matrices.

By using fast rectangular matrix multiplication (Gall & Urrutia, 2018), we only update the projection maintenance when necessary and we update matrices related to the batch of sketching matrices. To implement the query, we pick one sketching matrix and compute their corresponding vectors $\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B} R^\top R h$. One of the main challenges in our data structure is to implement this sophisticated step with a Kronecker product-based projection matrix. We show that as long as $W = U \Lambda U^\top$ with only the diagonal matrix Λ changing, we can leverage matrix Woodbury identity and still implement this step relatively fast. For query, we note that a naive approach will be just multiplying the $n^2 \times n^2$ projection matrix with a vector, which will take $O(n^4)$ time. To break this quadruple barrier, however, is non-trivial. While using sketching seemingly speeds up the matrix-vector product, this is not enough: since we update the projection in a lazy fashion, during query we are required to "complete" the low rank update. Since W is positive semi-definite, the orthonormal eigenbasis might be dense, causing a dense $n^2 \times n^2$ multiplication with a n^2 vector. Hence, we require the eigenbasis $U \in \mathbb{R}^{n \times n}$ to be relatively sparse, i.e, $\mathrm{nnz}(U) = O(n^{1.5 + a/2})$ for $a \in (0, 1)$. Equivalently, we can seek for a simultaneously diagonalization using a sparse matrix. In this work, we keep this assumption, and leave removing it as a future direction.

The second main result uses techniques from differential privacy to develop robust data structures against an adaptive adversary. The intuition of such data structure is to protect the privacy of internal randomness (i.e., sketching matrices) from the adversary.

In the inspiring prior work due to (Beimel et al., 2022), they show a generic reduction algorithm that given a data structure that is robust against an oblivious adversary, outputs a data structure that is robust against an adaptive adversary. However, their mechanism has drawbacks — it requires the data structure to output a real number. Naively adapting their method to higher-dimensional output will lead to significantly more data structures and much slower update and query time. To better characterize the issue caused by high dimensional output, we design a generic data structure for the set query problem. In this problem, we are given a sequence of matrices $P^{(0)}, P^{(1)}, \dots, P^{(T)} \in \mathbb{R}^{n \times n}$ and a sequence of vectors $h^{(1)}, \dots, h^{(T)} \in \mathbb{R}^n$. At each iteration $t \in [T]$, we update $P^{(t-1)}$ to $P^{(t)}$ and we are given a set of indices $Q_t \subseteq [n]$ with support size $k \leq n$, and we only need to approximate entries in set Q_t. This model has important applications in estimating the heavy-hitter coordinates of a vector (Price, 2011; Jiang et al., 2020b). To speed up the matrix-vector product, we use batched sketching matrices. Our method departs from the standard approach that uses T sketches for handling adaptive adversary, by using only $\widetilde{O}(\sqrt{T})$ sketches. The key here is to use strong composition (Dwork & Roth, 2014) over T iterations, hence showing $\widetilde{O}(\sqrt{T})$ data structures are sufficient. For each coordinate, we use a differentially private median procedure together with Chernoff bound. However, note that we have to estimate k entries and we require them to succeed simultaneously. To address this issue, we compute a high-precision private median per entry, and again use strong composition over k entries. This incurs an extra $\sqrt{k}$ time for each entry, yielding an overall $k^{1.5}$ cost in addition to the standard query runtime. If one wants to estimate all entries, this gives rise to an extra additive $n^{1.5}$ time per query. For most applications, this extra term is either matched by (Cohen et al., 2019; Lee et al., 2019; Song & Yu, 2021) or subsumed by query time, as in our Kronecker product projection maintenance task.

5. Kronecker Product Projection Maintenance Data Structure

In this section, we provide the main theorem for Kronecker product projection maintenance together with the data structure.

Before proceeding, we introduce an amortization tool regarding matrix-matrix multiplication that helps us analyze the running time of certain operations:

Definition 5.1 ((Cohen et al., 2019)). Given $i \in [r]$, we define the weight function as

$$g_i = \begin{cases} n^{-a}, & \text{if } i < n^a; \\ i^{\frac{\omega-2}{1-a}-1} n^{-\frac{a(\omega-2)}{1-a}}, & \text{otherwise.} \end{cases}$$

Consider multiplying a matrix of size $n \times r$ with a matrix of size $r \times n$. If $r \leq n^a$, then multiplying these matrices takes $O(n^{2+o(1)})$ time, otherwise, it takes $O(n^2 + r^{\frac{\omega-2}{1-a}} n^{2-\frac{a(\omega-2)}{1-a}})$ time. Both of these quantities can be captured by $O(r g_r \cdot n^{2+o(1)})$.

Theorem 5.2 (Kronecker Product Projection Maintenance. Informal version of Theorem B.15)**.** *Given a collection of matrices $A_1, \cdots, A_m \in \mathbb{R}^{n \times n}$. We define $B_i = W^{1/2} A_i W^{1/2} \in \mathbb{R}^{n \times n}, \forall i \in [m]$[1]. We define $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ to be the matrix where i-th row is the vectorization of $A_i \in \mathbb{R}^{n \times n}$ and $\mathsf{B} \in \mathbb{R}^{m \times n^2}$ to be the matrix where i-th row is the vectorization of $B_i \in \mathbb{R}^{n \times n}$. Let b denote the sketching dimension and let T denote the number of iterations. Let $R_1, \cdots, R_s \in \mathbb{R}^{b \times n^2}$ denote a list of sketching matrices. Let $\mathsf{R} \in \mathbb{R}^{sb \times n^2}$ denote the batch sketching matrices. Let $\varepsilon_{\mathrm{mp}} \in (0, 0.1)$ be a precision parameter. Let $a \in (0,1)$. There is a dynamic data structure that given a sequence of online matrices*

$$W^{(1)}, \cdots, W^{(T)} \subset \mathbb{R}^{n \times n}; \quad \textit{and} \quad h^{(1)}, \cdots, h^{(T)} \in \mathbb{R}^{n^2}$$

approximately maintains the projection matrices

$$\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B}$$

for matrices $W^{(k)} = U \Lambda^{(k)} U^\top \in \mathbb{R}^{n \times n}$ where $\Lambda^{(k)}$ is a diagonal matrix with non-negative entries and U is an orthonormal eigenbasis.

The data structure has the following operations:

- INIT$(\varepsilon_{\mathrm{mp}} \in (0, 0.1))$*: This step takes*

  $$mn^\omega + m^\omega + \mathcal{T}_{\mathrm{mat}}(m, m, n^2) + \mathcal{T}_{\mathrm{mat}}(n^2, n^2, sb)$$

  time in the worst case.

- UPDATE(W)*: Output a matrix $\widetilde{V} \in \mathbb{R}^{n \times n}$ such that for all $i \in [n]$*

  $$(1 - \varepsilon_{\mathrm{mp}}) \cdot \lambda_i(\widetilde{V}) \leq \lambda_i(W) \leq (1 + \varepsilon_{\mathrm{mp}}) \cdot \lambda_i(\widetilde{V})$$

  where $\lambda_i(W)$ denote the i-th entry of the Λ matrix for W. This operation takes $O(n^{f(a,c)})$ time in the worst case, where n^{1+c} is the rank change in UPDATE *and $f(a,c)$ is defined in Def. B.14.*

[1] Our algorithm also works if $B_i = A_i W$.

- QUERY(h): *Output* $\widetilde{\mathsf{B}}^\top(\widetilde{\mathsf{B}}\widetilde{\mathsf{B}}^\top)^{-1}\widetilde{\mathsf{B}}R_l^\top R_l \cdot h$ *for the* $\widetilde{B}$ *defined by positive definite matrix* $\widetilde{V} \in \mathbb{R}^{n\times n}$ *outputted by the last call to* UPDATE. *This operation takes time*

$$n^{3+a+o(1)} + n^{2+b+o(1)},$$

if $\mathrm{nnz}(U) = O(n^{3/2+a/2})$.

For simplicity, consider the regime $m = n^2$. Then the initialization takes $O(m^\omega) + \mathcal{T}_{\mathrm{mat}}(m, m, sb)$ time, for $sb < m$, this is $O(m^\omega)$. For update, it takes $O(m^{\frac{f(a,c)}{2}})$ time, and the parameter c captures the *auxiliary rank* of the update. Each time we perform a low rank update, we make sure that the rank is at most $r = n^{1+c}$, and the complexity depends on c. In particular, if $\omega = 2$, which is the commonly-held belief, then $f(a, c) = 4$ for any $c \in [0, 1]$.

In both cases, the amortized cost per iteration is $o(m^2)$. For query, the cost is $m^{1.5+a/2+o(1)}$. This means that in addition to the initialization, the amortized cost per iteration of our data structure is $o(m^2)$.

We give an amortized analysis for UPDATE as follows:

Lemma 5.3. *Assume the notations are the same as those stated in Theorem 5.2. Furthermore, if the initial vector* $W^{(0)} \in \mathbb{R}^{n\times n}$ *and the (random) update sequence*

$$W^{(1)}, W^{(2)}, \cdots, W^{(T)} \in \mathbb{R}^{n\times n}$$

satisfies

$$\sum_{i=1}^{n}(\mathbb{E}[\ln\lambda_i(W^{(k+1)})] - \ln(\lambda_i(W^{(k)})))^2 \le C_1^2$$

and

$$\sum_{i=1}^{n}(\mathbf{Var}[\ln\lambda_i(W^{(k+1)})])^2 \le C_2^2$$

with the expectation and variance are conditioned on $\lambda_i(W^{(k)})$ *for all* $k = 0, 1, \cdots, T-1$. *Then, the amortized expected time*[2] *per call of* UPDATE(W) *is*

$$(C_1/\varepsilon_{\mathrm{mp}} + C_2/\varepsilon_{\mathrm{mp}}^2) \cdot (n^{f(c)+\omega-5/2+o(1)} + n^{f(c)-a/2+o(1)}).$$

Let us set $C_1, C_2 = 1/\log n$ and $\varepsilon_{\mathrm{mp}} = 0.01$ for the sake of discussion. Notice that under current best matrix multiplication exponent, by properly choosing a based on the auxiliary rank c, we can make sure that $f(a, c) - 5/2 \le \omega - 1/2$ and $f(a, c) - a/2 < 4$, hence, if our algorithm has $\sqrt{n}$ iterations, this means that the overall running time is at most

$$n^{2\omega} + n^{f(a,c)-a/2+0.5},$$

recall $m = n^2$, in terms of m, it becomes

$$m^\omega + m^{\frac{f(a,c)-a/2}{2}+1/4},$$

since $f(a, c) - a/2 < 4$, the second term is strictly smaller than $m^{2+1/4}$.

Algorithm 1 An informal version of our projection maintenance data structure

```
procedure INIT($\mathsf{A} \in \mathbb{R}^{m\times n^2}, \varepsilon_{\mathrm{mp}} \in (0, 0.1), W \in \mathbb{R}^{n\times n}$)
    Let $W = U\Lambda U^\top$
    Store $\mathsf{G} \leftarrow \mathsf{A}(U \otimes U)$
    Store $M \leftarrow \mathsf{G}^\top(\mathsf{G}(\Lambda\otimes\Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}$
    Prepare batched sketching $\mathsf{R} \leftarrow [R_1, \ldots, R_s] \in \mathbb{R}^{sb\times n^2}$
    Store $Q \leftarrow M(\Lambda^{1/2}\otimes\Lambda^{1/2})(U^\top \otimes U^\top)\mathsf{R}^\top$
    Store $P \leftarrow (\Lambda^{1/2}\otimes\Lambda^{1/2})(U^\top \otimes U^\top)Q$
end procedure

procedure UPDATE($W^{\mathrm{new}}$)
    $y_i \leftarrow \ln\lambda_i^{\mathrm{new}} - \ln\lambda_i$
    Let $r$ denotes the number such that $|y_i| \ge \varepsilon_{\mathrm{mp}}/2$
    if $r < n^a$ then
        $\widehat{\lambda} \leftarrow \lambda$
        Keep $M, Q, P$ the same
    else
        $\widehat{\lambda}, r \leftarrow$ SOFTTHRESHOLD($\lambda, \lambda^{\mathrm{new}}, r$)    ▷ Create a new vector that finds the correct number of entries needs to be updated
        Update $M, Q, P$ using matrix Woodbury identity
    end if
    $\widetilde{\lambda}_i \leftarrow \begin{cases}\widehat{\lambda}_i & \text{if } |\ln\lambda_i^{\mathrm{new}} - \ln\widehat{\lambda}_i| \le \varepsilon_{\mathrm{mp}}/2\\ \lambda_i^{\mathrm{new}} & \text{otherwise}\end{cases}$
    return $\widetilde{\lambda}$
end procedure

procedure QUERY($h^{\mathrm{new}} \in \mathbb{R}^{n^2}$)
    Let $\widetilde{P}$ be the projection whose $\lambda$ being updated to $\widetilde{\lambda}$
    Compute $p_g$ as $\widetilde{P}R^\top Rh$ using matrix Woodbury identity
    Compute $p_l$ as $(I - \widetilde{P})R^\top Rh$
    return $p_g, p_l$
end procedure
```

We give an overview of our data structure (Algorithm 1).

[2] When the input is deterministic, the output and the running time of UPDATE is also deterministic.

As we have covered in Section 4, we maintain the matrix

$$\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B}\mathsf{R},$$

where R is a batch of s sketching matrices. When receiving an update $W^{\rm new}$, we utilize the fact that $W^{\rm new}$ has the form $U\Lambda^{\rm new}U^\top$ where only the diagonal matrix Λ changes. We then perform a lazy update on Λ, when its entries don't change too much, we defer the update. Otherwise, we compute a threshold on how many entries need to be updated, and update all maintained variables using matrix Woodbury identity. Then, we use a fresh sketching matrix to make sure that the randomness has not been leaked.

By using an amortization framework based on fast rectangular matrix multiplication (Gall & Urrutia, 2018), we show that the amortized update time is faster than n^4, which is the size of the projection matrix. The query time is also faster than directly multiplying a length n^2 vector with an $n^2 \times n^2$ matrix.

6. Set Query Data Structure

In this section, we study an abstraction and generalization of the online matrix-vector multiplication problem. Given a projection matrix $\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B}$ and a query vector h, we only want to output a subset of entries of the vector $\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B}h$. A prominent example is we know some entries of $\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B}h$ are above some threshold τ and have already located their indices using sparse recovery tools, then the goal is to output the estimations of values of these entries.

To improve the runtime efficiency and space usage of Monte Carlo data structures, randomness is typically exploited and made internal to the data structure. Examples such as re-using sketching matrices and locality-sensitive hashing (Indyk & Motwani, 1998). To utilize the efficiency brought by internal randomness, these data structures assume the query sequence is chosen *oblivious* to its predetermined randomness. This assumption, however, is not sufficient when incorporating a data structure in an iterative process, oftentimes the input query is chosen based on the output from the data structure over prior iterations. Since the query is no longer independent of the internal randomness of the data structure, the success probability guaranteed by the Monte Carlo data structure usually fails.

From an adversary model perspective, this means that the adversary is *adaptive*, meaning that it can design input query based on the randomness leaked from the data structure over prior interactions. If we desire to use our projection maintenance data structure (Alg. 1) for efficient query, we need to initialize T different sketching matrices and for each iteration, using a fresh new sketching. This is commonly adapted by prior works, such as (Lee et al., 2019; Song & Yu, 2021; Qin et al., 2023a). However, the linear dependence on T becomes troublesome for large number of iterations.

How to reuse the randomness of the data structure while preventing the randomness leakage to an adaptive adversary? (Beimel et al., 2022) provides an elegant solution based on differential privacy. Build upon and extend their framework, we show that $\widetilde{O}(\sqrt{T})$ sketches suffice instead of T sketches.

In Section 6.1, we present the definition of the set query and estimation problem. In Section 6.2, we present our main result for the set query problem.

6.1. Problem Definition

In this section, we present the definition and the goal of the set query problem.

Definition 6.1 (Set Query)**.** Let $G \in \mathbb{R}^{n\times n}$ and $h \in \mathbb{R}^n$. Given a set $Q \subseteq [n]$ and $|Q| = k$, the goal is to estimate the norm of coordinates of Gh in set Q. Given a precision parameter ε, for each $j \in Q$, we want to design a function f such that

$$f(G, h)_j \in (g_j^\top h)^2 \pm \varepsilon \|g_j\|_2^2 \|h\|_2^2$$

where g_j denotes the j-th row of G.

6.2. Robust Set Query Data Structure

In this section, we design a robust set query data structure against an adaptive adversary.

To give an overview, consider estimating only one coordinate. We prepare $\widetilde{O}(\sqrt{T})$ sketching matrices and initialize them in a batched fashion. During query stage, we sample $\widetilde{O}(1)$ sketching matrices, and compute the inner product between the corresponding row of the (sketched) projection matrix and the sketched vector. This gives us $\widetilde{O}(1)$ estimators, we then run a PRIVATEMEDIAN algorithm (Theorem C.17) to obtain a real-valued output. This makes sure that we *do not reveal the randomness of the sketching matrices we use*. Using a standard composition result in differential privacy (Theorem C.14), we reduce the required number of sketches from T to $\widetilde{O}(\sqrt{T})$.

Lifting from a single coordinate estimation to k coordinates, however, is highly nontrivial. A direct application of strong composition over k coordinates will mandate $\widetilde{O}(\sqrt{kT})$ sketches. If $k < T$, this approach still leads to an improvement over T sketches, but its application scenario is much more limited. We observe the fundamental limitation of the aforementioned approach is that the per-coordinate estimation is relatively low-accuracy, therefore, to compensate for the error, we have to use more data structures. Our strategy is to instead estimate each coordinate to high-precision. This will incur extra estimation cost per coordinate, but the blowup is moderate. Moreover, we ensure that $\widetilde{O}(\sqrt{T})$ sketches suffices, effectively generalize the (Beimel et al., 2022) result to higher dimension.

Theorem 6.2 (Reduction to Adaptive Adversary: Set Query. Informal version of Theorem D.2)**.** *Let $\delta, \alpha > 0$ be parameters. Let f be a function that maps elements from domain $G \times H$ to an element in $\mathcal{U}^d$, where*

$$\mathcal{U} := [-U, -\frac{1}{U}] \cup \{0\} \cup [\frac{1}{U}, U]$$

for $U > 1$. Suppose there is a dynamic algorithm $\mathcal{A}$ against an oblivious adversary that, given an initial data point $x_0 \in X$ and T updates, guarantees the following:

- *The preprocessing time is $\mathcal{T}_{\mathrm{prep}}$.*
- *The per round update time is $\mathcal{T}_{\mathrm{update}}$.*
- *The per round query time is $\mathcal{T}_{\mathrm{query}}$ and given a set $Q_t \subseteq [n]$ with cardinality k, with probability $\geq 9/10$, the algorithm outputs $f(G_t, h_t)_j$ where $j \in Q_t$, and each $f(G_t, h_t)_j$ satisfies the following guarantee:*

$$f(g_j, h_t)_j \geq (g_j^\top h_t)^2 - \gamma \|g_j\|_2^2 \|h_t\|_2^2$$
$$f(g_j, h_t)_j \leq (g_j^\top h_t)^2 + \gamma \|g_j\|_2^2 \|h_t\|_2^2$$

  where g_j denotes the j-th row of matrix G_t.

Then, there exists a dynamic algorithm $\mathcal{B}$ against an adaptive adversary, guarantees the following:

- *The preprocessing time is*

$$\widetilde{O}(\sqrt{T} \log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\mathrm{prep}}).$$

- *The per round update time is*

$$\widetilde{O}(\sqrt{T} \log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\mathrm{update}}).$$

- *The per round query time is*

$$\widetilde{O}(\log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\mathrm{query}} + k^{3/2} \mathrm{poly}\log(\frac{\log U}{\alpha\delta}))$$

  and, with probability $1 - \delta$, for every $j \in Q_t$, the answer u_t is an $(\alpha + \gamma + \alpha\gamma)$-approximation of $(g_j^\top h_t)^2$ for all t, i.e.

$$(u_t)_j \geq (g_j^\top h_t)^2 - (\alpha + \gamma + \alpha\gamma) \|g_j\|_2^2 \|h_t\|_2^2$$
$$(u_t)_j \leq (g_j^\top h_t)^2 + (\alpha + \gamma + \alpha\gamma) \|g_j\|_2^2 \|h_t\|_2^2$$

Remark 6.3. Note that if we pick $k = n$ for projection maintenance or $k = n^2$ for Kronecker product projection maintenance, then we recover the projection matrix-vector multiplication problem.

In the projection maintenance task, we reduce the number of sketches *from T to $\widetilde{O}(\sqrt{T})$*, implying a faster preprocessing and update time compare to (Cohen et al., 2019; Lee et al., 2019; Song & Yu, 2021). The query time requires an extra $\widetilde{O}(n^{1.5})$ term, which matches $\mathcal{T}_{\mathrm{query}}$ in their applications. In the case of Kronecker product projection maintenance, we retain the same improvement on number of sketches, and incurring an extra $\widetilde{O}(n^3)$ factor in query. It is subsumed by $\mathcal{T}_{\mathrm{query}}$.

We also want to point out that our robust set query algorithm is more sensitive than the standard matrix-vector product task, since its running time depends on the number of coordinates that is needed by specific applications. For problems that only require outputting a small number of coordinates, it gives a tighter running time.

Beyond set query problem, our methodology for producing a high-dimensional, robust output against an adaptive adversary can well extend to other problems. In particular, suppose a Monte Carlo data structure that against an oblivious adversary outputs an n-dimensional vector. To output an estimator against an adaptive adversary, we can compute a high-precision private median over all vectors produced by the $\widetilde{O}(\sqrt{T})$ data structures. Suppose we only require k coordinates of the result, then the runtime blows up by a factor of $\widetilde{O}(k^{1.5})$ while reducing the number of data structures from T to $\widetilde{O}(\sqrt{T})$.

Very recently, the work (Cherapanamjeri et al., 2023) adapts the (Beimel et al., 2022) framework for a wide range of applications, including the norm of matrix-vector product, the value of linear regression, the distance estimates and the optimal value of kernel density estimation. However, these results are just outputting a single *value*, rather than a vector which is often times of particularly interest. Our framework, on the other hand, supports outputting a vector, which will hopeful lead to further developments to these problems.

Acknowledgement

The authors would like to thank Jonathan Kelner for many helpful discussions, Shyam Narayanan for discussions about differential privacy and Jamie Morgenstern for continued support and encouragement. Xin Yang is supported in part by NSF grant No. CCF-2006359. Yuanyuan Yang is supported by NSF grant No. CCF-2045402 and NSF grant No. CCF-2019844. Lichen Zhang is supported by NSF grant No. CCF-1955217 and NSF grant No. CCF-2022448.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Alon, N., Matias, Y., and Szegedy, M. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

Andoni, A., Lin, C., Sheng, Y., Zhong, P., and Zhong, R. Subspace embedding and linear regression with orlicz norm. In *International Conference on Machine Learning (ICML)*, pp. 224–233. PMLR, 2018.

Bagdasaryan, E., Poursaeed, O., and Shmatikov, V. Differential privacy has disparate impact on model accuracy. *Advances in Neural Information Processing Systems (NeurIPS)*, 32:15479–15488, 2019.

Bassily, R., Nissim, K., Smith, A., Steinke, T., Stemmer, U., and Ullman, J. Algorithmic stability for adaptive data analysis. In *STOC*, 2016.

Beimel, A., Kaplan, H., Mansour, Y., Nissim, K., Saranurak, T., and Stemmer, U. Dynamic algorithms against an adaptive adversary: Generic constructions and lower bounds. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1671–1684, 2022.

Bernstein, S. On a modification of chebyshev's inequality and of the error formula of laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math*, 1(4):38–49, 1924.

Boutsidis, C. and Woodruff, D. P. Optimal cur matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 353–362. ACM, 2014.

Boutsidis, C., Woodruff, D. P., and Zhong, P. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pp. 236–249, 2016.

Brand, J. v. d. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 259–278. SIAM, 2020.

Brand, J. v. d., Lee, Y. T., Sidford, A., and Song, Z. Solving tall dense linear programs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 775–788, 2020.

Brand, J. v. d., Peng, B., Song, Z., and Weinstein, O. Training (overparametrized) neural networks in near-linear time. In *ITCS*, 2021.

Brand, J. v. d., Song, Z., and Zhou, T. Algorithm and hardness for dynamic attention maintenance in large language models. *arXiv preprint arXiv:2304.02207*, 2023.

Bun, M., Nissim, K., Stemmer, U., and Vadhan, S. Differentially private release and learning of threshold functions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 634–649, 2015.

Bun, M., Kamath, G., Steinke, T., and Wu, Z. S. Private hypothesis selection. *IEEE Trans. Inform. Theory*, 67 (3):1981–2000, 2021.

Chakraborty, D., Kamma, L., and Larsen, K. G. Tight cell probe bounds for succinct boolean matrix-vector multiplication. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pp. 1297–1306, 2018.

Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 693–703. Springer, 2002.

Chaudhuri, K. and Monteleoni, C. Privacy-preserving logistic regression. In *NIPS*, volume 8, pp. 289–296. Citeseer, 2008.

Cherapanamjeri, Y., Silwal, S., Woodruff, D., Zhang, F., Zhang, Q., and Zhou, S. Robust algorithms on adaptive inputs from bounded adversaries. In *ICLR*, 2023.

Chernoff, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pp. 493–507, 1952.

Clarkson, K. L. and Woodruff, D. P. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference(STOC)*, pp. 81–90, 2013.

Cohen, M. B., Lee, Y. T., and Song, Z. Solving linear programs in the current matrix multiplication time. In *STOC*, 2019.

Deng, Y., Song, Z., and Weinstein, O. Discrepancy minimization in input-sparsity time. *arXiv preprint arXiv:2210.12468*, 2022a.

Deng, Y., Song, Z., Weinstein, O., and Zhang, R. Fast distance oracles for any symmetric norm. In *Advances in Neural Information Processing Systems*, 2022b.

Deng, Y., Li, Z., and Song, Z. Attention scheme inspired softmax regression. *arXiv preprint arXiv:2304.10411*, 2023a.

Deng, Y., Li, Z., and Song, Z. An improved sample complexity for rank-1 matrix sensing. *arXiv preprint arXiv:2303.06895*, 2023b.

Deng, Y., Mahadevan, S., and Song, Z. Randomized and deterministic attention sparsification algorithms for over-parameterized feature dimension. *arXiv preprint arXiv:2304.04397*, 2023c.

Diao, H., Song, Z., Woodruff, D., and Yang, X. Total least squares regression in input sparsity time. *Advances in Neural Information Processing Systems*, 32, 2019.

Ding, W., Kamath, G., Wang, W., and Shah, N. B. Calibration with privacy in peer review. In *2022 IEEE International Symposium on Information Theory (ISIT)*, 2022.

Dong, S., Lee, Y. T., and Ye, G. A nearly-linear time algorithm for linear programs with small treewidth: a multiscale representation of robust central path. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1784–1797, 2021.

Dwork, C. Differential privacy. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pp. 1–12, 2006.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 486–503. Springer, 2006.

Dwork, C., Rothblum, G. N., and Vadhan, S. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 51–60. IEEE, 2010.

Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., and Roth, A. L. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pp. 117–126, 2015.

Esfandiari, H., Mirrokni, V., and Zhong, P. Almost linear time density level set estimation via dbscan. In *AAAI*, 2021.

Gall, F. L. and Urrutia, F. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1029–1046, 2018.

Gao, Y., Mahadevan, S., and Song, Z. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*, 2023a.

Gao, Y., Song, Z., and Yang, X. Differentially private attention computation. *arXiv preprint arXiv:2305.04701*, 2023b.

Gao, Y., Song, Z., and Yin, J. An iterative algorithm for rescaled hyperbolic functions regression. *arXiv preprint arXiv:2305.00660*, 2023c.

Gu, Y. and Song, Z. A faster small treewidth sdp solver. *arXiv preprint arXiv:2211.06033*, 2022.

Gu, Y., Song, Z., Yin, J., and Zhang, L. Low rank matrix completion via robust alternating minimization in nearly linear time. *arXiv preprint arXiv:2302.11068*, 2023.

Henzinger, M., Krinninger, S., Nanongkai, D., and Saranurak, T. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pp. 21–30, 2015.

Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. ISSN 01621459.

Hopkins, S. B., Kamath, G., and Majid, M. Efficient mean estimation with pure differential privacy via a sum-of-squares exponential mechanism. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, 2022.

Hopkins, S. B., Kamath, G., Majid, M., and Narayanan, S. Robustness implies privacy in statistical estimation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, 2023.

Hu, H., Song, Z., Weinstein, O., and Zhuo, D. Training overparametrized neural networks in sublinear time. In *arXiv preprint arXiv: 2208.04508*, 2022.

Huang, B., Jiang, S., Song, Z., Tao, R., and Zhang, R. A faster quantum algorithm for semidefinite programming via robust ipm framework. *arXiv preprint arXiv:2207.11154*, 2022a.

Huang, B., Jiang, S., Song, Z., Tao, R., and Zhang, R. Solving sdp faster: A robust ipm framework and efficient implementation. In *FOCS*, 2022b.

Indyk, P. and Motwani, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC)*, pp. 604–613, 1998.

Jayaraman, B. and Evans, D. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1895–1912, 2019.

Jiang, H., Kathuria, T., Lee, Y. T., Padmanabhan, S., and Song, Z. A faster interior point method for semidefinite programming. In *FOCS*, 2020a.

Jiang, H., Lee, Y. T., Song, Z., and Wong, S. C.-w. An improved cutting plane method for convex optimization, convex-concave games and its applications. In *STOC*, 2020b.

Jiang, H., Lee, Y. T., Song, Z., and Zhang, L. Convex minimization with integer minima in $\widetilde{O}(n^4)$ time. *CoRR*, abs/2304.03426, 2023a.

Jiang, R. and Li, D. Simultaneous diagonalization of matrices and its applications in quadratically constrained quadratic programming. *SIAM Journal on Optimization*, 2016.

Jiang, S., Song, Z., Weinstein, O., and Zhang, H. Faster dynamic matrix inverse for faster lps. In *STOC*, 2021.

Jiang, S., Natura, B., and Weinstein, O. A faster interior-point method for sum-of-squares optimization. *49th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2022.

Jiang, S., Peng, B., and Weinstein, O. The complexity of dynamic least-squares regression. *arxiv preprint arxiv:2201.00228*, 2023b.

Kairouz, P., Oh, S., and Viswanath, P. The composition theorem for differential privacy. In *International conference on machine learning*, pp. 1376–1385. PMLR, 2015.

Kamath, G., Sheffet, O., Singhal, V., and Ullman, J. Differentially private algorithms for learning mixtures of separated gaussians. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Kamath, G., Singhal, V., and Ullman, J. Private mean estimation of heavy-tailed distributions. In *Proceedings of the 33rd Annual Conference on Learning Theory (COLT)*, 2020.

Kamath, G., Liu, X., and Zhang, H. Improved rates for differentially private stochastic convex optimization with heavy-tailed data. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 10633–10660, 2022.

Larsen, K. G. and Williams, R. Faster online matrix-vector multiplication. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2182–2189, 2017.

Lee, J. D., Shen, R., Song, Z., Wang, M., and Yu, Z. Generalized leverage score sampling for neural networks. In *NeurIPS*, 2020.

Lee, Y. T., Song, Z., and Zhang, Q. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory (COLT)*, pp. 2140–2157. PMLR, 2019.

Li, S., Song, Z., Xia, Y., Yu, T., and Zhou, T. The closeness of in-context learning and weight shifting for softmax regression. *arXiv preprint arXiv:2304.13276*, 2023a.

Li, Z., Song, Z., and Zhou, T. Solving regularized exp, cosh and sinh regression problems. *arXiv preprint arXiv:2303.15725*, 2023b.

Liang, Y., Song, Z., Wang, M., Yang, L., and Yang, X. Sketching transformed matrices with applications to natural language processing. In *International Conference on Artificial Intelligence and Statistics*, pp. 467–481. PMLR, 2020.

Lu, Y., Dhillon, P., Foster, D. P., and Ungar, L. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems (NIPS)*, pp. 369–377, 2013.

Luo, Z., Wu, D. J., Adeli, E., and Fei-Fei, L. Scalable differential privacy with sparse network finetuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5059–5068, 2021.

Meng, X. and Mahoney, M. W. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing (STOC)*, pp. 91–100, 2013.

Mohapatra, S., Sasy, S., He, X., Kamath, G., and Thakkar, O. The role of adaptive optimizers for honest private hyperparameter selection. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*, 2022.

Nelson, J. and Nguyên, H. L. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 117–126. IEEE, 2013.

Price, E. Efficient sketches for the set query problem. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, 2011.

Qin, L., Song, Z., Zhang, L., and Zhuo, D. An online and unified algorithm for projection matrix vector multiplication with application to empirical risk minimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 101–156. PMLR, 2023a.

Qin, L., Song, Z., and Zhang, R. A general algorithm for solving rank-one matrix sensing. *arXiv preprint arXiv:2303.12298*, 2023b.

Razenshteyn, I., Song, Z., and Woodruff, D. P. Weighted low rank approximations with provable guarantees. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pp. 250–263, 2016.

Reddy, A., Song, Z., and Zhang, L. Dynamic tensor product regression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Song, Z. *Matrix theory: optimization, concentration, and algorithms*. The University of Texas at Austin, 2019.

Song, Z. and Yu, Z. Oblivious sketching-based central path method for linear programming. In *International Conference on Machine Learning (ICML)*, pp. 9835–9847. PMLR, 2021.

Song, Z., Woodruff, D. P., and Zhong, P. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*, 2017.

Song, Z., Woodruff, D. P., and Zhong, P. Relative error tensor low rank approximation. In *SODA*, 2019.

Song, Z., Yang, S., and Zhang, R. Does preprocessing help training over-parameterized neural networks? *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021a.

Song, Z., Zhang, L., and Zhang, R. Training multi-layer over-parametrized neural network in subquadratic time. *arXiv preprint arXiv:2112.07628*, 2021b.

Song, Z., Sun, B., Weinstein, O., and Zhang, R. Sparse fourier transform over lattices: A unified approach to signal reconstruction. *arXiv preprint arXiv:2205.00658*, 2022a.

Song, Z., Yang, X., Yang, Y., and Zhou, T. Faster algorithm for structured john ellipsoid computation. *arXiv preprint arXiv:2211.14407*, 2022b.

Subramani, P., Vadivelu, N., and Kamath, G. Enabling fast differentially private sgd via just-in-time compilation and vectorization. In *Advances in Neural Information Processing Systems*, 2021.

Sun, J., Yang, X., Yao, Y., Xie, J., Wu, D., and Wang, C. Dpauc: Differentially private auc computation in federated learning. *arXiv preprint arXiv:2208.12294*, 2022.

Torkzadehmahani, R., Kairouz, P., and Paten, B. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshop)*, 2019.

Triastcyn, A. and Faltings, B. Bayesian differential privacy for machine learning. In *International Conference on Machine Learning*, pp. 9583–9592. PMLR, 2020.

Vaidya, P. M. A new algorithm for minimizing convex functions over convex sets. In *30th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 338–343, 1989a.

Vaidya, P. M. Speeding-up linear programming using fast matrix multiplication. In *30th annual symposium on foundations of computer science*, pp. 332–337. IEEE Computer Society, 1989b.

Wang, R., Zhong, P., Du, S. S., Salakhutdinov, R. R., and Yang, L. F. Planning with general objective functions: Going beyond total rewards. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Weggenmann, B. and Kerschbaum, F. Syntf: Synthetic and differentially private term frequency vectors for privacy-preserving text mining. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 305–314, 2018.

Williams, O. and McSherry, F. Probabilistic inference and differential privacy. *Advances in Neural Information Processing Systems (NeurIPS)*, 23:2451–2459, 2010.

Woodruff, D. P. and Zhong, P. Distributed low rank approximation of implicit functions of a matrix. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pp. 847–858. IEEE, 2016.

Wu, R., Yang, X., Yao, Y., Sun, J., Liu, T., Weinberger, K. Q., and Wang, C. Differentially private multiparty data release for linear regression. *arXiv preprint arXiv:2206.07998*, 2022.

Xiao, C., Zhong, P., and Zheng, C. Bourgan: generative networks with metric embeddings. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2275–2286, 2018.

Xu, Z., Song, Z., and Shrivastava, A. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures.

In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Yang, X., Sun, J., Yao, Y., Xie, J., and Wang, C. Differentially private label protection in split learning. *arXiv preprint arXiv:2203.02073*, 2022.

Ye, G. Fast algorithm for solving structured convex programs. *The University of Washington, Undergraduate Thesis*, 2020.

Yu, D., Naik, S., Backurs, A., Gopi, S., Inan, H. A., Kamath, G., Kulkarni, J., Lee, Y. T., Manoel, A., Wutschitz, L., Yekhanin, S., and Zhang, H. Differentially private fine-tuning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022.

Yue, X., Du, M., Wang, T., Li, Y., Sun, H., and Chow, S. S. M. Differential privacy for text analytics via natural text sanitization. In *Findings, ACL-IJCNLP 2021*, 2021.

Zhu, Y., Yu, X., Chandraker, M., and Wang, Y.-X. Private-knn: Practical differential privacy for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11854–11862, 2020.

Appendix

Roadmap. In Section A, we present the preliminaries of this paper. In Section B, we design a Kronecker product projection maintenance data structure that has fast update and query time. In Section C, we use differential privacy techniques to design a robust norm estimation data structure. In Section D, we extend our DP mechanisms to develop a robust set query data structure.

A. Preliminaries

Notations For any integer $n > 0$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. Let $\Pr[\cdot]$ denote probability and $\mathbb{E}[\cdot]$ denote expectation. We use $\|x\|_2$ to denote the ℓ_2 norm of a vector x. We use $\mathcal{N}(\mu, \sigma^2)$ to denote the Gaussian distribution with mean μ and variance σ^2. We use $\widetilde{O}(f(n))$ to denote $O(f(n) \cdot \mathrm{poly}\log(f(n))$. We denote $\omega \approx 2.38$ as the matrix multiplication exponent. We denote $\alpha \approx 0.31$ as the dual exponent of matrix multiplication.

We use $\|A\|$ and $\|A\|_F$ to denote the spectral norm and the Frobenius norm of matrix A, respectively. We use $A^\top$ to denote the transpose of matrix A. We use I_m to denote the identity matrix of size $m \times m$. For α being a vector or matrix, we use $\|\alpha\|_0$ to denote the number of nonzero entries of α. Given a real square matrix A, we use $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ to denote its largest and smallest eigenvalue, respectively. Given a real matrix A, we use $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$ to denote its largest and smallest singular value, respectively. We use A^{-1} to denote the matrix inverse for matrix A. For a square matrix A, we use $\mathrm{tr}[A]$ to denote the trace of A. We use b and n^b interchangeably to denote the sketching dimension, and $b \in [0, 1]$ when the sketching dimension is n^b.

Given an $n_1 \times d_1$ matrix A and an $n_2 \times d_2$ matrix B, we use $\otimes$ to denote the Kronecker product, i.e., $A \otimes B$ is a matrix where its $(i_1 + (i_2 - 1) \cdot n_1, j_1 + (j_2 - 1) \cdot d_1)$-th entry is $A_{i_1,j_1} \cdot B_{i_2,j_2}$. For matrix $A \in \mathbb{R}^{n \times n}$, we denote $\mathrm{vec}(A)$ as the vectorization of A. We use $\langle \cdot, \cdot \rangle$ to denote the inner product, when applied to two vectors, this denotes the standard dot product between two vectors, and when applied to two matrices, this means $\langle A, B \rangle = \sum_{i,j} A_{i,j} B_{i,j}$. Further, $\langle A, B \rangle = \mathrm{tr}[A^\top B]$.

We denote the data/constraint matrix as $\mathsf{A} \in \mathbb{R}^{m \times n^2}$, weight matrix as $W \in \mathbb{R}^{n \times n}$, and the resulting projection matrix as $\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B}$, where $\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2}) \in \mathbb{R}^{m \times n^2}$. Additionally, we denote matrix $\Delta \in \mathbb{R}^{n \times n}$ as the update matrix of projection maintenance, Δ has rank k and has the same eigenbasis as matrix W. We denote $A_1, \ldots, A_m \in \mathbb{R}^{n \times n}$ as the collection of data constraint matrices, $\mathsf{A} = \begin{bmatrix} \mathrm{vec}(A_1) & \mathrm{vec}(A_2) & \cdots & \mathrm{vec}(A_m) \end{bmatrix}^\top \in \mathbb{R}^{m \times n^2}$ as the batched constraint matrix, and $\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B}$ as the projection matrix, where m is given. We denote $h \in \mathbb{R}^{n^2}$ as the vector that the data structure receives to be projected.

We denote $\varepsilon_{\mathsf{mp}} \in [0, 0.1]$ as the tolerance parameter. We denote T as the number of iterations. We denote δ as the failure probability, and α as the parameter for the dynamic algorithm against an adaptive adversary. We denote $\mathcal{T}_{\mathrm{prep}}, \mathcal{T}_{\mathrm{update}}, \mathcal{T}_{\mathrm{query}}$ as the preprocessing time, update time, and query time for the dynamic algorithm against an oblivious adversary. We denote $U > 1$ as the output parameter and $\mathcal{U}$ as the output range of the above dynamic algorithm, where every coordinate of the output v satisfies $v \in \mathcal{U} = [-U, -\frac{1}{U}] \cup \{0\} \cup [\frac{1}{U}, U]$.

Probability tools We present the probability tools we will use in this paper, and all of them are exponentially decreasing bounds. At first, we present the Chernoff bound, which bounds the probability that the sum of independent random *Boolean* variables deviates from its true mean by a certain amount.

Lemma A.1 (Chernoff bound (Chernoff, 1952)). *Let $X = \sum_{i=1}^n X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^n p_i$. Then*

- $\Pr[X \geq (1 + \delta)\mu] \leq \exp(-\delta^2 \mu / 3)$, $\forall \delta > 0$;
- $\Pr[X \leq (1 - \delta)\mu] \leq \exp(-\delta^2 \mu / 2)$, $\forall 0 < \delta < 1$.

Next, we present the Hoeffding bound, which bounds the probability that the sum of independent random *bounded* variables deviates from its true mean by a certain amount.

Lemma A.2 (Hoeffding bound (Hoeffding, 1963)). *Let $X_1, \cdots, X_n$ denote n independent bounded variables in $[a_i, b_i]$.*

Let $X = \sum_{i=1}^n X_i$, then we have

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq 2 \exp\left(- \frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2} \right).$$

Finally, we present the Bernstein inequality, which bounds the probability that the sum of independent random *bounded zero-mean* variables deviates from its true mean.

Lemma A.3 (Bernstein inequality (Bernstein, 1924))**.** *Let $X_1, \cdots, X_n$ be independent zero-mean random variables. Suppose that $|X_i| \leq M$ almost surely, for all i. Then, for all positive t,*

$$\Pr\left[\sum_{i=1}^n X_i > t \right] \leq \exp\left(- \frac{t^2/2}{\sum_{j=1}^n \mathbb{E}[X_j^2] + Mt/3} \right).$$

B. Kronecker Product Projection Maintenance Data Structure

This section is organized as follows: We introduce some basic calculation rules for Kronecker product in Section B.1. We give the visualization of OMV, OPMV and OKPMV in Section B.2, Section B.3, and Section B.4, respectively. We introduce the projection matrix and its properties in Section B.5. We present our data structure in Section B.6. We present our main results for Kronecker projection maintenance in Section B.7.

B.1. Basic Linear Algebra for Kronecker Product

In this section, we state a number of useful facts for Kronecker product:

At first, we present the mixed product property regarding the interchangeability of the conventional matrix product and the Kronecker product.

Fact B.1 (Mixed Product Property)**.** *Given conforming matrices A, B, C and D, we have*

$$(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D),$$

where $\cdot$ denotes matrix multiplication.

Next, we present the inversion property regarding the calculation on the inverse of Kronecker product of two conforming matrices.

Fact B.2 (Inversion)**.** *Let A, B be full rank square matrices. Then we have*

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}.$$

Next, we present a fact regarding the vectorization of conventional matrix product of two conforming matrices and their Kronecker product with identity matrix.

Fact B.3. *Let I_m, I_k denote $m \times m$ and $k \times k$ identity matrix, respectively, and $A \in \mathbb{R}^{m \times k}, B \in \mathbb{R}^{k \times m}$ be conforming matrices, then:*

$$\mathrm{vec}(AB) = (I_m \otimes A)\mathrm{vec}(B) = (B^\top \otimes I_k)\mathrm{vec}(A)$$

We present a fact regarding the vectorization of conventional matrix product of three conforming matrices and their Kronecker product.

Fact B.4. *Let A, B, C be conforming matrices, then:*

$$\mathrm{vec}(ABC) = (C^\top \otimes A)\mathrm{vec}(B)$$

We present a fact regarding the trace of the multiplication of two conforming matrices and their vectorization.

Fact B.5. *Let A, B be conforming matrices, then:*

$$\mathrm{tr}[A^\top B] = \mathrm{vec}(A)^\top \mathrm{vec}(B) = \mathrm{vec}(B)^\top \mathrm{vec}(A)$$

Finally, we present the cyclic property of trace calculation: The calculation of the trace of the conventional matrix product is invariant under cyclic permutation.

Fact B.6 (Cyclic)**.** *Let A, B, V be conforming matrices, then:*

$$\mathrm{tr}[ABC] = \mathrm{tr}[BCA] = \mathrm{tr}[CAB].$$

B.2. Online Matrix Vector Multiplication

In this section, we present the visualization of online matrix vector multiplication.

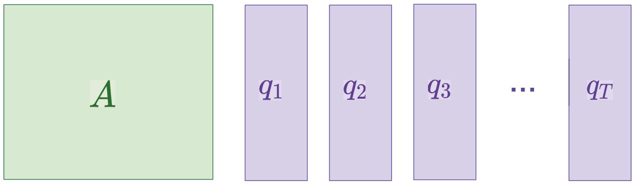

Figure 1: Online matrix vector multiplication (Definition 3.1).

B.3. Online Projection Matrix Vector Multiplication

In this section, we present the visualization of online projection matrix vector multiplication.

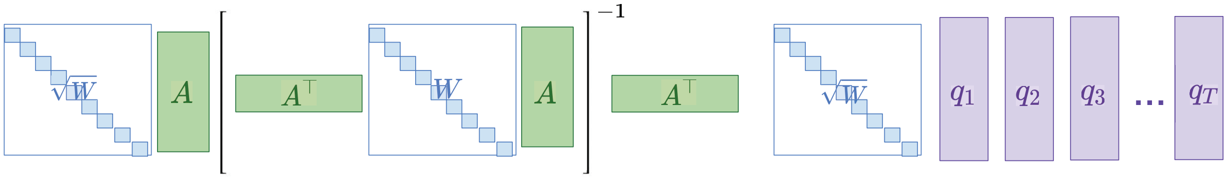

Figure 2: Online projection matrix vector multiplication (Definition 3.2). Usually, we say $\sqrt{W}A(A^\top WA)^{-1}A^\top\sqrt{W}$ is a projection matrix. We say $(A^\top WA)^{-1}A^\top\sqrt{W}$ is a projection matrix without left arm. We say $A(A^\top WA)^{-1}A^\top\sqrt{W}$ is a projection matrix without left hand. Technically, we call $\sqrt{W}A$ arm, and call $\sqrt{W}$ hand.

B.4. Online Kronecker Projection Matrix Vector Multiplication

In this section, we present the visualizations of online Kronecker projection matrix vector multiplication.

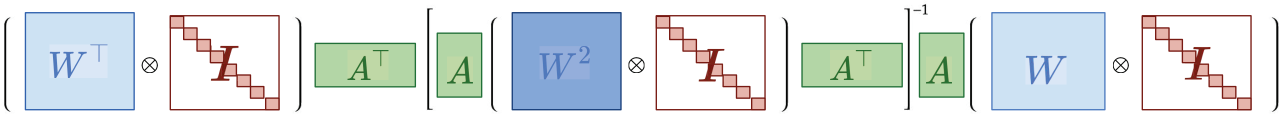

Figure 3: Online Kronecker matrix vector multiplication (Definition 3.3), where $B_i = A_i W$, and the projection matrix is defined as $\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B} = (W^\top \otimes I)\mathsf{A}^\top(\mathsf{A}(W^2 \otimes I)\mathsf{A}^\top)^{-1}\mathsf{A}(W \otimes I)$.

B.5. Preliminaries

We present the definitions of the matrices we will be using across the sections.

Definition B.7. Given a collection of matrices $A_1, \cdots, A_m \in \mathbb{R}^{n\times n}$, we define $\mathsf{A} \in \mathbb{R}^{m\times n^2}$ to be the batched matrix whose i-th row is the vectorization of A_i, for each $i \in [m]$. Let $W \in \mathbb{R}^{n\times n}$ be a positive semidefinite matrix. We define

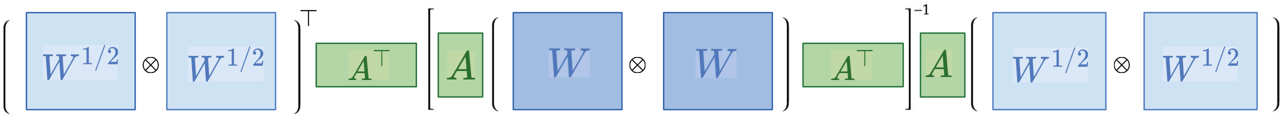

Figure 4: Online Kronecker matrix vector multiplication (Definition 3.3), where $B_i = W^{1/2} A_i W^{1/2}$, and the projection matrix is defined as $\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B} = (W^{1/2}\otimes W^{1/2})^\top \mathsf{A}^\top(\mathsf{A}(W\otimes W)\mathsf{A}^\top)^{-1}\mathsf{A}(W^{1/2}\otimes W^{1/2})$.

$\mathsf{B} \in \mathbb{R}^{m\times n^2}$ to be a matrix where each row is the vectorization of $B_i = A_i W$. The projection matrix corresponds to B is

$$\mathsf{B}^\top(\mathsf{B}\mathsf{B}^\top)^{-1}\mathsf{B} \in \mathbb{R}^{n^2\times n^2}.$$

Next, we present a fact regarding the batched matrix $\mathsf{B} \in \mathbb{R}^{m\times n^2}$, matrix $W \in \mathbb{R}^{n\times n}$, and the batched matrix A, if $B_i = A_i W$.

Fact B.8. *For $B_i, A_i, W \in \mathbb{R}^{n\times n}$, if $B_i = A_i W$, then we have*

- $\mathsf{B} = \mathsf{A}(W\otimes I) \in \mathbb{R}^{m\times n^2}$.
- $\mathsf{B}\mathsf{B}^\top = \mathsf{A}(W^2\otimes I)\mathsf{A}^\top \in \mathbb{R}^{m\times m}$.

where the i-th row of B, and A is $\mathrm{vec}(B_i)^\top, \mathrm{vec}(A_i)^\top$, respectively.

Proof. We note that each row of B is in the form of $\mathrm{vec}(B_i)^\top = \mathrm{vec}(A_i W)^\top$, hence,

$$\begin{aligned}\mathrm{vec}(B_i) &= \mathrm{vec}(A_i W)\\ &= \mathrm{vec}(I A_i W)\\ &= (W^\top \otimes I)\mathrm{vec}(A_i).\end{aligned}$$

where the last step follows from Fact B.3. Therefore, we have:

$$\begin{aligned}\mathrm{vec}(B_i)^\top &= \mathrm{vec}(A_i)^\top (W^\top\otimes I)^\top\\ &= \mathrm{vec}(A_i)^\top (W\otimes I).\end{aligned}$$

Hence, we derive that:

$$\mathsf{B} = \mathsf{A}(W\otimes I).$$

To verify $\mathsf{B}\mathsf{B}^\top = \mathsf{A}(W^2\otimes I)\mathsf{A}^\top$, we first compute $\mathsf{B}\mathsf{B}^\top$ by the original definition, we have:

$$\begin{aligned}(\mathsf{B}\mathsf{B}^\top)_{i,j} &= \mathrm{vec}(A_i W)^\top \mathrm{vec}(A_j W)\\ &= \mathrm{tr}[W A_i^\top A_j W]\\ &= \mathrm{tr}[A_j W^2 A_i^\top].\end{aligned}$$

where the second step follows from Fact B.5, the third step follows from Fact B.6.

Then, we calculate the (i,j)-th coordinate of $\mathsf{A}(W^2\otimes I)\mathsf{A}^\top$, and we have:

$$\begin{aligned}(\mathsf{A}(W\otimes I)(W\otimes I)\mathsf{A}^\top)_{i,j} &= \mathrm{vec}(A_i)^\top (W\otimes I)(W\otimes I)\mathrm{vec}(A_j)\\ &= \mathrm{vec}(I)^\top (I\otimes A_i^\top)(W\otimes I)(W\otimes I)(I\otimes A_j)\mathrm{vec}(I)\\ &= \mathrm{vec}(I)^\top (W\otimes A_i^\top)(W\otimes A_j)\mathrm{vec}(I)\\ &= \mathrm{vec}(A_i W)^\top \mathrm{vec}(A_j W)\\ &= \mathrm{tr}[W A_i^\top A_j W]\\ &= \mathrm{tr}[A_j W^2 A_i^\top].\\ &= (\mathsf{B}\mathsf{B}^\top)_{i,j}\end{aligned}$$

where the first step follows from the definition of A that $\mathrm{vec}(A_i)^\top$ is the i-th row of the matrix A, and $\mathrm{vec}(A_j)$ is the j-th column of the matrix $\mathsf{A}^\top$. The second step follows from Fact B.3. The third step follows from Fact B.1. The fourth step follows from Fact B.3. The fifth step follows from Fact B.5. The sixth step follows from Fact B.6. The final step follows by the definition of B. □

Next, we present a fact regarding the batched matrix $\mathsf{B} \in \mathbb{R}^{m \times n^2}$, matrix $W \in \mathbb{R}^{n \times n}$, and the batched matrix $\mathsf{A} \in \mathbb{R}^{m \times n^2}$, if $B_i = W^{1/2} A_i W^{1/2}$.

Fact B.9. *For positive semidefinite matrix $W \in \mathbb{R}^{n \times n}$, if $B_i = W^{1/2} A_i W^{1/2}$, then we have:*

- $\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2}) \in \mathbb{R}^{m \times n^2}$
- $\mathsf{B}\mathsf{B}^\top = \mathsf{A}(W \otimes W)\mathsf{A}^\top \in \mathbb{R}^{m \times m}$

Proof. Suppose $B_i = W^{1/2} A_i W^{1/2}$ whose vectorization is $\mathrm{vec}(W^{1/2} A_i W^{1/2})$, by Fact B.4, we have that:

$$\mathrm{vec}(W^{1/2} A_i W^{1/2}) = (W^{1/2} \otimes W^{1/2}) \mathrm{vec}(A_i).$$

Transposing the right hand side gives us:

$$\mathrm{vec}(A_i)^\top (W^{1/2} \otimes W^{1/2}),$$

therefore, we conclude that:

$$\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2}).$$

Consequently, we have:

$$\begin{aligned} \mathsf{B}\mathsf{B}^\top &= \mathsf{A}(W^{1/2} \otimes W^{1/2})(W^{1/2} \otimes W^{1/2})\mathsf{A}^\top \\ &= \mathsf{A}(W \otimes W)\mathsf{A}^\top. \end{aligned}$$

where the last step follows from Fact B.1. □

Next, we present a fact regarding the rank change of the matrix $W^2 \otimes I$ and the matrix $W \otimes W$ when W experiences a rank-k change.

Fact B.10. *Suppose $W \in \mathbb{R}^{n \times n}$ undergoes a rank-k change, i.e., $W \leftarrow W + \Delta$ where Δ has rank-k, then*

- *The matrix $W^2 \otimes I$ undergoes a rank-$3nk$ change.*
- *The matrix $W \otimes W$ undergoes a rank-$(2nk + k^2)$ change.*

Proof. For $W^2 \otimes I$, it suffices to understand the rank change on W^2. Note that:

$$(W + \Delta)^2 = W^2 + W\Delta + \Delta W + \Delta^2,$$

since Δ is rank-k, we know that $W\Delta$, ΔW and Δ^2 all have rank at most k. Hence, if we let $\widetilde{\Delta}$ to denote $W\Delta + \Delta W + \Delta^2$, we have:

$$\mathrm{rank}(\widetilde{\Delta}) \leq 3k.$$

Finally, note that:

$$\begin{aligned} (W + \Delta)^2 \otimes I &= (W^2 + \widetilde{\Delta}) \otimes I \\ &= W^2 \otimes I + \widetilde{\Delta} \otimes I, \end{aligned}$$

we have that the rank change of the matrix $W^2 \otimes I$ is the same as the rank of the matrix $\widetilde{\Delta} \otimes I$ that is at most $3nk$.

We now analyze the rank change of $W \otimes W$. Consider

$$(W+\Delta) \otimes (W+\Delta) = W \otimes W + W \otimes \Delta + \Delta \otimes W + \Delta \otimes \Delta,$$

the components $W \otimes \Delta$ and $\Delta \otimes W$ both have ranks nk, and $\Delta \otimes \Delta$ has rank k^2. Hence, if we let $\widetilde{\Delta}$ to denote $W \otimes \Delta + \Delta \otimes W + \Delta \otimes \Delta$, then $\mathrm{rank}(\widetilde{\Delta}) \leq 2nk + k^2$. □

Next, we present a fact regarding the rank change of the matrix $W^{1/2}$, when W experiences a rank-k change Δ that has the same eigenbasis of W.

Lemma B.11. *Suppose $W \in \mathbb{R}^{n \times n}$ undergoes a rank-k change Δ and W, Δ have the same eigenbasis, then the matrix $(W+\Delta)^{1/2}$ undergoes a rank-k change, i.e.,*

$$(W+\Delta)^{1/2} = W^{1/2} + \overline{\Delta},$$

where $\overline{\Delta}$ is rank k and shares the same eigenbasis as W.

Proof. By spectral theorem, we know that there exists U, Λ, and $\widetilde{\Delta}$ such that, $W = U \Lambda U^\top$ and $\Delta = U \widetilde{\Delta} U^\top$, while $\widetilde{\Delta}$ has only k nonzero entries. Hence, we notice that $W + \Delta = U(\Lambda + \widetilde{\Lambda})U^\top$ has only k entries being changed. Note that $(W+\Delta)^{1/2} = U(\Lambda + \widetilde{\Lambda})^{1/2} U^\top$, which means the diagonal only has k entries being changed. We can write it as:

$$\begin{aligned}(W+\Delta)^{1/2} &= U(\Lambda + \widetilde{\Lambda})^{1/2} U^\top \\ &= U \Lambda^{1/2} U^\top + U D U^\top,\end{aligned}$$

where D is a diagonal matrix with only k nonzeros. Hence, we can write it as $W^{1/2} + \overline{\Delta}$ where $\overline{\Delta}$ is rank k, as desired. □

Next, we present a fact regarding the rank change of $\mathsf{A}(W \otimes I)$ and $\mathsf{A}(W^{1/2} \otimes W^{1/2})$, if the matrix W experiences a rank-k change.

Fact B.12. *Suppose $W \in \mathbb{R}^{n \times n}$ undergoes a rank-k change, i.e., $W \leftarrow W + \Delta$ where Δ has rank-k, then*

- *If $\mathsf{B} = \mathsf{A}(W \otimes I)$, then B undergoes a rank-nk change.*
- *If $\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2})$, then B undergoes a rank-$(2nk + k^2)$ change.*

Proof. We prove item by item.

- Suppose $\mathsf{B} = \mathsf{A}(W \otimes I)$, and we have $W + \Delta$, then

  $$\mathsf{A}((W+\Delta) \otimes I) = \mathsf{A}(W \otimes I) + \mathsf{A}(\Delta \otimes I),$$

  note that $\Delta \otimes I$ is of rank nk, so we conclude that B experiences a rank-nk change.
- Suppose $\mathsf{B} = \mathsf{A}(W^{1/2} \otimes W^{1/2})$, then:

  $$\begin{aligned}(W+\Delta)^{1/2} \otimes (W+\Delta)^{1/2} &= (W^{1/2} + \overline{\Delta}) \otimes (W^{1/2} + \overline{\Delta}) \\ &= (W^{1/2} \otimes W^{1/2}) + W^{1/2} \otimes \overline{\Delta} + \overline{\Delta} \otimes W^{1/2} + \overline{\Delta} \otimes \overline{\Delta}\end{aligned}$$

  where the first step is by Lemma B.11. Both $W^{1/2} \otimes \overline{\Delta}$ and $\overline{\Delta} \otimes W^{1/2}$ have rank nk, and the last term has rank k^2. This completes the proof. □

Remark B.13. The above results essentially show that if we give a low rank update to $W \in \mathbb{R}^{n \times n}$ and we wish $(W+\Delta)^{1/2}$ is also a low rank update to $W^{1/2}$, then the update Δ must share the same eigenbasis as W.

We define a function which will be heavily used in Section B.7.

Definition B.14. Let θ and ω be two fixed parameters, which satisfy that $\mathcal{T}_{\mathrm{mat}}(n^2, n, n^2) = n^\theta$ and $\mathcal{T}_{\mathrm{mat}}(n, n, n) = n^\omega$. We define the function $f(a, c)$ as

$$f(a, c) := \frac{c(\theta - \omega - 2) + a(2 + \theta - c\theta - \omega + 2c\omega) - \theta}{a - 1}.$$

B.6. Our Data Structure

In this section, we present our data structure for initialization (Algorithm 2), update (Algorithm 3) and query (Algorithm 4).

Algorithm 2 Initialization and members

```
data structure KROCKERPROJMAINTAIN                                        ▷ Theorem B.15
members
    $W \in \mathbb{R}^{n \times n}$
    $\mathsf{A} \in \mathbb{R}^{m \times n^2}$                                            ▷ Fixed data matrix
    $\mathsf{G} \in \mathbb{R}^{m \times n^2}$                                ▷ Data matrix with eigenbasis
    $M \in \mathbb{R}^{n^2 \times n^2}$                                           ▷ Inverse Hessian
    $\lambda, \widetilde{\lambda} \in \mathbb{R}^n$                         ▷ Eigenvalues and its approximation
    $Q \in \mathbb{R}^{n^2 \times sb}$
    $P \in \mathbb{R}^{n^2 \times sb}$
    $\varepsilon_{\mathrm{mp}} \in (0, 0.1)$                                    ▷ Accuracy parameter
    $a \in [0, \alpha]$                                            ▷ Cutoff threshold
end members

procedure INIT($\mathsf{A} \in \mathbb{R}^{m \times n^2}, W \in \mathbb{R}^{n \times n}$)                            ▷ Lemma B.16
    $\mathsf{A} \leftarrow \mathsf{A}$
    $W \leftarrow W$
    Let $W = U \Lambda U^\top$                    ▷ Compute the spectral decomposition for $W$
    $\mathsf{G} \leftarrow \mathsf{A}(U \otimes U)$
    Generate $R_{1,*}, \ldots, R_{s,*} \in \mathbb{R}^{b \times n^2}$ to be sketching matrices
    $\mathsf{R} \leftarrow [R_{1,*}, \ldots, R_{s,*}] \in \mathbb{R}^{bs \times n^2}$
    $\lambda \leftarrow \lambda$
    $M \leftarrow \mathsf{G}^\top (\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1} \mathsf{G}$
    $Q \leftarrow M(\Lambda^{1/2} \otimes \Lambda^{1/2})(U^\top \otimes U^\top)\mathsf{R}^\top$
    $P \leftarrow (U \otimes U)(\Lambda^{1/2} \otimes \Lambda^{1/2})Q$
end procedure

private:
procedure SOFTTHRESHOLD($\lambda \in \mathbb{R}^n, \lambda^{\mathrm{new}} \in \mathbb{R}^n, r \in \mathbb{N}_+$)
    $y_i \leftarrow \ln \lambda_i^{\mathrm{new}} - \ln \lambda_i$
    Let $\pi : [n] \to [n]$ be a sorting permutation such that $|y_{\pi(i)}| \geq |y_{\pi(i+1)}|$
    while $1.5 \cdot r < n$ and $|y_{\pi(\lceil 1.5 \cdot r \rceil)}| \geq (1 - 1/\log n)|y_{\pi(r)}|$ do
        $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, n)$
    end while
    $\widehat{\lambda}_{\pi(i)} \leftarrow \begin{cases} \lambda^{\mathrm{new}}_{\pi(i)} & i \in \{1, 2, \ldots, r\} \\ \lambda_{\pi(i)} & i \in \{r+1, \ldots, n\} \end{cases}$
    return $\widehat{\lambda}, r$
end procedure
end data structure
```

B.7. Main Results

The goal of this section is to prove Theorem B.15 that, given a sequence of online matrices and queries, there exists a data structure that approximately maintains the projection matrix and the requested matrix-vector product.

Theorem B.15 (Formal verison of Theorem 5.2)**.** *Given a collection of matrices $A_1, \cdots, A_m \in \mathbb{R}^{n \times n}$. We define $B_i = W^{1/2} A_i W^{1/2} \in \mathbb{R}^{n \times n}, \forall i \in [m]$. We define $\mathsf{A} \in \mathbb{R}^{m \times n^2}$ to be the matrix whose i-th row is the vectorization of $A_i \in \mathbb{R}^{n \times n}$ and $\mathsf{B} \in \mathbb{R}^{m \times n^2}$ to be the matrix whose i-th row is the vectorization of $B_i \in \mathbb{R}^{n \times n}$. Let b denote the sketching dimension*

Algorithm 3 UPDATE part of our data structure.

```
data structure KROCKERPROJMAINTAIN
procedure UPDATE($W^{\mathrm{new}}$)    ▷ Lemma B.26
                                        ▷ $W^{\mathrm{new}} = U\mathrm{diag}(\lambda^{\mathrm{new}})U^\top$
    $y_i \leftarrow \ln \lambda_i^{\mathrm{new}} - \ln \lambda_i$
    $r \leftarrow$ the number of indices $i$ such that $|y_i| \geq \varepsilon_{\mathrm{mp}}/2$
    if $r < n^a$ then    ▷ No update
        $\widehat{\lambda} \leftarrow \lambda$
        $V^{\mathrm{new}} \leftarrow W$
        $M^{\mathrm{new}} \leftarrow M$
        $Q^{\mathrm{new}} \leftarrow Q$
        $P^{\mathrm{new}} \leftarrow P$
    else
        $\widehat{\lambda}, r \leftarrow$ SOFTTHRESHOLD($\lambda, \lambda^{\mathrm{new}}$)
        $C \leftarrow \widehat{\lambda} - \lambda$    ▷ Entries updated by $\lambda^{\mathrm{new}}$
        $\Delta \leftarrow \Lambda \otimes C + C \otimes \Lambda + C \otimes C$    ▷ $\Delta \in \mathbb{R}^{n^2 \times n^2}$ is diagonal, has at most $nr$ nonzero entries
        $S \leftarrow \pi([r])$ be the first $r$ indices in the permutation, $\widetilde{S} \leftarrow \{i, i+n, \ldots, i+n(n-1) : i \in S\}$
        Let $M_{\widetilde{S}} \in \mathbb{R}^{n^2 \times nr}$ be the $nr$ columns corresponding to $\widetilde{S}$
        Let $M_{\widetilde{S},\widetilde{S}}$ be the $nr$ rows and columns corresponding to $\widetilde{S}$
        Let $\Delta_{\widetilde{S},\widetilde{S}}$ be the $nr$ entries of $\Delta$
        $M^{\mathrm{new}} \leftarrow M - M_{\widetilde{S}} \cdot (\Delta_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} M_{\widetilde{S}}^\top$
        Regenerate $\mathsf{R}$
        $\Gamma \leftarrow (\Lambda + C)^{1/2} \otimes (\Lambda + C)^{1/2} - \Lambda^{1/2} \otimes \Lambda^{1/2}$
        $Q^{\mathrm{new}} \leftarrow Q + (M^{\mathrm{new}} \cdot \Gamma) \cdot \mathsf{R}^\top + (M^{\mathrm{new}} - M) \cdot (\Lambda^{1/2} \otimes \Lambda^{1/2}) \cdot (U^\top \otimes U^\top) \cdot \mathsf{R}^\top$
        $P^{\mathrm{new}} \leftarrow P + \Gamma^\top \cdot Q^{\mathrm{new}} + (U \otimes U) \cdot (\Lambda^{1/2} \otimes \Lambda^{1/2}) \cdot (Q^{\mathrm{new}} - Q)$
        $V^{\mathrm{new}} \leftarrow U\mathrm{diag}(\widehat{\lambda})U^\top$
    end if
    $\lambda \leftarrow \widehat{\lambda}$
    $Q \leftarrow Q^{\mathrm{new}}$
    $P \leftarrow P^{\mathrm{new}}$
    $M \leftarrow M^{\mathrm{new}}$
    $W \leftarrow V^{\mathrm{new}}$
    $\widetilde{\lambda}_i \leftarrow \begin{cases} \widehat{\lambda}_i & \text{if } |\ln \lambda_i^{\mathrm{new}} - \ln \widehat{\lambda}_i| \leq \varepsilon_{\mathrm{mp}}/2 \\ \lambda_i^{\mathrm{new}} & \text{otherwise} \end{cases}$
    return $U\mathrm{diag}(\widetilde{\lambda})U^\top$
end procedure
end data structure
```

Algorithm 4 Query

```
procedure QUERY($h^{\mathrm{new}}$)    ▷ Lemma B.26
    Let $S$ denote the set of indices such that $|y_i| \geq \varepsilon_{\mathrm{mp}}/2$
    $\widetilde{C} \leftarrow \widetilde{\lambda} - \lambda$
    $\widetilde{\Delta} \leftarrow \Lambda \otimes \widetilde{C} + \widetilde{C} \otimes \Lambda + \widetilde{C} \otimes \widetilde{C}$
    $\widetilde{S} \leftarrow \{i, i+n, \ldots, i+n(n-1) : i \in S\}$
    $\widetilde{\Gamma} \leftarrow (\Lambda + \widetilde{C})^{1/2} \otimes (\Lambda + \widetilde{C})^{1/2} - \Lambda^{1/2} \otimes \Lambda^{1/2}$
    $p_g \leftarrow (U \otimes U) \cdot (\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2}) \cdot (M_{*,\widetilde{S}}) \cdot (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot (Q_{\widetilde{S},l} + M_{\widetilde{S},*} \cdot \widetilde{\Gamma} \cdot (U^\top \otimes U^\top) \cdot R_{*,l}^\top) \cdot R_{*,l} h^{\mathrm{new}}$
    $p_l \leftarrow (U \otimes U) \cdot (\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2}) \cdot (Q_{*,l} + M \cdot \widetilde{\Gamma} \cdot (U^\top \otimes U^\top) \cdot R_{*,l}^\top) R_{*,l} h^{\mathrm{new}} - p_g$
    return $p_l$
end procedure
```

*and let T denote the number of iterations. Let $\varepsilon_{\mathrm{mp}} \in (0, 0.1)$ and $a \in (0, 1)$ be parameters. Let $R_1, \cdots, R_s \in \mathbb{R}^{b \times n^2}$ denote a list of s sketching matrices, and let $\mathsf{R} \in \mathbb{R}^{sb \times n^2}$ denote the batched matrix of these matrices. Then, there is a dynamic maintenance data structure (*KroneckerProjMaintain*) that given a sequence of online matrices*

$$W^{(1)}, \cdots, W^{(T)} \subset \mathbb{R}^{n \times n}; \quad \textit{and} \quad h^{(1)}, \cdots, h^{(T)} \in \mathbb{R}^{n^2}$$

that approximately maintains the projection matrices

$$\mathsf{B}^\top (\mathsf{B}\mathsf{B}^\top)^{-1} \mathsf{B}$$

for positive semidefinite matrices $W \in \mathbb{R}^{n \times n}$ through the following two operations:

- Update(W)*: Output a positive semidefinite matrix $\widetilde{V} \in \mathbb{R}^{n \times n}$ such that for all $i \in [n]$*

  $$(1 - \varepsilon_{\mathrm{mp}}) \cdot \lambda_i(\widetilde{V}) \leq \lambda_i(W) \leq (1 + \varepsilon_{\mathrm{mp}}) \cdot \lambda_i(\widetilde{V})$$

  where $\lambda_i(W)$ denote the i-th eigenvalue of matrix $W \in \mathbb{R}^{n \times n}$.
- Query(h)*: Output $\widetilde{\mathsf{B}}^\top (\widetilde{\mathsf{B}}\widetilde{\mathsf{B}}^\top)^{-1} \widetilde{\mathsf{B}} R_l^\top R_l \cdot h$ for the $\widetilde{B}$ defined by positive semidefinite matrix $\widetilde{V} \in \mathbb{R}^{n \times n}$ outputted by the last call to* Update*.*

The data structure takes $\mathcal{T}_{\mathrm{mat}}(mn, n, n) + \mathcal{T}_{\mathrm{mat}}(m, n^2, m) + \mathcal{T}_{\mathrm{mat}}(m, n^2, s \cdot b) + \mathcal{T}_{\mathrm{mat}}(m, m, s \cdot b) + m^\omega$ time to initialize and if $\mathrm{nnz}(U) = O(n^{1.5+a/2})$, where U is the fixed eigenbasis for W, then each call of Query *takes time*

$$n^{2+b+o(1)} + n^{3+a+o(1)}.$$

Furthermore, if the initial matrix $W^{(0)} \in \mathbb{R}^{n \times n}$ and the (random) update sequence $W^{(1)}, W^{(2)}, \cdots, W^{(T)} \in \mathbb{R}^{n \times n}$ satisfies

$$\sum_{i=1}^{n} (\mathbb{E}[\ln \lambda_i(W^{(k+1)})] - \ln(\lambda_i(W^{(k)})))^2 \leq C_1^2$$

and

$$\sum_{i=1}^{n} (\mathbf{Var}[\ln \lambda_i(W^{(k+1)})])^2 \leq C_2^2$$

with the expectation and variance is conditioned on $\lambda_i(W^{(k)})$ for all $k = 0, 1, \cdots, T-1$. Then, the amortized expected time per call of Update(W) *is*

$$(C_1/\varepsilon_{\mathrm{mp}} + C_2/\varepsilon_{\mathrm{mp}}^2) \cdot \max\{\mathcal{T}_1, \mathcal{T}_2\}.$$

Here, $\mathcal{T}_1 = n^{\omega - 2.5 + f(a,c) + o(1)} + n^{f(a,c) - a/2 + o(1)}$ and $\mathcal{T}_2 = n^{\omega + f(a,c) - 4.5 + o(1)} sb + n^{f(a,c) - (4+a)/2 + o(1)} sb$.

Proof. The correctness of update matrices and queries follows from Lemma B.26. The runtime of initialization follows from Lemma B.16. The runtime of Query follows from Lemma B.17.

For the runtime of update, we note that by Lemma B.24, we pay $O(n^{1+c+f(a,c)+o(1)} g_{n^{1+c}}) = O(r g_r n^{f(a, \log_n r/n)})$ time. Using a potential analysis similar to (Cohen et al., 2019), we have that

$$\sum_{t=1}^{T} r_t g_{r_t} = O(T \cdot (C_1/\varepsilon_{mp} + C_2/\varepsilon_{mp}^2) \cdot \log^{1.5} n \cdot (n^{\omega - 2.5} + n^{-a/2})),$$

this concludes the proof of the amortized running time of update.

Regarding the guarantees of eigenvalues, we can reuse a similar analysis of (Cohen et al., 2019), Section 5.4 and 5.5. □

Next, we present the runtime analysis of the initialization of our data structure.

Lemma B.16 (Initialization Time)**.** *Let s denote the number of sketches. Let b denote the size of each sketch. The* INIT *takes time*

$$mn^\omega + m^\omega + \mathcal{T}_{\mathrm{mat}}(m, m, n^2) + \mathcal{T}_{\mathrm{mat}}(n^2, n^2, sb).$$

Suppose $m = n^2$, then the time becomes

$$m^\omega + \mathcal{T}_{\mathrm{mat}}(m, m, sb).$$

Proof. The initialization contains the following computations:

- Compute the spectral decomposition for $W \in \mathbb{R}^{n \times n}$, takes $O(n^\omega)$ time.
- Compute matrix $\mathsf{G} = \mathsf{A}(U \otimes U)$. We note that A can be viewed as m different $n \times n$ matrices, and we can use the identity $\mathrm{vec}(A_i)(U \otimes U) = \mathrm{vec}(U^\top A_i U)$, hence, it takes $O(mn^\omega)$ time to compute G. Note that the naive computation of G takes $\mathcal{T}_{\mathrm{mat}}(m, n^2, n^2)$ time which is about $mn^{2(\omega-1)}$ time, and it's worse than the time by using the Kronecker product identity.
- Compute $M = \mathsf{G}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}$. We split this computation into several parts:
  - Compute $\mathsf{G}(\Lambda \otimes \Lambda)$. Since Λ is diagonal, this takes $O(mn^2)$ time.
  - Computing $\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top$ takes $\mathcal{T}_{\mathrm{mat}}(m, n^2, m)$ time.
  - Computing the inverse $(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}$ takes $O(m^\omega)$ time.
  - Finally, computing M takes $\mathcal{T}_{\mathrm{mat}}(n^2, m, m)$ time.

  Hence, computing M takes $\mathcal{T}_{\mathrm{mat}}(m, n^2, m) + m^\omega$ time.
- Computing Q takes two steps: Appling sketching $\mathsf{R}^\top$ takes $\mathcal{T}_{\mathrm{mat}}(n^2, n^2, sb)$ time, and computing the product between M and $(\Lambda^{1/2}U^\top \otimes \Lambda^{1/2}U^\top)\mathsf{R}^\top$ takes $\mathcal{T}_{\mathrm{mat}}(n^2, n^2, sb)$ time.
- Computing P takes $\mathcal{T}_{\mathrm{mat}}(n^2, n^2, sb)$ time.

Thus, the total running time is

$$mn^\omega + m^\omega + \mathcal{T}_{\mathrm{mat}}(m, m, n^2) + \mathcal{T}_{\mathrm{mat}}(n^2, n^2, sb).$$

Specifically, for $m = n^2$, the time becomes $m^\omega + \mathcal{T}_{\mathrm{mat}}(m, m, sb)$. □

Next, we present the runtime analysis of the QUERY of our data structure.

Lemma B.17 (Query Time)**.** *Let b denote the size of sketch, then* QUERY *takes time*

$$O(n^{3+a} + n^{2+b}).$$

Proof. First, observe that $\|\widetilde{C}\|_0 \leq n^a$, therefore, $|\widetilde{S}| \leq n^{1+a}$.

We compute time for each term as follows:

- Computing $\widetilde{\Delta}$ takes $O(n^{1+a})$ time.
- Compute $\widetilde{\Gamma}$. Note that this is a diagonal matrix with at most n^{2a} nonzero entries, so it takes $O(n^{2a})$ time to compute.
- Computing p_g, involving the following steps:
  - Computing $R_{*,l}h$, takes $O(n^{2+b})$ time.
  - Computing $R_{*,l}^\top(R_{*,l}h)$ takes $O(n^{2+b})$ time.

  - Compute $(U^\top \otimes U^\top) R_{*,l}^\top (R_{*,l} h)$ takes $O(n^{3+a})$ time since $\mathrm{nnz}(U) = O(n^{1.5+a/2})$ therefore $\mathrm{nnz}(U \otimes U) = \mathrm{nnz}(U)^2 = O(n^{3+a})$.
  - Computing $\widetilde{\Gamma} \cdot (U^\top \otimes U^\top) \cdot R_{*,l}^\top (R_{*,l} h)$ takes $O(n^{2a})$ time since $\widetilde{\Gamma}$ has $O(n^{2a})$ nonzero entries on the diagonal.
  - Computing $M_{\widetilde{S},*} \cdot \widetilde{\Gamma} \cdot (U^\top \otimes U^\top) \cdot R_{*,l}^\top (R_{*,l} h)$ takes $O(n^{3+a})$ time.
  - Similarly, computing $Q_{\widetilde{S},l} R_{*,l} h^{\mathrm{new}}$ takes $O(n^{2+b} + n^{1+a+b})$ time.
  - Computing the inverse $(\widetilde{\Delta}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}$ takes $O(n^{(1+a)\omega})$ time.
  - Computing $(\widetilde{\Delta}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} M_{\widetilde{S},*} \cdot \widetilde{\Gamma} \cdot R_{*,l}^\top (R_{*,l} h)$ takes $O(n^{2+2a})$ time.
  - Computing $M_{*,\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} M_{\widetilde{S},*} \cdot \widetilde{\Gamma} \cdot (U^\top \otimes U^\top) \cdot R_{*,l}^\top (R_{*,l} h)$ takes $O(n^{3+a})$ time.
  - Computing $(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2}) M_{*,\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} M_{\widetilde{S},*} \cdot \widetilde{\Gamma} \cdot R_{*,l}^\top (R_{*,l} h)$, since Λ is a diagonal matrix, it takes $O(n^2)$ time to form this matrix, and multiplying it with a vector of length n^2 takes $O(n^2)$ time.
  - Computing $(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2}) M_{*,\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} M_{\widetilde{S},*} \cdot \widetilde{\Gamma} \cdot R_{*,l}^\top (R_{*,l} h)$ takes $O(n^{3+a})$ by the sparsity of U.

- Compute p_l. Note that the product $R_{*,l}^\top R_{*,l} h^{\mathrm{new}}$ takes $O(n^{2+b})$ time, $(U^\top \otimes U^\top) R_{*,l}^\top R_{*,l} h^{\mathrm{new}}$ takes $O(n^{3+a})$ time, and $\widetilde{\Gamma}(U^\top \otimes U^\top)(R_{*,l}^\top R_{*,l} h^{\mathrm{new}})$ takes $O(n^{2a})$ time due to the sparsity of $\widetilde{\Gamma}$ and the resulting vector contains at most $O(n^{2a})$ nonzero entries. Therefore, computing $M(\widetilde{\Gamma}(U^\top \otimes U^\top) R_{*,l}^\top R_{*,l} h^{\mathrm{new}})$ takes $O(n^{2+2a})$ time.

  Similarly, computing $Q_{*,l} R_{*,l} h^{\mathrm{new}}$ takes $O(n^{2+b})$ time.

  Finally, computing the product between an $n^2 \times n^2$ diagonal matrix and a length n^2 vector takes $O(n^2)$ time, and multiplying the vector with $(U \otimes U)$ takes $O(n^{3+a})$ time.

Overall, it takes

$$O(n^{3+a} + n^{2+b})$$

time to realize this step. □

To adapt an amortized analysis for UPDATE, we introduce several definitions.

Definition B.18. Given $i \in [r]$, we define the weight function as

$$g_i = \begin{cases} n^{-a} & \text{if } i < n^a \\ i^{\frac{\omega-2}{1-a}-1} n^{-\frac{a(\omega-2)}{1-a}} & \text{otherwise} \end{cases}$$

This is a well-known weight function used in (Cohen et al., 2019; Lee et al., 2019) and many subsequent works that use rank-aware amortization for matrix multiplication.

Definition B.19. Let $\theta \in [4,5]$ be the value such that

$$\mathcal{T}_{\mathrm{mat}}(n^2, n, n^2) = n^\theta.$$

Note that when $\omega = 2$, we have that $\theta = 4$.

Lemma B.20. *Let $c \in [0,1)$, then we have*

$$\mathcal{T}_{\mathrm{mat}}(n^2, n^{1+c}, n^2) = n^{c(2\omega-\theta)+\theta}$$

Recall Definition B.14, we define the function $f(a,c)$ as follows:

$$f(a,c) := \frac{c(\theta - \omega - 2) + a(2 + \theta - c\theta - \omega + 2c\omega) - \theta}{a-1}.$$

We will use this function f to simplify our amortization.

Corollary B.21. *We have that*

$$\mathcal{T}_{\mathrm{mat}}(n^2, n^{1+c}, n^2) = n^c g_{n^c} \cdot n^{f(a,c)}$$

Proof. By basic algebraic manipulation, we have

$$\begin{aligned} c(2\omega - \theta) + \theta &= \frac{(c-a)(\omega-2)}{1-a} + \frac{c(\theta - \omega - 2) + a(2 + \theta - c\theta - \omega + 2c\omega) - \theta}{a-1} \\ &= \frac{(c-a)(\omega-2)}{1-a} + f(a,c), \end{aligned}$$

where the second step is by definition of $f(a,c)$. □

Lemma B.22 (Property of $f(a,c)$)**.** *If $\omega = 2$ and $\theta = 4$, then*

$$f(a,c) = 4,$$

for any $a \in (0,1)$ and $c \in (0,1)$.

Proof. Suppose $\omega = 2$ and $\theta = 4$, We can simplify $f(a,c)$ as

$$\begin{aligned} f(a,c) &= \frac{4 + a(4c - 4 + 2 - 4c - 2)}{1-a} \\ &= \frac{4 - 4a}{1-a} \\ &= 4 \end{aligned}$$

□

Remark B.23. Our proof shows that when $\omega = 2$ and therefore $\theta = 4$ which is the common belief of the time complexity of matrix multiplication, the term $f(a,c)$ is always 4. As we will show below, the amortized running time of update is

$$O(rg_r n^4),$$

using a result proved in (Cohen et al., 2019), this means the amortized running time is

$$\widetilde{O}(n^{\omega+1.5} + n^{4-a/2}).$$

This means the amortized time of update is subquadruple, which leads to an improvement over a special class of SDP.

Lemma B.24 (Update Time)**.** *The procedure* UPDATE *takes time $O(rg_r \cdot n^{f(a,c)})$.*

Proof. We note that if the number of indices i with $|y_i| \geq \varepsilon_{\mathrm{mp}}/2$ is at most n^a, then we simply update some variables in the data structure.

In the other case, we perform the following operations. Let $r = n^{1+c}$, then

- Forming Δ in $O(n^{2+c})$ time.
- Adding two $nr \times nr$ matrices takes $O(n^{4+2c})$ time.
- Inverting an $nr \times nr$ matrix takes $O(n^{(2+c)\omega})$ time.
- Computing matrix multiplication of $n^2 \times nr$ matrix with $nr \times n^2$ matrix takes time $O(rg_r \cdot n^{f(a,c)})$.

To compute Q^{new}, note that $(U^\top \otimes U^\top)\mathsf{R}^\top$ can be pre-computed and stored, yielding a matrix of size $n^2 \times sb$.

- Computing $(\Lambda^{1/2} \otimes \Lambda^{1/2}) \cdot (U^\top \otimes U^\top) \cdot \mathsf{R}^\top$ takes $O(sbn^2)$ time since Λ is diagonal.

- Computing $(M^{\mathrm{new}} - M) \cdot (\Lambda^{1/2} \otimes \Lambda^{1/2}) \cdot (U^\top \otimes U^\top) \cdot \mathsf{R}^\top$ can be viewed as a product of four matrices:

  $$n^2 \times nr \to nr \times nr \to nr \times n^2 \to n^2 \times sb,$$

  the time is thus dominated by $\mathcal{T}_{\mathrm{mat}}(n^2, nr, \max\{n^2, sb\})$.

- Computing $(M^{\mathrm{new}} \cdot \Gamma) \cdot \mathsf{R}^\top$. Note that Γ is a diagonal matrix with only nr nonzero entries, therefore $M^{\mathrm{new}} \cdot \Gamma$ can be viewed as selecting and scaling nr columns of M^{new}, which gives a matrix of size $n^2 \times nr$. Multiplying with $\mathsf{R}^\top$ then takes $\mathcal{T}_{\mathrm{mat}}(n^2, nr, sb)$ time.

Therefore, the total running time is $O(rg_r \cdot n^{f(a,c)})$ if $sb \le n^2$. Otherwise, the running time is $\mathcal{T}_{\mathrm{mat}}(n^2, nr, sb)$, which is $O(rg_r \cdot n^{f(a,c)-2} sb)$.

□

Next, we present the Matrix Woodbury Identity regarding the calculation of the inverse of the matrix $(A + UCV)^{-1}$.

Lemma B.25 (Matrix Woodbury Identity). *Let $A \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times k}, C \in \mathbb{R}^{k\times k}$ and $V \in \mathbb{R}^{k\times n}$ and both A and C are non-singular. Then we have*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

Next, we present the proof of the correctness of our UPDATE and QUERY procedure in our data structure.

Lemma B.26 (The correctness of Update and Query). *By line 30 of* UPDATE(W^{new}) *(Algorithm 3), the variables satisfy*

$$\begin{aligned}
M^{\mathrm{new}} &= (U^\top \otimes U^\top)\mathsf{A}^\top(\mathsf{A}(W^{\mathrm{new}} \otimes W^{\mathrm{new}})\mathsf{A}^\top)^{-1}\mathsf{A}(U \otimes U)\\
Q^{\mathrm{new}} &= M^{\mathrm{new}}((\Lambda^{\mathrm{new}})^{1/2}U^\top \otimes (\Lambda^{\mathrm{new}})^{1/2}U^\top)\mathsf{R}^\top\\
P^{\mathrm{new}} &= (U(\Lambda^{\mathrm{new}})^{1/2} \otimes U(\Lambda^{\mathrm{new}})^{1/2})M^{\mathrm{new}}((\Lambda^{\mathrm{new}})^{1/2}U^\top \otimes (\Lambda^{\mathrm{new}})^{1/2}U^\top)\mathsf{R}^\top
\end{aligned}$$

Additionally, the output of QUERY(h^{new}) *satisfies*

$$p_l^{\mathrm{new}} = \widetilde{P} \cdot R_{*,l}^\top R_{*,l} \cdot h^{\mathrm{new}}$$

where $\widetilde{P} = \widetilde{\mathsf{B}}^\top(\widetilde{\mathsf{B}}\widetilde{\mathsf{B}})^{-1}\widetilde{\mathsf{B}}$ and $\widetilde{\mathsf{B}}$ is defined based on $\widetilde{V}$ which is outputted by UPDATE(W).

Remark B.27. We generalize the Lemma E.3 in (Song & Yu, 2021) from the diagonal W case to the positive semidefinite W case.

Proof. **Correctness for M.** The correctness follows from Lemma B.28.

Correctness for Q.

We have

$$\begin{aligned}
& Q^{\mathrm{new}}\\
&= Q + (M^{\mathrm{new}} \cdot \Gamma) \cdot \mathsf{R}^\top + (M^{\mathrm{new}} - M) \cdot (\Lambda^{1/2}U^\top \otimes \Lambda^{1/2}U^\top) \cdot \mathsf{R}^\top\\
&= M(\Lambda^{1/2}U^\top \otimes \Lambda^{1/2}U^\top)\mathsf{R}^\top + M^{\mathrm{new}}((\Lambda + C)^{1/2}U^\top \otimes (\Lambda + C)U^\top)\mathsf{R}^\top - M^{\mathrm{new}}(\Lambda^{1/2}U^\top \otimes \Lambda^{1/2}U^\top)\mathsf{R}^\top\\
&+ (M^{\mathrm{new}} - M) \cdot (\Lambda^{1/2}U^\top \otimes \Lambda^{1/2}U^\top) \cdot \mathsf{R}^\top\\
&= M(\Lambda^{1/2}U^\top \otimes \Lambda^{1/2}U^\top)\mathsf{R}^\top + M^{\mathrm{new}}((\Lambda + C)^{1/2}U^\top \otimes (\Lambda + C)^{1/2}U^\top)\mathsf{R}^\top - M(\Lambda^{1/2}U^\top \otimes \Lambda^{1/2}U^\top)\mathsf{R}^\top\\
&= M^{\mathrm{new}}((\Lambda + C)^{1/2}U^\top \otimes (\Lambda + C)^{1/2}U^\top)\mathsf{R}^\top
\end{aligned}$$

where the first step follows from line 24 in Algorithm 3, the second step follows from definition of Q, the third step follows from re-organizing terms, and the last step follows from cancelling the first term and the last term.

Correctness for P.

$$\begin{aligned}P^{\mathrm{new}} &= P + \Gamma^\top \cdot Q^{\mathrm{new}} + (U\Lambda^{1/2} \otimes U\Lambda^{1/2}) \cdot (Q^{\mathrm{new}} - Q)\\ &= (U\Lambda^{1/2} \otimes U\Lambda^{1/2})Q + (U(\Lambda + C)^{1/2} \otimes U(\Lambda + C)^{1/2}) \cdot Q^{\mathrm{new}} - (U\Lambda^{1/2} \otimes U\Lambda^{1/2})Q^{\mathrm{new}}\\ &+ (U\Lambda^{1/2} \otimes U\Lambda^{1/2}) \cdot (Q^{\mathrm{new}} - Q)\\ &= (U(\Lambda + C)^{1/2})Q^{\mathrm{new}}\end{aligned}$$

where the first step follows from line 23 in Algorithm 3, the second step follows from definition of P, and the last step follows from merging the terms.

Correctness of QUERY.

We first unravel p_g^{new}:

$$\begin{aligned}&(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(Q_{\widetilde{S},l} + M_{\widetilde{S},*}\widetilde{\Gamma}(U^\top \otimes U^\top)R^\top_{*,l})\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(M_{\widetilde{S},*}(\Lambda^{1/2} \otimes \Lambda^{1/2})(U^\top \otimes U^\top)R^\top_{*,l} + M_{\widetilde{S},*}\widetilde{\Gamma}(U^\top \otimes U^\top)R^\top_{*,l})\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(M_{\widetilde{S},*})(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(U^\top \otimes U^\top)R^\top_{*,l}.\end{aligned}$$

Hence,

$$\begin{aligned}&p_g^{\mathrm{new}}\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(Q_{\widetilde{S},l} + M_{\widetilde{S},*}\widetilde{\Gamma}(U^\top \otimes U^\top)R^\top_{*,l})R_{*,l}h^{\mathrm{new}}\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(M_{\widetilde{S},*})(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(U^\top \otimes U^\top)R^\top_{*,l}R_{*,l}h^{\mathrm{new}}.\end{aligned}$$

To see p_l^{new}, it suffices to show the following:

$$\begin{aligned}&(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(Q_{*,l} + M\widetilde{\Gamma}(U^\top \otimes U^\top)R^\top_{*,l})\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M(\Lambda^{1/2} \otimes \Lambda^{1/2})(U^\top \otimes U^\top)R^\top_{*,l} + M\widetilde{\Gamma}(U^\top \otimes U^\top)R^\top_{*,l})\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})M(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(U^\top \otimes U^\top)R^\top_{*,l}.\end{aligned}$$

To stitch everything together, we notice that

$$\begin{aligned}&M - (M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(M_{\widetilde{S},*})\\ =\ &\mathsf{G}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G} - \mathsf{G}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}_{*,\widetilde{S}}(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}G_{\widetilde{S},*}\mathsf{G}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}\\ =\ &\mathsf{G}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G} - \mathsf{G}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}_{*,\widetilde{S}}(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + \mathsf{G}^\top_{\widetilde{S},*}(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}_{*,\widetilde{S}})^{-1}\mathsf{G}^\top_{\widetilde{S},*}(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}\\ =\ &\mathsf{G}^\top((\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1} - (\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}_{*,\widetilde{S}}(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + \mathsf{G}^\top_{\widetilde{S},*}(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}_{*,\widetilde{S}})^{-1}\mathsf{G}^\top_{\widetilde{S},*}(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1})\mathsf{G}\\ =\ &\mathsf{G}^\top(\mathsf{G}(\widetilde{\Lambda} \otimes \widetilde{\Lambda})\mathsf{G}^\top)^{-1}\mathsf{G}.\end{aligned}$$

Therefore,

$$\begin{aligned}&p_l^{\mathrm{new}}\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})M(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(U^\top \otimes U^\top)R^\top_{*,l}R_{*,l}h^{\mathrm{new}} - p_g^{\mathrm{new}}\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M - (M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(M_{\widetilde{S},*}))(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(U^\top \otimes U^\top)R^\top_{*,l}R_{*,l}h^{\mathrm{new}}\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(M - (M_{*,\widetilde{S}})(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}(M_{\widetilde{S},*}))(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(U^\top \otimes U^\top)R^\top_{*,l}R_{*,l}h^{\mathrm{new}}\\ =\ &(U \otimes U)(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})\mathsf{G}^\top(\mathsf{G}(\widetilde{\Lambda} \otimes \widetilde{\Lambda})\mathsf{G}^\top)^{-1}\mathsf{G}(\widetilde{\Lambda}^{1/2} \otimes \widetilde{\Lambda}^{1/2})(U^\top \otimes U^\top)R^\top_{*,l}R_{*,l}h^{\mathrm{new}}\\ =\ &\widetilde{P}R^\top_{*,l}R_{*,l}h^{\mathrm{new}}\end{aligned}$$

as desired. □

The following corollary uses the assumption that all W we received share the same eigenspace, and presents the formula of matrix M^{new} as a function of G and Λ^{new}.

Lemma B.28. *Let $\mathsf{G} = \mathsf{A}(U \otimes U)$. Suppose that $W = U\Lambda U^\top$ and we receive $W^{\mathrm{new}} = W + UCU^\top$ where $C \in \mathbb{R}^{n\times n}$ but only has k nonzero entries.*

Then we have

$$\mathsf{G}^\top (\mathsf{G}(\Lambda^{\mathrm{new}} \otimes \Lambda^{\mathrm{new}})\mathsf{G}^\top)^{-1}\mathsf{G} = M^{\mathrm{new}}.$$

Proof. We prove via matrix Woodbury identity:

$$\begin{aligned}
& \mathsf{G}^\top (\mathsf{G}(\Lambda^{\mathrm{new}} \otimes \Lambda^{\mathrm{new}})\mathsf{G}^\top)^{-1}\mathsf{G} \\
= {} & \mathsf{G}^\top (\mathsf{G}((\Lambda + C) \otimes (\Lambda + C))\mathsf{G}^\top)^{-1}\mathsf{G} \\
= {} & \mathsf{G}^\top (\mathsf{G}((\Lambda \otimes \Lambda) + \Delta)\mathsf{G}^\top)^{-1}\mathsf{G} \\
= {} & \mathsf{G}^\top (\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G} \\
& - \mathsf{G}^\top ((\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}_{*,\widetilde{S}}(\Delta^{-1} + \mathsf{G}_{*,\widetilde{S}}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1}\mathsf{G}_{*,\widetilde{S}})^{-1}\mathsf{G}_{*,\widetilde{S}}^\top(\mathsf{G}(\Lambda \otimes \Lambda)\mathsf{G}^\top)^{-1})\mathsf{G} \\
= {} & M - M_{*,\widetilde{S}}(\Delta^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1}M_{*,\widetilde{S}}^\top \\
= {} & M^{\mathrm{new}}.
\end{aligned}$$

where the first step follows from the definition of $\Lambda^{\mathrm{new}} = \Lambda + C$, the second step follows from the linearity of Kronecker product calculation and the definition of $\Delta := C \otimes \Lambda + \Lambda \otimes C + C \otimes C$, the third step follows from matrix Woodbury identity, the fourth step follows from plugging in the definitions, and the final step follows from the definition of M^{new} in the algorithm. □

C. Differential Privacy

This section is organized as follows: We present the preliminaries on coordinate-wise embedding in Section C.1. We present the preliminaries on differential privacy in Section C.2. We present the formal results on the data structure with norm guarantee in Section C.3.

C.1. Coordinate-wise Embedding

C.1.1. Definition and Results

First, we state the definition of coordinate-wise embedding:

Definition C.1 (Coordinate-wise embedding (Song & Yu, 2021))**.** We say a random matrix $R \in \mathbb{R}^{b\times n}$ from a family Π satisfies (α, β, δ)-coordinatewise embedding(CE) property if for any two fixed vector $g, h \in \mathbb{R}^n$, we have the following:

1. $\mathbb{E}_{R\sim\Pi}[g^\top R^\top Rh] = g^\top h$.
2. $\mathbb{E}_{R\sim\Pi}[(g^\top R^\top Rh)^2] \le (g^\top h)^2 + \frac{\alpha}{b}\|g\|_2^2\|h\|_2^2$.
3. $\Pr_{R\sim\Pi}[|g^\top R^\top Rh - g^\top h| \ge \frac{\beta}{\sqrt{b}}\|g\|_2\|h\|_2] \le \delta$.

In (Song & Yu, 2021), they had proved that for certain choices of α, β, δ, the coordinate-wise embedding properties are existing. Additionally, we give the (α, β, δ)-guarantee for some commonly used sketching matrices in Section C.1.2.

Next, we present the data structure whose output satisfied coordinate-wise embedding property.

Lemma C.2 (Simple coordinate-wise embedding data structure)**.** *There exists a randomized data structure such that, for any oblivious sequence $\{g_0, \dots, g_{T-1}\} \in (\mathbb{R}^n)^T$ and $\{h_0, \dots, h_{T-1}\} \in (\mathbb{R}^n)^T$ and parameters α, β, δ, with probability at least $1 - T\delta$, we have for any $t \in \{0, \dots, T-1\}$, each pair of vectors (g_t, h_t), satisfies (α, β, δ)-coordinatewise embedding property (Def. C.1).*

Proof. The algorithm is simply picking an (α, β, δ)-coordinate-wise embedding matrix R and apply it to g_t and h_t. □

Note that (α, β, δ)-CE gives three guarantees: expectation, variance and high probability. For our applications, we focus on the high probability part and parameters β, when coupled with the sketching dimension b, gives us the approximation factor γ.

At first, we present the approximation factor γ of given *vectors* g and h.

Lemma C.3. *Let $R \in \mathbb{R}^{b \times n}$ satisfies (α, β, δ)-coordinate-wise embedding property, then given vectors $g, h \in \mathbb{R}^n$, then we have, with probability at least $1 - \delta$,*

$$\begin{aligned} |\langle Rg, Rh \rangle| &= |\langle g, h \rangle| \pm \gamma \|g\|_2 \|h\|_2, \\ \langle Rg, Rh \rangle^2 &= \langle g, h \rangle^2 \pm \gamma \|g\|_2^2 \|h\|_2^2. \end{aligned}$$

where $\gamma = \frac{\beta}{\sqrt{b}}$.

Proof. By property 3 of coordinate-wise embedding (Def. C.1), we have that, with probability $1 - \delta$,

$$|\langle Rg, Rh \rangle - \langle g, h \rangle| \leq \frac{\beta}{\sqrt{b}} \|g\|_2 \|h\|_2.$$

Note that for any two real numbers a and b, we have

$$||a| - |b|| \leq |a - b|,$$

therefore,

$$\begin{aligned} ||\langle Rg, Rh \rangle| - |\langle g, h \rangle|| &\leq |\langle Rg, Rh \rangle - \langle g, h \rangle| \\ &\leq \frac{\beta}{\sqrt{b}} \|g\|_2 \|h\|_2. \end{aligned}$$

Suppose $|\langle Rg, Rh \rangle| \geq |\langle g, h \rangle|$, then we have:

$$|\langle Rg, Rh \rangle| \leq |\langle g, h \rangle| + \frac{\beta}{\sqrt{b}} \|g\|_2 \|h\|_2.$$

Square both sides of the inequality yields:

$$\begin{aligned} \langle Rg, Rh \rangle^2 &\leq \langle g, h \rangle^2 + \frac{\beta^2}{b} \|g\|_2^2 \|h\|_2^2 + \frac{2\beta}{\sqrt{b}} |\langle g, h \rangle| \|g\|_2 \|h\|_2 \\ &\leq \langle g, h \rangle^2 + \frac{3\beta}{\sqrt{b}} \|g\|_2^2 \|h\|_2^2. \end{aligned}$$

where the last step follows from Cauchy-Schwartz inequality that $|\langle g, h \rangle| \leq \|g\|_2 \|h\|_2$ and the property that $\beta/b \in (0, 1)$.

Suppose $|\langle Rg, Rh \rangle| \leq |\langle g, h \rangle|$, then we have:

$$|\langle Rg, Rh \rangle| \geq |\langle g, h \rangle| - \frac{\beta}{\sqrt{b}} \|g\|_2 \|h\|_2.$$

Again, we square both sides:

$$\begin{aligned}\langle Rg, Rh\rangle^2 &\geq \langle g, h\rangle^2 + \frac{\beta^2}{b}\|g\|_2^2\|h\|_2^2 - \frac{2\beta}{\sqrt{b}}|\langle g, h\rangle|\|g\|_2\|h\|_2 \\ &\geq \langle g, h\rangle^2 + \frac{\beta^2}{b}\|g\|_2^2\|h\|_2^2 - \frac{2\beta}{\sqrt{b}}\|g\|_2^2\|h\|_2^2 \\ &\geq \langle g, h\rangle^2 - \frac{\beta}{\sqrt{b}}\|g\|_2^2\|h\|_2^2 \\ &\geq \langle g, h\rangle^2 - \frac{3\beta}{\sqrt{b}}\|g\|_2^2\|h\|_2^2,\end{aligned}$$

where the second step follows from Cauchy-Schwartz property that $|\langle g, h\rangle| \leq \|g\|_2\|h\|_2$, and the third step follows from the property that $\beta/b \in (0, 1)$.

Then, by choosing $\beta = \beta/3$, we get desired result. □

Then, we present the approximation factor γ of given matrix G and vector h.

Corollary C.4. *Let $R \in \mathbb{R}^{b\times n}$ satisfies (α, β, δ)-coordinate-wise embedding property, then given matrix $G \in \mathbb{R}^{n\times n}$ and $h \in \mathbb{R}^n$, then we have, with probability at least $1 - \delta$,*

$$\|GR^\top Rh\|_2^2 = \|Gh\|_2^2 \pm \frac{\beta}{\sqrt{b}}\|G\|_F^2\|h\|_2^2.$$

Proof. We apply Lemma C.3 to each row i of G. Use g_i to denote i-th row of G, we have that:

$$\langle g_i, h\rangle^2 - \frac{\beta}{\sqrt{b}}\|g_i\|_2^2\|h\|_2^2 \leq \langle Rg_i, Rh\rangle^2 \leq \langle g_i, h\rangle^2 + \frac{\beta}{\sqrt{b}}\|g_i\|_2^2\|h\|_2^2,$$

Observe that we have the following properties:

$$\begin{aligned}\sum_{i=1}^n \langle g_i, h\rangle^2 &= \|Gh\|_2^2, \\ \sum_{i=1}^n \|g_i\|_2^2\|h\|_2^2 &= \|G\|_F^2\|h\|_2^2, \\ \sum_{i=1}^n \langle Rg_i, Rh\rangle^2 &= \|GR^\top Rh\|_2^2.\end{aligned}$$

Then, summing over all $i \in [n]$ concludes the proof. □

Remark C.5. The above two results show that, given that the data structure satisfies (α, β, δ)-coordinate-wise embedding property, the same data structure has a $\gamma = \frac{\beta}{\sqrt{b}}$-approximation guarantee.

C.1.2. Guarantee on Several Well-known Sketching Matrices

In this section, we present the definitions of several commonly used sketching matrices, and their parameters α, β, δ when acting as the matrices for coordinate-wise embedding.

We give definitions of the sketching matrices below, starting with the definition of random Gaussian matrix.

Definition C.6 (Random Gaussian Matrix, folklore)**.** Let $R \in \mathbb{R}^{b\times n}$ denote a random Gaussian matrix such that all entries are i.i.d. sampled from $\mathcal{N}(0, 1/b)$.

Next, we present the definition of subsampled randomized Hadamard/Fourier transform(SRHT) matrix, which can be applied efficiently via fast Fourier transform (FFT).

sketching matrix	α	β
Random Gaussian (Definition C.6)	$O(1)$	$O(\log^{1.5}(n/\delta))$
SRHT (Definition C.7)	$O(1)$	$O(\log^{1.5}(n/\delta))$
AMS (Definition C.8)	$O(1)$	$O(\log^{1.5}(n/\delta))$
Count-Sketch (Definition C.9)	$O(1)$	$O(\sqrt{b}\log(1/\delta))$ or $O(1/\sqrt{\delta})$
Sparse Embedding (Definition C.11)	$O(1)$	$O(\sqrt{b/s}\log^{1.5}(n/\delta))$

Table 1: Summary for different sketching matrices. (Table 1 in (Song & Yu, 2021))

Definition C.7 (Subsampled Randomized Hadamard/Fourier Transform(SRHT) Matrix (Lu et al., 2013))**.** We use $R \in \mathbb{R}^{b\times n}$ to denote a subsampled randomized Hadamard transform matrix[3]. Then R has the form

$$R = \sqrt{\frac{n}{b}} \cdot SHD,$$

where $S \in \mathbb{R}^{b\times n}$ is a random matrix whose rows are b uniform samples (without replacement) from the standard basis of $\mathbb{R}^n$, $H \in \mathbb{R}^{n\times n}$ is a normalized Walsh-Hadamard matrix and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose diagonal elements are i.i.d. $\{-1, +1\}$ random variables.

Next, let us present the definition of AMS sketch matrix which is generated by 4-wise hash functions.

Definition C.8 (AMS Sketch Matrix (Alon et al., 1999))**.** Suppose that $g_1, g_2, \cdots, g_b$ be b random hash functions picking from a 4-wise independent hash family

$$\mathcal{G} = \Big\{ g : [n] \to \{-\frac{1}{\sqrt{b}}, +\frac{1}{\sqrt{b}}\} \Big\}.$$

Then $R \in \mathbb{R}^{b\times n}$ is a AMS sketch matrix if we set $R_{i,j} = g_i(j)$.

Next, we present the definition of count sketch matrix, which is also generated by hash functions.

Definition C.9 (Count-Sketch Matrix (Charikar et al., 2002))**.** Suppose that $h : [n] \to [b]$ is a random 2-wise independent hash function

Assume that $\sigma : [n] \to \{-1, +1\}$ is a random 4-wise independent hash function.

Then we say $R \in \mathbb{R}^{b\times n}$ is a count-sketch matrix if the matrix satisfy that $R_{h(i),i} = \sigma(i)$ for all $i \in [n]$ and zero everywhere else.

Next, we present one definition of sparse embedding matrix.

Definition C.10 (Sparse Embedding Matrix I (Nelson & Nguyên, 2013))**.** Let $R \in \mathbb{R}^{b\times n}$ be a sparse embedding matrix with parameter s if each column of R has exactly s non-zero elements being $\pm 1/\sqrt{s}$ uniformly at random. Note that those locations are picked uniformly at random without replacement (and independent across columns) [4].

Finally, we present another equivalent definition of sparse embedding matrix.

Definition C.11 (Sparse Embedding Matrix II (Nelson & Nguyên, 2013))**.** Suppose that $h : [n] \times [s] \to [b/s]$ is a random 2-wise independent hash function.

Assume that $\sigma : [n] \times [s] \to \{-1, 1\}$ be 4-wise independent.

We use $R \in \mathbb{R}^{b\times n}$ to represent a sparse embedding matrix II with parameter s if we set $R_{(j-1)b/s+h(i,j),i} = \sigma(i,j)/\sqrt{s}$ for all $(i, j) \in [n] \times [s]$ and zero everywhere else.

[3] In this case, we require $\log n$ to be an integer.

[4] The signs need only be $O(\log d)$-wise independent. Each column can be specified by a $O(\log d)$-wise independent permutation. The seeds specifying the permutations in different columns need only be $O(\log d)$-wise independent.

C.2. Differential Privacy Background

In this section, we first present the definition of privacy (Dwork, 2006). Then we also present the simple composition theorem from (Dwork & Roth, 2014), the standard advanced composition theorem from (Dwork et al., 2010), and amplification via sampling theorem from (Bun et al., 2015). After that, we present the generalization guarantee on (ε, δ)-DP algorithms (Dwork et al., 2015), and the private median algorithm.

Here, we present the definition of (ε, δ)-differential privacy. The intuition of this definition is that any particular row of the dataset cannot have large impact on the output of the algorithm.

Definition C.12 (Differential Privacy)**.** We say a randomized algorithm $\mathcal{A}$ is (ε, δ)-differentially private if for any two databases S and S' that differ only on one row and any subset of outputs T, the following

$$\Pr[\mathcal{A}(S) \in T] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') \in T] + \delta,$$

holds. Note that, here the probability is over the randomness of $\mathcal{A}$.

Next, we present the simple composition theorem, where the combination of differentially-private output has a privacy guarantee as the sum of their privacy guarantee.

Theorem C.13 (Simple Composition (Corollary 3.15 of (Dwork & Roth, 2014)))**.** *Let $\varepsilon_1, \ldots, \varepsilon_k \in (0, 1]$, if each $\mathcal{A}_i$ is ε_i-differentially private, then their combination, defined to be $\mathcal{A}_{[k]} = \mathcal{A}_k \circ \ldots \circ \mathcal{A}_1$ is $\sum_{s=1}^{k} \varepsilon_s$-differentially private.*

The above composition theorem gives a linear growth on the privacy guarantee. Next, the following advanced composition tool (Theorem C.14) demonstrates that the privacy parameter ε_0 need not grow linearly in k. However it only requires roughly $\sqrt{k}$.

Theorem C.14 (Advanced Composition, see (Dwork et al., 2010))**.** *Given three parameters $\varepsilon \in (0, 1]$, $\delta_0 \in (0, 1]$ and $\delta \in [0, 1]$. If $\mathcal{A}_1, \cdots, \mathcal{A}_k$ are each (ε, δ)-*DP *algorithms, then the k-fold adaptive composition $\mathcal{A}_k \circ \cdots \circ \mathcal{A}_1$ is $(\varepsilon_0, \delta_0 + k\delta)$-*DP *where*

$$\varepsilon_0 = \sqrt{2k \ln(1/\delta_0)} \cdot \varepsilon + 2k\varepsilon^2$$

Next, we present the amplification theorem, where we can boost the privacy guarantee by subsampling a subset of the database of the original DP algorithm as the input.

Theorem C.15 (Amplification via sampling (Lemma 4.12 of (Bun et al., 2015)[5]))**.** *Suppose that $\varepsilon \in (0, 1]$ is an accuracy parameter. Let $\mathcal{A}$ denote an ε-*DP *algorithm. Let S denote a dataset with $|S|$.*

Suppose that $\mathcal{A}'$ is the algorithm that,

- *constructs a database $T \subset S$ by sub-sampling with repetition $k \leq n/2$ rows from S,*
- *returns $\mathcal{A}(T)$.*

Finally, we have

$$\mathcal{A}' \quad \text{is} \quad (\frac{6k}{n}\varepsilon)\text{-DP}.$$

Next, we present the generalization theorem (see (Dwork et al., 2015), (Bassily et al., 2016)) which gives the accuracy guarantee of our DP algorithm on *adaptive* inputs.

Theorem C.16 (Generalization of Differential Privacy (DP))**.** *Given two accuracy parameters $\varepsilon \in (0, 1/3)$ and $\delta \in (0, \varepsilon/4)$. Suppose that the parameter t satisfy that $t \geq \varepsilon^{-2} \log(2\varepsilon/\delta)$.*

We $\mathcal{D}$ to represent a distribution on a domain X. Suppose $S \sim \mathcal{D}^t$ is a database containing t elements sampled independently from $\mathcal{D}$. Let $\mathcal{A}$ be an algorithm that, given any database S of size t, outputs a predicate $h : X \to \{0, 1\}$.

[5] (Bun et al., 2015) gives a more general bound, and uses (ε, δ)-DP.

If $\mathcal{A}$ is (ε, δ)-DP, then the empirical average of h on sample S, i.e.,

$$h(S) = \frac{1}{|S|} \sum_{x \in S} h(x),$$

and h's expectation is taken over underlying distribution $\mathcal{D}$, i.e.,

$$h(\mathcal{D}) = \mathop{\mathbb{E}}_{x \sim \mathcal{D}} [h(x)]$$

are within 10ε with probability at least $1 - \delta/\varepsilon$:

$$\Pr_{S \sim \mathcal{D}^t, h \leftarrow \mathcal{A}(S)} \Big[\big| h(S) - h(\mathcal{D}) \big| \geq 10\varepsilon \Big] \leq \delta/\varepsilon.$$

Finally, we present the private median algorithm which has a differentially private output that is close to the median of the database.

Theorem C.17 (Private Median)**.** *Given two accuracy parameter $\varepsilon \in (0,1)$ and $\beta \in (0,1)$. We use X to represent a finite domain with total order. Let $\Gamma = O(\varepsilon^{-1} \log(|X|/\beta))$.*

Then there is an $(\varepsilon, 0)$-DP algorithm $\textsc{PrivateMedian}_{\varepsilon,\beta}$ that, given a database $S \in X^*$ in

$$O(|S| \cdot \varepsilon^{-1} \log^3(|X|/\beta) \cdot \mathrm{poly} \log |S|)$$

time outputs an element $x \in X$ (possibly $x \notin S$) such that, with probability $1 - \beta$, there are

- *$\geq |S|/2 - \Gamma$ elements in S that are $\geq x$,*
- *$\geq |S|/2 - \Gamma$ elements in S that are $\leq x$.* [6]

C.3. Data Structure with Norm Guarantee

In this section, we present the definition of γ-approximation and the guarantee of the norm estimation algorithm against an adaptive adversary.

Definition C.18. Given matrix $G \in \mathbb{R}^{n \times n}$ and $h \in \mathbb{R}^n$, we define function $f : \mathbb{R}^{n \times n} \times \mathbb{R}^n \to \mathbb{R}$ as

$$f(G, h) = \|G R^\top R h\|_2^2,$$

where $R \in \mathbb{R}^{b \times n}$ satisfies (α, β, δ)-coordinate-wise embedding property (Def. C.1). We say $f(G, h)$ is a γ-approximation of $\|Gh\|_2^2$ if

$$\|Gh\|_2^2 - \gamma \|G\|_F^2 \|h\|_2^2 \leq f(G, h) \leq \|Gh\|_2^2 + \gamma \|G\|_F^2 \|h\|_2^2.$$

The goal of this section is to prove the norm estimation guarantee(Theorem C.19) that, when given an approximation algorithm against an *oblivious* adversary, we can adapt it to an approximation algorithm against an *adaptive* adversary with slightly worse approximation guarantee.

Theorem C.19 (Reduction to Adaptive Adversary: Norm Estimation.)**.** *Given two parameters $\delta > 0, \alpha > 0$. Suppose $\mathcal{U} := [-U, -\frac{1}{U}] \cup \{0\} \cup [\frac{1}{U}, U]$ for $U > 1$. We define function f such that maps elements from domain $G \times H$ to an element in $\mathcal{U}$.*

Assume there is a dynamic algorithm $\mathcal{A}$ against an oblivious adversary that, given an initial data point $x_0 \in X$ and T updates, the following conditions are holding:

- *The preprocessing time is $\mathcal{T}_{\mathrm{prep}}$.*

[6] The runtime dependency on domain size can be improved by other papers, if we have relatively small domain, we can use this theorem.

- *The update time per round is $\mathcal{T}_{\mathrm{update}}$.*
- *The query time is $\mathcal{T}_{\mathrm{query}}$ and, with probability $\geq 9/10$, the answer $f(G_t, h_t)$ is a γ-approximation of $\|G_t h_t\|_2^2$ for every t, i.e.,*

$$\|G_t h_t\|_2^2 - \gamma\|G_t\|_F^2\|h_t\|_2^2 \leq f(G_t, h_t) \leq \|G_t h_t\|_2^2 + \gamma\|G_t\|_F^2\|h_t\|_2^2$$

Then, there is a dynamic algorithm $\mathcal{B}$ against an adaptive adversary, with probability at least $1-\delta$, obtains an $(\alpha+\gamma+\alpha\gamma)$-approximation of $\|G_t h_t\|_2^2$, guarantees the following:

- *The preprocessing time is $\widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{prep}})$.*
- *The update time per round is $\widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{update}})$.*
- *The per round query time is $\widetilde{O}(\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{query}})$ and, with probability $\geq 9/10$, the answer u_t is an $(\alpha+\gamma+\alpha\gamma)$-norm approximation of $\|G_t h_t\|_2^2$ for every t, i.e.*

$$\|G_t h_t\|_2^2 - (\alpha+\gamma+\alpha\gamma)\|G_t\|_F^2\|h_t\|_2^2 \leq u_t \leq \|G_t h_t\|_2^2 + (\alpha+\gamma+\alpha\gamma)\|G_t\|_F^2\|h_t\|_2^2$$

where $\|G\|_F$ denote the Frobenius norm of matrix G.

Moreover, $\mathcal{B}$ undergoes T updates in $\widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta})\cdot(t_p+\mathcal{T}_{\mathrm{update}}) + T\log(\frac{\log U}{\alpha\delta})\cdot\mathcal{T}_{\mathrm{query}})$ total update time, and hence $\mathcal{B}$ has an amortized running time of $\widetilde{O}((\mathcal{T}_{\mathrm{prep}}+\mathcal{T}_{\mathrm{update}})\log(\frac{\log U}{\alpha\delta})/\sqrt{T} + \log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{query}})$. The $\widetilde{O}$, hides $\mathrm{poly}\log(T)$ *factors.*

Proof. **Algorithm** $\mathcal{B}$. We first describe the algorithm $\mathcal{B}$.

- Suppose $L = \widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta}))$. Let us initialize L copies of $\mathcal{A}$. Let us call them $\mathcal{A}^{(1)}, \cdots, \mathcal{A}^{(L)}$. Suppose the initial data point x_0.
- For time step $t = 1, \ldots, T$:
  - We update each copy of $\mathcal{A}$ by (G_t, h_t).
  - We independently uniformly sample $q = \widetilde{O}(\log(\frac{\log U}{\alpha\delta}))$ indices and we denote this index set as S_t.
  - For every $l \in S_t$, we query $\mathcal{A}^{(l)}$ and let $\widehat{f}_t^{(l)}$ denote its output for current update. For these nonzero output, we round them to the nearest power of $(1+\alpha)$, and denote it by $\widetilde{f}_t^{(l)}$. To be specific, $\widetilde{f}_t^{(l)}$ satisfies the following:

$$\widetilde{f}_t^{(l)} = \frac{\widehat{f}_t^{(l)}}{|\widehat{f}_t^{(l)}|}(1+\alpha)^{\lceil \log_{(1+\alpha)}|\widehat{f}_t^{(l)}|\rceil}$$

  - Finally, we aggregate the rounded output $\widetilde{f}_t^{(l)}$ by PRIVATEMEDIAN in Lemma C.17, and then output the differentially private norm estimate u_t.

where we use $\widetilde{O}$ to hide the $\mathrm{poly}\log T$ factor.

Next, let us present the formal algorithm of the above statement:

C.3.1. PROOF OVERVIEW

In this section, we present the choice of our parameters, the informal and formal presentation of the proposed algorithm, and the intuition behind our implementation of differential privacy.

Parameters Here, we choose the parameters of the algorithm as follows:

$$\varepsilon_{\mathsf{pm}} = \frac{1}{4}, \quad \delta_0 = \delta/(4T), \quad q = \widetilde{O}(\log(\frac{\log U}{\alpha\delta_0})) \quad \text{and} \quad L = \widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta_0})). \tag{1}$$

Algorithm 5 Our Norm Estimation Algorithm.

```
procedure ReductionAlgorithm($T, U, \alpha, \delta$)                ▷ Theorem C.19
    $\delta_0 \leftarrow \delta/2T$
    $L \leftarrow \widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta_0}))$
    for $l \in [L]$ do
        Initialize $\mathcal{A}^{(l)}$ with the initial data point $x_0$
    end for
    for $t = 1 \to T$ do
        for $l \in [L]$ do
            $\mathcal{A}^{(l)}$.Update$(G_t, h_t)$.
        end for
        $q \leftarrow \widetilde{O}(\log(\frac{\log U}{\alpha\delta_0}))$
        We independently uniformly sample $q$ indices as the index set $S_t \subset [L]$.
        for $l \in S_t$ do
            $\widehat{f}_t^{(l)} \leftarrow \mathcal{A}^{(l)}$.Query()
        end for
        for $l \in S_t$ do
            $\widetilde{f}_t^{(l)} \leftarrow \frac{\widehat{f}_t^{(l)}}{|\widehat{f}_t^{(l)}|}(1+\alpha)^{\lceil \log_{(1+\alpha)} |\widehat{f}_t^{(l)}| \rceil}$
        end for
        $u_t \leftarrow$ PrivateMedian$(\widetilde{f}_t^{(l)})$
    end for
end procedure
```

Accuracy Guarantee In the following sections, we argue that $\mathcal{B}$ maintains an accurate approximation of $\|Gh\|_2^2$ against an adaptive adversary. At first, we prove that the transcript $\mathcal{T}$ between the `Adversary` and the algorithm $\mathcal{B}$ is differentially private with respect to the database $\mathcal{R}$, where $\mathcal{R}$ is a matrix generated by the randomness of $\mathcal{B}$. Then, we prove that for all t, the aggregated output u_t is indeed an $(\alpha + \gamma + \alpha\gamma)$-approximation of $\|Gh\|_2^2$ with probability $1 - \delta$ by Chernoff bound (Lemma A.1).

Privacy Guarantee Let $r^1, \dots, r^L \in \{0, 1\}^*$ denote the random strings used by copies of the oblivious algorithm $\mathcal{A}$ as[7] $\mathcal{A}^1, \dots, \mathcal{A}^L$ during the T updates. We further denote $\mathcal{R} = \{r^1, \dots, r^L\}$, and we view every r^l as a row of the database $\mathcal{R}$. In the following paragraphs, we will show that the transcript between the `Adversary` and the above algorithm $\mathcal{B}$ is differentially private with respect to $\mathcal{R}$.

To proceed, for each step t, fixing the random strings $\mathcal{R}$, we define $u_t(\mathcal{R})$[8] as the output of algorithm $\mathcal{B}$, and $\mathcal{T}_t(\mathcal{R}) = ((G_t, h_t), u_t(\mathcal{R}))$ as the transcript between the `Adversary` and algorithm $\mathcal{B}$ at time step t. Furthermore, we denote $\mathcal{T}(\mathcal{R}) = \{x_0, \mathcal{T}_1(\mathcal{R}), \dots, \mathcal{T}_T(\mathcal{R})\}$ as the transcript. We view $\mathcal{T}_t$ and $\mathcal{T}$ as algorithms that return the transcripts given a database $\mathcal{R}$. In this light, we prove in Section C.3.2 that the transcript $\mathcal{T}$ is differentially private with respect to $\mathcal{R}$. □

Runtime Analysis Here, we present the calculation of the total runtime of our algorithm.

Lemma C.20 (Runtime)**.** *The total runtime of $\mathcal{B}$ is at most*

$$\widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{prep}} + T^{3/2}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{update}} + T\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{query}}),$$

where $\widetilde{O}$ hides the poly log *factor of T.*

Proof. We can calculate the update time in the following:

- Preprocess L copies of $\mathcal{A}$: $L \cdot \mathcal{T}_{\mathrm{prep}}$.

[7] In our application, the random string is used to generate the random sketching matrices.

[8] $\widehat{f}_t(\mathcal{R})$ is still a random variable due to private median step.

- Handle T updates: $LT \cdot \mathcal{T}_{\mathrm{update}}$.
- For each step t,
  - Query q many copies of $\mathcal{A}$ cost: $q \cdot \mathcal{T}_{\mathrm{query}}$
  - By binary search, rounding every output $\widehat{f}_t$ to the nearest power of $(1+\alpha)$ takes

    $$O(q \cdot \log \frac{\log U}{\alpha})$$

    time.
  - By Lemma C.17, computing PRIVATEMEDIAN with privacy guarantee $\varepsilon_{\mathsf{pm}}$ takes

    $$\widetilde{O}(q \cdot \mathrm{poly}\log(\frac{\log U}{\alpha\delta_0}))$$

    time.

Therefore, we conclude that the total update time of $\mathcal{B}$ is at most

$$\widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{prep}} + T^{3/2}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{update}} + T\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{query}}).$$

We can upper bound the t_{total} as follows:

$$\begin{aligned} t_{\mathrm{total}} &= L \cdot \mathcal{T}_{\mathrm{prep}} + LT \cdot \mathcal{T}_{\mathrm{update}} + T(q \cdot \mathcal{T}_{\mathrm{query}} + t_{\mathsf{pm}} + O(q \log \frac{\log U}{\alpha})) \\ &= O(\mathcal{T}_{\mathrm{prep}} \cdot \sqrt{T}\log(\frac{T\log U}{\alpha\delta}) \cdot \sqrt{\log\frac{T}{\delta}}) + O(T \cdot \mathcal{T}_{\mathrm{update}} \cdot \sqrt{T}\log(\frac{T\log U}{\alpha\delta}) \cdot \sqrt{\log\frac{T}{\delta}}) \\ &+ O(T\log(\frac{T\log U}{\alpha\delta})\mathcal{T}_{\mathrm{query}} + T \cdot \frac{1}{\varepsilon_{\mathsf{pm}}}\log^3(|X_{\mathsf{pm}}|/\beta) \cdot \mathrm{poly}\log|X_{\mathsf{pm}}|) \\ &= \widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{prep}} + T^{3/2}\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{update}} + T\log(\frac{\log U}{\alpha\delta})\mathcal{T}_{\mathrm{query}}) \end{aligned}$$

where the first step follows from plugging in the running time of query $\mathcal{T}_{\mathrm{prep}}$, update $\mathcal{T}_{\mathrm{update}}$ and private median t_{pm}, the second step follows from the choice of q from Eq.(1), and the last step follows from hiding the log factors into $\widetilde{O}(\cdot)$. □

C.3.2. PRIVACY GUARANTEE

We start by presenting the privacy guarantee for the transcript $\mathcal{T}_t$.

Lemma C.21. *For every time step t, $\mathcal{T}_t$ is $(\frac{6q}{L} \cdot \varepsilon_{\mathsf{pm}}, 0)$-DP with respect to $\mathcal{R}$.*

Proof. For a given step t, the only way that the transcript $\mathcal{T}_t(\mathcal{R}) = (G_t, h_t, u_t(\mathcal{R}))$ could leak information about $\mathcal{R}$ is by revealing the output $u_t(\mathcal{R})$. In this light, we analyze the differential privacy guarantee of the algorithm $\mathcal{B}$ by analyzing the privacy of the output $u_t(\mathcal{R})$.

If we run PRIVATEMEDIAN with $\varepsilon = \varepsilon_{\mathsf{pm}}$ and $\beta = \delta_0$ on *all* copies of $\mathcal{A}$, then from Theorem C.17, we get that the output $u_t(\mathcal{R})$ would be $(\varepsilon_{\mathsf{pm}}, 0)$-DP.

Instead, we run PRIVATEMEDIAN with $\varepsilon = \varepsilon_{\mathsf{pm}}$ and $\beta = \delta_0$ on the subsampled q copies of $\mathcal{A}$ as in Line 19 of the Algorithm 5. Then, from amplification theorem (Theorem C.15), by this subsampling, we can boost the privacy guarantee by $\frac{6q}{L}$, and hence $u_t(\mathcal{R})$ is $(\frac{6q}{L} \cdot \varepsilon_{\mathsf{pm}}, 0)$-DP with respect to $\mathcal{R}$.

□

Next, we present the privacy guarantee for the composition of the transcripts as $\mathcal{T}$.

Corollary C.22. *$\mathcal{T}$ is $(\frac{1}{200}, \frac{\delta_0}{400})$-DP with respect to R.*

Proof. Since our initialization of sketching matrices does not depend on the transcript $\mathcal{T}$, x_0 does not affect the privacy guarantee here. By Lemma C.21, each $\mathcal{T}_t$ is $\frac{3q}{2L}$-DP with respect to $\mathcal{R}$.

Moreover, we can view $\mathcal{T}$ as a T-fold adaptive composition as follows:

$$\mathcal{T}_T \circ \mathcal{T}_{T-1} \circ \cdots \circ \mathcal{T}_2 \circ \mathcal{T}_1.$$

In this light, we apply the advanced composition theorem (Theorem C.14) with $\varepsilon_1 = \frac{3q}{2L},\ \delta_2 = \delta_0/400, \delta_1 = 0, k = T$, we have that $\mathcal{T}$ is $(\varepsilon_1, \delta_1 k + \delta_2)$-DP, where:

$$\begin{aligned}\varepsilon_1 &= \sqrt{2k\ln(1/\delta_2)} \cdot \varepsilon_{\mathsf{pm}} + 2k\varepsilon_{\mathsf{pm}}^2 \\ &= \sqrt{2T\ln(400/\delta_0)} \cdot \varepsilon_{\mathsf{pm}} + 2T\frac{9q^2}{4L^2}\varepsilon_{\mathsf{pm}}^2 \\ &\leq \frac{1}{400} + \frac{1}{400} = \frac{1}{200}\end{aligned}$$

for $L = 600 \cdot q\sqrt{4T\ln(400/\delta_0)}$. □

Next, we prove that algorithm $\mathcal{B}$ has accuracy guarantee against an adaptive adversary. Let $x_{[t]} = (x_0, x_1, \ldots, x_t)$ denote the input sequence up to time t, where $x_t = (G_t, h_t)$. Let $\mathcal{A}(r, x_{[t]})$ denote the output of the algorithm $\mathcal{A}$ on input sequence $x_{[t]}$, given the random string r. Then, let $\mathbf{1}[x_{[t]}, r]$ denote the indicator whether $\mathcal{A}(r, x_{[t]})$ is an $(\gamma+\alpha+\alpha\gamma)$-approximation of $\|G_t h_t\|_2^2$, i.e:

$$\mathbf{1}[x_{[t]}, r] = \mathbf{1}\{\text{the event } \mathsf{E}_{t,r} \text{ holds}\}$$

where event $\mathsf{E}_{t,r}$ is defined as

$$\|G_t h_t\|_2^2 - (\alpha+\gamma+\alpha\gamma)\|G_t\|_F^2\|h_t\|_2^2 \leq f(G_t, h_t) \leq \|G_t h_t\|_2^2 + (\alpha+\gamma+\alpha\gamma)\|G_t\|_F^2\|h_t\|_2^2$$

Now, we show that most instances of the copies of oblivious algorithm $\mathcal{A}$ maintains the $(\gamma+\alpha+\alpha\gamma)$-approximation of $\|G_t h_t\|_2^2$.

C.3.3. Accuracy Guarantee

In this section, we present the accuracy guarantee of our algorithm $\mathcal{B}$.

Lemma C.23 (Accuracy of Algorithm $\mathcal{A}$)**.** *With probability 9/10, the output $\widetilde{u}_t$ of the algorithm $\mathcal{A}$ for every time step t is an $(\alpha+\gamma+\gamma\alpha)$-approximation of $\|G_t h_t\|_2^2$, i.e., $\mathbb{E}[\mathbf{1}[x_{[t]}, r]] = 9/10$.*

Proof. We know that the oblivious algorithm $\mathcal{A}$ will output an γ-norm approximation of Gh with probability $9/10$ as $\widehat{f}$. For these $\widehat{f}$, we proved that by rounding them up to the nearest power of $(1+\alpha)$, the resulting $\widetilde{u}$ remains to be an $(\gamma+\alpha+\alpha\gamma)$-approximation of $\|Gh\|_2^2$. Hence, with probability $9/10$, the following two inequalities hold true simultaneously:

$$\|G_t h_t\|_2^2 - \gamma\|G_t\|_F^2\|h_t\|_2^2 \leq \widehat{f}_t \leq \widetilde{u}_t$$

where this step directly follows from the approximation guarantee of oblivious algorithm $\mathcal{A}$.

$$\begin{aligned}\widetilde{u}_t &\leq (1+\alpha)\widehat{f}_t \\ &\leq (1+\alpha)(\|G_t h_t\|_2^2 + \gamma\|G_t\|_F^2\|h_t\|_2^2) \\ &\leq \|G_t h_t\|_2^2 + \alpha\|G_t h_t\|_2^2 + (1+\alpha)\gamma\|G_t\|_F^2\|h_t\|_2^2 \\ &\leq \|G_t h_t\|_2^2 + (\alpha+\gamma+\alpha\gamma)\|G_t\|_F^2\|h_t\|_2^2\end{aligned}$$

where the first step follows from plugging in the approximation guarantee of oblivious algorithm $\mathcal{A}$ and our proposed rounding-up procedure, the second step follows from equation expansion, and the last step follows from applying Cauchy-Schwarz inequality that $\|G_t h_t\|_2 \leq \|G_t\|_F\|h_t\|_2$. □

Since every copies of $\mathcal{A}$ will output an $\widetilde{u}$ satisfies the above approximation result with probability $9/10$, we have that $\mathbb{E}[\mathbf{1}[x_{[t]}, r]] = 9/10$.

Then, we present the accuracy guarantee of all L copies of $\mathcal{A}$ at every time step t.

Lemma C.24 (Accuracy of *all* copies of $\mathcal{A}$)**.** *For every time step t, $\sum_{l=1}^{L} \mathbf{1}[x_{[t]}, r^{(l)}] \geq \frac{4}{5} L$ with probability at least $1 - \delta_0$.*

Proof. We view each r as an i.i.d draw from a distribution $\mathcal{D}$, and we present the generalization guarantee on the database $\mathcal{R}$. From Lemma C.23, we know that $\mathbb{E}[\mathbf{1}[x_{[t]}, r]] = 9/10$. By Corollary C.22, algorithm $\mathcal{B}$ is $(\varepsilon_3, \delta_3)$-DP with $\varepsilon_3 = \frac{1}{200}$ and $\delta_3 = \frac{\delta_0}{400}$. Moreover, we can check that $L = \widetilde{O}(\sqrt{T} \log(\frac{\log U}{\alpha \delta_0})) \geq \varepsilon_3^{-2} \log(2\varepsilon_3/\delta_3)$

Then, by applying generalization theorem (Theorem C.16) with $t = L$, we have that:

$$\Pr_{\mathcal{R} \sim \mathcal{D}^L, \mathbf{1}[x_{[t]}, \cdot] \leftarrow \mathcal{B}(\mathcal{R})} \left[\frac{1}{|\mathcal{R}|} \sum_{l=1}^{L} \mathbf{1}[x_{[t]}, r^{(l)}] - \mathop{\mathbb{E}}_{r \sim \mathcal{D}}[\mathbf{1}[x_{[t]}, r]] \geq 10 \cdot \varepsilon_3 \right] \leq \delta_3/\varepsilon_3$$

and

$$\Pr_{\mathcal{R} \sim \mathcal{D}^L, \mathbf{1}[x_{[t]}, \cdot] \leftarrow \mathcal{T}(\mathcal{R})} \left[\frac{1}{L} \sum_{l=1}^{L} \mathbf{1}[x_{[t]}, r^{(l)}] - \mathop{\mathbb{E}}_{r \sim \mathcal{D}}[\mathbf{1}[x_{[t]}, r]] \geq \frac{1}{20} \right] \leq \frac{\delta_0}{2}$$

where the second step follows from plugging in the value of parameters ε_3 and δ_3. Then, it immediately follows that with probability at least $1 - \delta_0/2$,

$$\frac{1}{L} \sum_{l=1}^{L} \mathbf{1}[x_{[t]}, r^{(l)}] \geq 9/10 - 1/20 = 0.85.$$

Thus, we complete the proof. □

Then, we prove that after the aggregation, PRIVATEMEDIAN outputs an $(\alpha + \gamma + \alpha\gamma)$-approximation of $\|G_t h_t\|_2^2$ as u_t with probability at least $1 - \delta_0/2$. Moreover, this statement holds true for all t simultaneously with probability $1 - \delta$.

Corollary C.25 (Accuracy of the final output)**.** *With probability $1 - \delta$, the following guarantee holds for all $t \in [T]$ simultaneously:*

$$\|G_t h_t\|_2^2 - (\alpha + \gamma + \alpha\gamma)\|G_t\|_F^2 \|h_t\|_2^2 \leq u_t \leq \|G_t h_t\|_2^2 + (\alpha + \gamma + \alpha\gamma)\|G_t\|_F^2 \|h_t\|_2^2$$

Proof. Consider a fixed step t, we know that $\mathcal{B}$ independently samples q indices as set S_t and queries those copies of $\mathcal{A}$ with those indices. For ease of notation, we let $\mathbf{1}[l] = \mathbf{1}[x_{[t]}, r^{(l)}]$ denote the indicator that whether $\mathcal{A}^{(l)}$ is accurate at time t.

Since $\mathcal{A}^{(l)}$ are $i.i.d$ sampled, the $\mathbf{1}[l]$ is also i.i.d distributed. Then, from Lemma C.24, we know that with probability $1 - \delta_0$, $\mathbb{E}[\sum_{l = \in [L]} \mathbf{1}[l] \geq 0.85L]$, which implies with probability $1 - \delta_0$, $\mathbb{E}[\mathbf{1}[l] \geq 0.85]$.

Then, for $l \in S_t$, with probability $1 - \delta_0$, $\mathbb{E}[\sum_{l \in S_t} \mathbf{1}[l] \geq 0.85q]$. Moreover, from Lemma C.17, we know that with probability $1 - \delta_0$, there are 49% fraction of outputs that are at least u_t as well as bigger than u_t in set S_t.

Then, by Hoeffding's bound (Lemma A.2), we have the following:

$$\Pr\Big[|\sum_{l \in S_t} \mathbf{1}[l] - \mathbb{E}[\sum_{l=1}^{L} \mathbf{1}[l]| \geq 0.05q \Big] \leq 2\exp(-\frac{1}{400} q)$$

Therefore, with probability $1 - 2\beta$, u_t is an $(\alpha + \gamma + \alpha\gamma)$-approximation of $\|G_t h_t\|_2$. Hence, with probability at most $\exp(-\Theta(q)) \leq \delta_0$, PRIVATEMEDIAN$_{\varepsilon_{\mathrm{pm}}, \delta_0}$ returns u_t without an $(\alpha + \gamma + \alpha\gamma)$- approximation guarantee.

Furthermore, by union bound, we have that with probability $1 - 2T\delta_0 = 1 - \delta$, for all t, u_t is an $(\alpha + \gamma + \alpha\gamma)$-approximation of $\|G_t h_t\|_2^2$. □

D. Robust Set Query Data Structure

This section is organized as follows: We present the definition of set query problem in Section D.1. We present our main results and the algorithm on the set query problem in Section D.2. We present the privacy guarantee for the transcript between the adversary and the algorithm of the t-th round in Section D.3, and for those of all rounds in Section D.4. We present the accuracy guarantee for the output of *each* copy of oblivious algorithm $\mathcal{A}$ in Section D.5, and for *all* copies in Section D.6. We present the accuracy guarantee of those outputs that aggregated by private median in Section D.7.

D.1. Definition

At first, we present the definition of the set query problem and the associate ε-approximation guarantee.

Definition D.1 (Set Query)**.** Let $G \in \mathbb{R}^{n \times n}$ and $h \in \mathbb{R}^n$. Given a set $Q \subseteq [n]$ and $|Q| = k$, the goal is to estimate the coordinates of Gh in set Q. Given a precision parameter ε, for each $j \in Q$, we want to design a function f that is an ε-approximation of $(g_j^\top h)^2$, i.e.,

$$(g_j^\top h)^2 - \varepsilon \|g_j\|_2^2 \|h\|_2^2 \leq f(G, h)_j \leq (g_j^\top h)^2 + \varepsilon \|g_j\|_2^2 \|h\|_2^2$$

where g_j denotes the j-th row of G.

In the remainder of this section, we denote k as the number of elements defined in the set query problem, and we denote q as the number of copies of algorithm $\mathcal{A}$ that we use.

D.2. Main Results

The goal of this section is to prove the following norm estimation guarantee (Theorem D.2) that, when given an approximation algorithm against an *oblivious* adversary for the set query problem, we can adapt it to an approximation algorithm against an *adaptive* adversary for the same problem with slightly worse approximation guarantee.

Theorem D.2 (Reduction to Adaptive Adversary: Set query Estimation. Formal version of Theorem 6.2)**.** *We define $\mathcal{U} := [-U, -\frac{1}{U}] \cup \{0\} \cup [\frac{1}{U}, U]$ for $U > 1$. Given two parameters $\delta, \alpha > 0$. We define function f to be a function that maps elements from domain $G \times H$ to an element in $\mathcal{U}^d$.*

Suppose there is a dynamic algorithm $\mathcal{A}$ against an oblivious adversary that, given an initial data point $x_0 \in X$ and T updates, the following conditions are holding:

- *The preprocessing time is $\mathcal{T}_{\text{prep}}$.*
- *The update time per round is $\mathcal{T}_{\text{update}}$.*
- *The query time is $\mathcal{T}_{\text{query}}$ and given a set $Q_t \subset [n]$ with cardinality k, with probability $\geq 9/10$, the algorithm outputs $f(G_t, h_t)_j$ where $j \in Q_t$, and each $f(G_t, h_t)_j$ satisfies the following guarantee:*

$$(g_j^\top h)^2 - \gamma \|g_j\|_2^2 \|h_t\|_2^2 \leq f(G_t, h_t)_j \leq (g_j^\top h)^2 + \gamma \|g_j\|_2^2 \|h_t\|_2^2$$

where g_j denotes the j-th row of matrix G_t.

Then, there exists a dynamic algorithm $\mathcal{B}$ against an adaptive adversary, with probability at least $1 - \delta$, obtains an $(\alpha + \gamma + \alpha\gamma)$-approximation of $(g_j^\top h)^2$ for every $j \in Q$, the following conditions are holding:

- *The preprocessing time is $\widetilde{O}(\sqrt{T} \log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\text{prep}})$.*
- *The update time per round is $\widetilde{O}(\sqrt{T} \log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\text{update}})$.*
- *The per round query time is $\widetilde{O}(\log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\text{query}} + k^{3/2} \mathrm{poly}\log(\frac{\log U}{\alpha\delta}))$ and, with probability $1 - \delta$, for every $j \in Q$, the answer $(u_t)_j$ is an $(\alpha + \gamma + \alpha\gamma)$-approximation of $(g_j^\top h)^2$ for every t, i.e.*

$$(g_j^\top h_t)^2 - (\gamma + \alpha + \gamma\alpha) \|g_j\|_2^2 \|h_t\|_2^2 \leq (u_t)_j \leq (g_j^\top h_t)^2 + (\gamma + \alpha + \gamma\alpha) \|g_j\|_2^2 \|h_t\|_2^2.$$

Algorithm 6 Our Norm Estimation Algorithm for Set Query Problem

```
procedure ReductionAlgorithm($T, U, \alpha, \delta$)                          ▷ Theorem 6.2
    $L \leftarrow \widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta}))$          ▷ The total number of copies
    $q \leftarrow \widetilde{O}(\log(\frac{\log U}{\alpha\delta}))$       ▷ The number of copies being used in each iteration
    for $l \in [L]$ do
        Initialize $\mathcal{A}^{(l)}$ with the initial data point $x_0$
    end for
    for $t = 1 \to T$ do
        We receive $a$ query set $Q_t \subset [n]$
        for $l = 1 \to L$ do
            $\mathcal{A}^{(l)}$.Update($G_t, h_t$).
        end for
        We independently uniformly sample $q$ indices and denote this index set as $S_t \subset [L]$.
        for $l \in S_t$ do
            $\widehat{f}_t^{(l)} \leftarrow \mathcal{A}^{(l)}$.Query()
        end for
        for $l \in S_t$ do
            for $j \in Q_t$ do
                $(\widetilde{f}_t^{(l)})_j \leftarrow \frac{(\widehat{f}_t^{(l)})_j}{|(\widehat{f}_t^{(l)})_j|}(1+\alpha)^{\lceil \log_{(1+\alpha)} |(\widehat{f}_t^{(l)})_j| \rceil}$
            end for
        end for
        for $j \in Q_t$ do
            $(u_t)_j \leftarrow$ PrivateMedian($\{(\widetilde{f}_t^{(l)})_j\}_{l \in S_t}$).
        end for
    end for
end procedure
```

At first, we provide the algorithm $\mathcal{B}$ for set query problem as follows:

Proof. **Algorithm** $\mathcal{B}$. We first describe the algorithm $\mathcal{B}$.

- Let $L = \widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta}))$. We initialize L copies of $\mathcal{A}$. We call them $\mathcal{A}^{(1)}, \cdots, \mathcal{A}^{(L)}$. Suppose the initial data point is x_0.
- For time step $t = 1, \ldots, T$:
    - We update each copy of $\mathcal{A}$ by (G_t, h_t).
    - We independently uniformly sample $q = \widetilde{O}(\log(\frac{\log U}{\alpha\delta}))$ indices as set S_t.
    - For $l \in S_t$, we query $\mathcal{A}^{(l)}$ and let $\widehat{f}_t^{(l)}$ denote its output for current update. For these nonzero outputs, we round every entry $j \in Q_t$ of them to the nearest power of $(1+\alpha)$, and denote it by $\widetilde{f}_t^{(l)}$, i.e., for every $l \in S_t$:

    $$(\widetilde{f}_t^{(l)})_j = \frac{(\widehat{f}_t^{(l)})_j}{|(\widehat{f}_t^{(l)})_j|}(1+\alpha)^{\lceil \log_{(1+\alpha)}(|(\widehat{f}_t^{(l)})_j|) \rceil}$$

    - Finally, for every $j \in Q_t$, we aggregate the rounded output by PrivateMedian (Lemma C.17) with input $\{(\widetilde{f}_t^{(l)})_j\}_{l \in S_t}$, and then output the differentially private norm estimate $(u_t)_j$.

where $\widetilde{O}$ hides the $\mathrm{poly}\log T$ factor.

Parameters Here, we choose the parameters of the algorithm as follows:

$$\varepsilon_{\mathsf{pm}} = \frac{1}{4\sqrt{k\log(400T/\beta)}}, \quad \beta = \delta/(4T), \quad L = \widetilde{O}(\sqrt{T}\log(\frac{\log U}{\alpha\delta})), \quad q = \widetilde{O}(\log(\frac{\log U}{\alpha\delta})). \tag{2}$$

Update time

- Preprocess L copies of $\mathcal{A}$: $L \cdot \mathcal{T}_{\mathrm{prep}}$.
- Handle T updates: $LT \cdot \mathcal{T}_{\mathrm{update}}$.
- For each step $t \in [T]$,
  - Query q many copies of $\mathcal{A}$ cost: $q \cdot \mathcal{T}_{\mathrm{query}}$
  - By binary search, rounding up every output $\widehat{f}_t$ to the nearest power of $(1+\alpha)$ takes $O(kq \cdot \log \frac{\log U}{\alpha})$ time.
  - By Lemma C.17, computing PRIVATEMEDIAN for q entries with $\varepsilon_{\mathsf{pm}} = \frac{1}{4k}$ takes $t_{\mathsf{pm}} = \widetilde{O}(kq \cdot \mathrm{poly}\log(\frac{\log U}{\alpha\beta}))$ time.

Therefore, we conclude that the total update time of algorithm $\mathcal{B}$ is at most t_{total}, and we can upper bound t_{total} as follows:

$$
\begin{aligned}
& t_{\mathrm{total}} \\
= {} & L \cdot \mathcal{T}_{\mathrm{prep}} + LT \cdot \mathcal{T}_{\mathrm{update}} + T(\mathcal{T}_{\mathrm{query}} + kt_{pm} + \widetilde{O}(k \log \frac{\log U}{\alpha})) \\
= {} & O(\mathcal{T}_{\mathrm{prep}} \cdot \sqrt{T} \log(\frac{T \log U}{\alpha\delta}) \cdot \sqrt{\log \frac{T}{\delta}}) + O(T \cdot \mathcal{T}_{\mathrm{update}} \cdot \sqrt{T} \log(\frac{T \log U}{\alpha\delta}) \cdot \sqrt{\log \frac{T}{\delta}}) \\
& + O(T \log(\frac{T \log U}{\alpha\delta}) \mathcal{T}_{\mathrm{query}} + kT \cdot \frac{1}{\varepsilon_{\mathsf{pm}}} \log^3(|X_{\mathsf{pm}}|/\beta) \cdot \mathrm{poly}\log|X_{\mathsf{pm}}|) \\
= {} & \widetilde{O}(\sqrt{T} \log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\mathrm{prep}} + T^{\frac{3}{2}} \log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\mathrm{update}} + T \log(\frac{\log U}{\alpha\delta}) \mathcal{T}_{\mathrm{query}} + k^{\frac{3}{2}} T \mathrm{poly}\log(\frac{\log U}{\alpha\delta}))
\end{aligned}
$$

where the first step follows from plugging in the running time of query $\mathcal{T}_{\mathrm{prep}}$, update $\mathcal{T}_{\mathrm{update}}$ and private median t_{pm} with privacy guarantee $\varepsilon_{\mathsf{pm}} = \frac{1}{4\sqrt{k \log(400T/\beta)}}$, the second step follows from the choice of L from Eq. (2), and the last step follows from hiding the log factors into $\widetilde{O}(\cdot)$.

Privacy Guarantee In the following sections, we argue that $\mathcal{B}$ maintains an accurate approximation of $(g_j^\top h)^2$ for $j \in Q_t$ against an adaptive adversary. At first, we prove that the transcript $\mathcal{T}$ between the `Adversary` and the algorithm $\mathcal{B}$ is differentially private with respect to the database $\mathcal{R}$, where $\mathcal{R}$ is a matrix generated by the randomness of $\mathcal{B}$. Then, we prove that for every $j \in Q_t$, the j-th coordinate of aggregated output u is indeed an $(\alpha + \gamma + \alpha\gamma)$-approximation of $\|g_j h\|_2^2$ with probability $1 - \delta$ by Chernoff–Hoeffding inequality.

Let $r^1, \dots, r^L \in \{0,1\}^*$ denote the random strings used by the oblivious algorithms[9] $\mathcal{A}^1, \dots, \mathcal{A}^L$ during the T updates. We further denote $\mathcal{R} = \{r^1, \dots, r^L\}$, and we view every r^l as a row of the database $\mathcal{R}$. In the following paragraphs, we will show that the transcript between the `Adversary` and the above algorithm $\mathcal{B}$ is differentially private with respect to $\mathcal{R}$.

To proceed, for each time step t, fixing the random strings $\mathcal{R}$, we define $u_t(\mathcal{R})$[10] as the output of algorithm $\mathcal{B}$, and $\mathcal{T}_t(\mathcal{R}) = ((G_t, h_t), u_t(\mathcal{R}))$ as the transcript between the `Adversary` and algorithm $\mathcal{B}$ at time step t. Furthermore, we denote

$$\mathcal{T}(\mathcal{R}) = \{x_0, \mathcal{T}_1(\mathcal{R}), \dots, \mathcal{T}_T(\mathcal{R})\}$$

as the transcript. We view $\mathcal{T}_t$ and $\mathcal{T}$ as algorithms that return the transcripts given a database $\mathcal{R}$. In this light, we prove in the following that they are differentially private with respect to $\mathcal{R}$. □

[9] In our application, the random string is used to generate the random sketching matrices.

[10] $u_t(\mathcal{R})$ is still a random variable due to private median step.

D.3. Privacy Guarantee for t-th Transcript

At first, we present the privacy guarantee for transcript $\mathcal{T}_t$ at time t.

Lemma D.3 (Privacy guarantee for $\mathcal{T}_t$)**.** *For every time step t, $\mathcal{T}_t$ is* $(\frac{3q}{2L}, \frac{\beta}{800T})$-DP *with respect to $\mathcal{R}$.*

Proof. For a given step t, the only way that a transcript $\mathcal{T}_t(\mathcal{R}) = (G_t, h_t, u_t(\mathcal{R}))$ could possibly leak information about database $\mathcal{R}$ is by revealing $u_t(\mathcal{R})$. By Theorem C.17, we have that $\textsc{PrivateMedian}_{\varepsilon_{\mathsf{pm}},\beta}$ gives an $(\varepsilon_{\mathsf{pm}}, 0)$-DP output $(u_t)_j$ on $j \in Q_t$. Then, from amplification theorem (Theorem C.15), the subsampling in our algorithm $\mathcal{B}$ boosts the privacy parameter by $\frac{6q}{L}$, and hence every queried coordinate of $u_t(\mathcal{R})$ is $(\frac{6q}{L} \cdot \varepsilon_{\mathsf{pm}}, 0)$-DP with respect to $\mathcal{R}$.

Now, applying advanced composition theorem (Theorem C.14) to the k coordinates of $u_t(\mathcal{R})$ with $\delta_0 = \beta/(800T)$ gives us a $(\frac{3q}{2L}, \frac{\beta}{800T})$-DP guarantee of $\mathcal{T}_t$, for every t. The reason is as follows:

$$\begin{aligned}
\varepsilon_0 &= \sqrt{2k \log(800T/\beta)} \cdot \varepsilon_{\mathsf{pm}} \cdot \frac{6q}{L} + 2k \cdot \varepsilon_{\mathsf{pm}}^2 \cdot (\frac{6q}{L})^2 \\
&\leq \frac{1}{2}\frac{6q}{L} + 2k \frac{36q^2}{L^2} \frac{1}{16k \log(400T/\beta)} \\
&\leq \frac{3}{4}\frac{6q}{L} \\
&= \frac{3q}{2L}
\end{aligned}$$

where the first step follows from plugging in $\varepsilon = \frac{6q}{L}\varepsilon_{\text{pm}}$ and $\delta_0 = \frac{\beta}{400T}$, the second step follows from calculation, the third step follows from $\frac{6q}{L} < 1$, and the last step follows from calculation. □

D.4. Privacy Guarantee for All Transcripts

Next, we present the privacy guarantee of the whole transcript $\mathcal{T}$.

Corollary D.4 (Privacy guarantee for $\mathcal{T}$)**.** *$\mathcal{T}$ is* $(\frac{1}{200}, \frac{\beta}{400})$-DP *with respect to $\mathcal{R}$.*

Proof. We can view $\mathcal{T}$ as an adaptive composition as:

$$\mathcal{T}_T \circ \mathcal{T}_{T-1} \circ \cdots \circ \mathcal{T}_2 \circ \mathcal{T}_1.$$

Since our initialization of sketching matrices does not depend on the transcript $\mathcal{T}$, x_0 does not affect the privacy guarantee here. By Lemma D.3, each $\mathcal{T}_t$ is $(\frac{3q}{2L}, \frac{\beta}{800T})$-DP with respect to R. Then, we apply the advanced composition theorem (Theorem C.14) with $\delta_1 = \beta/800$, we have that $\mathcal{T}$ is $(\varepsilon_1, \delta_0 T + \delta_1)$-DP, where:

$$\begin{aligned}
\varepsilon_1 =& \sqrt{2T \log(1/\delta_1)} \cdot \frac{3q}{2L} + 2T \frac{9q^2}{4L^2} \\
=& \sqrt{2T \log(800/\beta)} \cdot \frac{3}{400 \cdot 12\sqrt{4T \log(400/\beta)}} + 2T \cdot \frac{9}{4 \cdot 400^2 \cdot 36 \cdot 4T \log(400/\beta)} \\
\leq& \frac{1}{400} + \frac{1}{400} \\
=& \frac{1}{200}
\end{aligned}$$

where the first step follows from the advanced composition theorem, the second step follows from plugging in the value of δ_1 and $L = 400 \cdot 6q\sqrt{4T \ln(400/\beta)}$, the third and the final step follows from calculation. Moreover, the δ guarantee of $\mathcal{T}$ is:

$$\begin{aligned}
\delta_0 T + \delta_1 =& \frac{\beta}{800T} \cdot T + \frac{\beta}{800} \\
=& \frac{\beta}{400}
\end{aligned}$$

where the first step follows from plugging in the value of δ_0 and δ_1, and the final step follows from calculation.

Hence, our algorithm is $(\frac{1}{200}, \frac{\beta}{400})$-DP. □

D.5. Accuracy of $\mathcal{A}$ on the t-th Output

Next, we prove that algorithm $\mathcal{B}$ has accuracy guarantee against an adaptive adversary. Let $x_{[t]} = (x_0, x_1, \dots, x_t)$ denote the input sequence up to time t, where $x_t = (G_t, h_t)$. Let $\mathcal{A}(r, x_{[t]})$ and $\widetilde{u}_t$ denote the output of the algorithm $\mathcal{A}$ on input sequence $x_{[t]}$, given the random string r. Then, let $\mathbf{1}[x_{[t]}, r]$ denote the indicator whether for every $j \in Q_t$, $\mathcal{A}(r, x_{[t]})_j$ is an $(\gamma + \alpha + \alpha\gamma)$-norm approximation of $(G_t)_j^\top h_t$, i.e:

$$\mathbf{1}[x_{[t]}, r] = \mathbf{1}\{\forall j \in Q_t, \text{the event } \mathsf{E}_{j,t,r} \text{ holds}\}$$

where $\mathsf{E}_{j,t,r}$ denotes the following event

$$(g_j^\top h_t)^2 - (\alpha + \gamma + \alpha\gamma)\|g_j\|_2^2\|h_t\|_2^2 \le \mathcal{A}(r, x_{[t]})_j \le (g_j^\top h_t)^2 + (\alpha + \gamma + \alpha\gamma)\|g_j\|_2^2\|h_t\|_2^2.$$

Now, we show that most instances of the oblivious algorithm $\mathcal{A}$ are $(\gamma + \alpha + \alpha\gamma)$-approximation of $(g_j^\top h_t)^2$.

Lemma D.5 (Accuracy of $\widetilde{u}_t$)**.** *For every time step t, with probability 9/10, the output $(\widetilde{u}_t)_j$ of the algorithm $\mathcal{A}$, is an $(\alpha + \gamma + \alpha\gamma)$-approximation of $(g_j^\top h_t)^2$ simultaneously for every $j \in Q$.*

Proof. We know that the oblivious algorithm $\mathcal{A}$ will output an γ-approximation of $(g_j^\top h_t)^2$ with probability $9/10$ as $\widehat{f}_j$, for every $j \in Q$. For these $\widehat{f}_j$, we proved that by rounding them to the nearest power of $(1 + \alpha)$, they still remains to be an $(\gamma + \alpha + \alpha\gamma)$-approximation of $\|g_j h_t\|_2^2$. Hence, with probability $9/10$, the following two statements hold true simultaneously:

$$\|g_j h_t\|_2^2 - \gamma\|g_j\|_2^2\|h_t\|_2^2 \le \widehat{f}_j \le (\widetilde{u}_t)_j$$

where this step follows from our rounding up procedure.

$$\begin{aligned}(\widetilde{u}_t)_j &\le (1+\alpha) \cdot \|\widehat{f}_j\|_2^2 \le (1+\alpha)(\|g_j h_t\|_2^2 + \gamma\|g_j\|_2^2\|h_t\|_2^2) \\ &= \|g_j h_t\|_2^2 + \alpha\|g_j h_t\|_2^2 + \gamma(1+\alpha)\|g_j\|_2^2\|h_t\|_2^2 \\ &\le \|g_j h_t\|_2^2 + (\alpha + \gamma + \alpha\gamma)\|g_j\|_2^2\|h_t\|_2^2\end{aligned}$$

where the first step follows from the rounding up procedure and the guarantee of $\widehat{f}_j$, the second step follows from equation expansion, and the last step follows from Cauchy-Schwarz.

□

Since every copy of $\mathcal{A}$ will have an output that satisfies the above approximation result with probability $9/10$, we have that $\mathbb{E}[\mathbf{1}[x_{[t]}, r]] = 9/10$.

D.6. Accuracy of All Copies of $\mathcal{B}$

In this section, we give the guarantee on the accuracy of *all* copies that the algorithm $\mathcal{B}$ maintains.

Lemma D.6 (Accuracy on all L copies of Algorithm $\mathcal{A}$)**.** *For every time step t, $\sum_{l=1}^L \mathbf{1}[x_{[t]}, r^{(l)}] \ge \frac{4}{5}L$ with probability at least $1 - \beta$.*

Proof. We view each r as an i.i.d draw from a distribution $\mathcal{D}$. By generalization theorem (Theorem C.16), we have that:

$$\Pr_{R\sim\mathcal{D}^L, \mathbf{1}[x_{[t]}]\leftarrow\mathcal{T}(\mathcal{R})}\Big[\frac{1}{L}\sum_{l=1}^L \mathbf{1}[x_{[t]}, r^{(l)}] - \mathop{\mathbb{E}}_{r\sim\mathcal{D}}[\mathbf{1}[x_{[t]}, r]] \ge \frac{1}{20}\Big] \le \beta/2$$

By plugging in $\varepsilon = \frac{1}{200}$, and $\delta = \frac{\beta}{400}$, we have that with probability at least $1 - \beta/2$,

$$\frac{1}{L}\sum_{l=1}^{L} \mathbf{1}[x_{[t]}, r^{(l)}] \geq 9/10 - 1/20 = 0.85.$$

□

D.7. Accuracy Guarantee of Private Median

In this section, we prove that after the aggregation, for all $j \in Q_t$, PRIVATEMEDIAN outputs an $(\alpha+\gamma+\alpha\gamma)$-approximation of $\|g_j h\|_2^2$ with probability at least $1-\beta$ by generalization theorem (Lemma C.16). Moreover, this statement holds true for all t simultaneously with probability $1-\delta$.

Corollary D.7 (Accuracy guarantee of private median)**.** *For all step t, with probability $1-\delta$, for every $j \in Q$, the following statement holds true:*

$$(g_j^\top h_t)^2 - (1-\gamma-\alpha-\gamma\alpha)\|g_j\|_2^2\|h_t\|_2^2 \leq (u_t)_j \leq (g_j^\top h_t)^2 + (1+\gamma+\alpha+\alpha\gamma)\|g_j\|_2^2\|h_t\|_2^2$$

Proof. Consider a fixed step t, we know that $\mathcal{B}$ independently samples q indices as set S_t and queries $\mathcal{A}^{(l)}$ for $l \in S_t$. For ease of notation, we let $\mathbf{1}[l] = \mathbf{1}[x_{[t]}, r^{(l)}]$ denote the whether $\mathcal{A}^l$ is accurate at time t.

From Lemma D.6, we know that with probability $1-\delta$, $\mathbb{E}[\sum_{l\in S_t}\mathbf{1}[l] \geq 0.85q]$. Then, by Hoeffding's bound (Lemma A.2), we have the following:

$$\Pr\Big[|\sum_{l\in S_t}\mathbf{1}[l] - \mathbb{E}[\sum_{l\in S_t}\mathbf{1}[l]]| \geq 0.05q\Big] \leq 2\exp(-q/400)$$

Hence, with probability at most $\exp(-\Theta(q)) \leq \beta$, $\text{PRIVATEMEDIAN}_{\varepsilon_{\mathsf{pm}},\beta}$ returns u_t that there exist a $j \in Q_t$ that $(u_t)_j$ doesn't have an $(\alpha+\gamma+\alpha\gamma)$-approximation guarantee.

From Lemma C.17, we know that with probability $1-\beta$, there are 49% fraction of outputs of $\mathcal{A}^{(l)}$ for $l \in S_t$ whose j-th coordinate is at least $(u_t)_j$ as well as bigger than $(u_t)_j$. Therefore, with probability $1-2\beta$, $(u_t)_j$ is an $(\alpha+\gamma+\alpha\gamma)$-approximation of $((G_t)_j^\top h_t)^2$ simultaneously for all $j \in Q$.

Furthermore, by union bound, we have that with probability $1-2T\beta = 1-\delta$, for every $j \in Q$, every $(u_t)_j$ is an $(\alpha+\gamma+\alpha\gamma)$-approximation of $((G_t)_j^\top h_t)^2$. □