Decremental $(1 + \epsilon)$-Approximate Maximum Eigenvector: Dynamic Power Method

Deeksha Adil[*]
Institute for Theoretical Studies
ETH Zürich
deeksha.adil@eth-its.ethz.ch

Thatchaphol Saranurak[†]
Department of Computer Science
University of Michigan
thsa@umich.edu

September 14, 2025

Abstract

We present a dynamic algorithm for maintaining $(1 + \epsilon)$-approximate maximum eigenvector and eigenvalue of a positive semi-definite matrix A undergoing *decreasing* updates, i.e., updates which may only decrease eigenvalues. Given a vector v updating $A \leftarrow A - vv^\top$, our algorithm takes $\tilde{O}(\mathrm{nnz}(v))$ amortized update time, i.e., polylogarithmic per non-zeros in the update vector.

Our technique is based on a novel analysis of the influential power method in the dynamic setting. The two previous sets of techniques have the following drawbacks (1) algebraic techniques can maintain exact solutions but their update time is at least polynomial per non-zeros, and (2) sketching techniques admit polylogarithmic update time but suffer from a crude additive approximation.

Our algorithm exploits an oblivious adversary. Interestingly, we show that any algorithm with polylogarithmic update time per non-zeros that works against an adaptive adversary and satisfies an additional natural property would imply a breakthrough for checking psd-ness of matrices in $\tilde{O}(n^2)$ time, instead of $O(n^\omega)$ time.

[*]Supported by Dr. Max Rössler, the Walter Haefner Foundation and the ETH Zürich Foundation
[†]Supported by NSF grant CCF-2238138.

Contents

1 Introduction

Computing eigenvalues and eigenvectors of a matrix is predominant in several applications, including principle component analysis, clustering in high-dimensional data, semidefinite programming, spectral graph partitioning, and algorithms such as Google's PageRank algorithm. In the era of massive and dynamic datasets, developing algorithms capable of efficiently updating spectral information with changing inputs has become indispensable.

The study of the change in the spectrum of a matrix undergoing updates spans over five decades, with Golub [Gol73] providing the first algebraic characterization of the change for positive semi-definite matrices via the *secular equation*. This result offered an explicit formula for exactly computing all new eigenvalues when a matrix undergoes a single rank-one update, assuming knowledge of the entire eigen-decomposition of the initial matrix. Subsequently, Bunch, Nielsen, and Sorensen [BNS78] showed how to additionally compute eigenvectors explicitly, and handle the case of repeated eigenvalues of the initial matrix. A decade later, Arbenz, Gander, and Golub [AGG88] extended the works of [Gol73; BNS78] to compute all new eigenvalues and eigenvectors of a positive semi-definite matrix undergoing an update of the form of a small-rank matrix, or a single batch update. Further works extend these techniques to computing singular values [Sta08], as well as computing new eigenvalues and eigenvectors when only the partial eigen-decomposition of the initial matrix is known [MSS19]. However, all these works require a full eigen-decomposition of the initial matrix and only handle a single update to the matrix.

Another independent line of work aims at finding a small and sparse matrix that has eigenvalues close to the original matrix. A recent work in this direction by Bhattacharjee et al. [Bha+23] provides a universal sparsifier $\boldsymbol{S}$ for positive semi-definite matrices $\boldsymbol{A}$, such that $\boldsymbol{S}$ has at most n/ϵ^4 non-zero entries and $\|\boldsymbol{A} - \boldsymbol{A} \circ \boldsymbol{S}\| \leq \epsilon n$. Swartworth and Woodruff [SW23] give an alternate technique by using *gaussian sketches* with $O(1/\epsilon^2)$ rows to approximate all eigenvalues of $\boldsymbol{A}$ to an additive error of $\epsilon\|\boldsymbol{A}\|_F$. The algorithms that find sparse matrices approximating the eigenvalues in these works may be extended to handle updates in the initial matrix quickly. However, the approximation of the eigenvalues achieved is quite crude, i.e., at least ϵn additive factor. This approximation is also shown to be tight for such techniques.

Now consider the simpler task of only maintaining the maximum eigenvalue and eigenvector of a matrix $\boldsymbol{A}$ as it undergoes updates. As we have seen above, known methods based on algebraic techniques require full spectral information before computing the new eigenvalues, and maintaining the entire eigen-decomposition can be slow. Sparsification-based algorithms are fast but only achieve large additive approximations of at least ϵn, which is not quite satisfactory. Works on streaming PCA, which have a similar goal of maintaining large eigenvectors focus on optimizing the space complexity instead of runtimes [AL17]. Previous works based on dynamically maintaining the matrix inverse can maintain $(1+\epsilon)$-multiplicative approximations to the maximum eigenvalues of an $n \times n$ symmetric matrix undergoing single-entry updates with an update time of $O(n^{1.447}/\text{poly}(\epsilon))$ [San04]. This was further improved to $O(n^{1.407}/\text{poly}(\epsilon))$[BNS19]. The update time is even slower for row, column, or rank-one updates. In any case, it takes polynomial time per non-zeros of the updates. This leads to the following natural question:

Is there a dynamic algorithm that can maintain a multiplicative approximation of the maximum eigenvalue of a matrix using polylogarithmic update time per non-zeros in the update?

In this paper, we study the problem of maintaining a $(1+\epsilon)$-multiplicative approximation to the maximum eigenvalue and eigenvector of a positive semi-definite matrix as it undergoes a sequence of rank-one updates that may only decrease the eigenvalues. We note that this is equivalent to finding the minimum eigenvalue and eigenvector of a positive semi-definite matrix undergoing rank-one updates that may only increase the eigenvalues. We now formally state our problem.

Problem 1.1 (Decremental Approximate Maximum Eigenvalue and Eigenvector)**.** *We are given a size parameter n, an accuracy parameter $\epsilon \in (0,1)$, a psd matrix $\boldsymbol{A}_0 \succeq 0$ of size $n \times n$, and an online sequence of vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_T$ that update $\boldsymbol{A}_t \leftarrow \boldsymbol{A}_{t-1} - \boldsymbol{v}_t \boldsymbol{v}_t^\top$ with a promise that $\boldsymbol{A}_t \succeq 0$ for all t. The goal is to maintain an ϵ-approximate eigenvalue $\lambda_t \in \mathbb{R}$ and ϵ-approximate eigenvector $\boldsymbol{w}_t \in \mathbb{R}^n$ of $\boldsymbol{A}_t$ for all time t. That is λ_t and unite vector $\boldsymbol{w}_t$ satisfying,*

$$\max_{\boldsymbol{u}:unit} \boldsymbol{u}^\top \boldsymbol{A}_t \boldsymbol{u} \geq \lambda_t \geq (1-\epsilon) \max_{\boldsymbol{u}:unit} \boldsymbol{u}^\top \boldsymbol{A}_t \boldsymbol{u}, \tag{1}$$

and

$$\boldsymbol{w}_t^\top \boldsymbol{A}_t \boldsymbol{w}_t \geq (1-\epsilon) \max_{\boldsymbol{u}:unit} \boldsymbol{u}^\top \boldsymbol{A}_t \boldsymbol{u}. \tag{2}$$

1.1 Our Results

We give an algorithm for the Decremental Approximate Maximum Eigenvalue and Eigenvector Problem that has an amortized update time of $\approx \widetilde{O}(nnz(\boldsymbol{v}_t)) \leq \widetilde{O}(n)$[1]. In other words, the total time required by our algorithm over T updates is at most $\widetilde{O}(nnz(\boldsymbol{A}_0) + \sum_{i=1}^T nnz(\boldsymbol{v}_i))$. Observe that our algorithm only requires time that is $\widetilde{O}(1)\times$ (time required to read the input) and can handle any number of decremental updates. Our algorithm works against an *oblivious* adversary, i.e., the update sequence is fixed from the beginning. This is the first algorithm that can handle a sequence of updates while providing a multiplicative approximation to the eigenvalues and eigenvectors in a total runtime of $\leq \widetilde{O}(n^2 + n \cdot T)$ and an amortized update time faster than previous algorithms by a factor of $n^{\Omega(1)}$. Formally, we prove the following:

Theorem 1.2. *There is an algorithm for Problem 1.1 under a sequence of T decreasing updates, that given parameters n, $\boldsymbol{A}_0$, and $\epsilon > 1/n$ as input, with probability at least $1 - 1/n$ works against an oblivious adversary in total time,*

$$O\left(\frac{\log^3 n \log^6 \frac{n}{\epsilon} \log \frac{\lambda_{\max}(\boldsymbol{A}_0)}{\lambda_{\max}(\boldsymbol{A}_T)}}{\epsilon^4}\left(nnz(\boldsymbol{A}_0) + \sum_{i=1}^{T} nnz(\boldsymbol{v}_i)\right)\right).$$

Our algorithm is a novel adaptation of the classical *power method* (see, e.g., [TB22]) to the dynamic setting, along with a new analysis that may be of independent interest.

Our work can also be viewed as the first step towards generalizing the dynamic algorithms for solving positive linear programs of [BKS23] to solving dynamic positive semi-definite programs. We discuss this connection in detail in Appendix A.

1.2 Towards Separation between Oblivious and Adaptive adversaries

We also explore the possibility of removing the assumption of an oblivious adversary and working against *adaptive adversaries.* Recall that update sequences are given by adaptive adversaries when the update sequence can depend on the solution of the algorithm.

[1] $\widetilde{O}$ hides polynomials in $\log n$.

We show that if there is an algorithm for Problem 1.1 with a total running time of at most $\widetilde{O}(n^2)$ such that the output $\boldsymbol{w}_t$'s satisfy an additional natural property (which is satisfied by the output of the power method, but *not* by our algorithm), then it contradicts the hardness of a well-known barrier in numerical linear algebra, therefore ruling out such algorithms.

We first state the barrier formally, and then state our result for adaptive adversaries. Recall that, given a matrix $\boldsymbol{A}$, the condition number of $\boldsymbol{A}$ is $\frac{\max_{x:\text{unit}} \|\boldsymbol{Ax}\|}{\min_{x:\text{unit}} \|\boldsymbol{Ax}\|}$.

Problem 1.3 (Checking psdness with certificate)**.** *Given $\delta \in (0,1)$, parameter κ, and a symmetric matrix $\boldsymbol{A}$ of size $n \times n$ with condition number at most κ, either*

- *Compute a matrix $\boldsymbol{X}$ where $\|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^T\| \leq \delta \min_{\|\boldsymbol{x}\|=1} \|\boldsymbol{Ax}\|$, certifying that $\boldsymbol{A}$ is a psd matrix, or*
- *Report that $\boldsymbol{A}$ is not a psd matrix.*

Recall that A is a psd matrix iff there exists $\boldsymbol{X}$ such that $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{X}^\top$. The matrix of $\boldsymbol{X}$ is called a *vector realization* of $\boldsymbol{A}$, and note that $\boldsymbol{X}$ is not unique. The problem above asks to compute a (highly accurate) vector realization of $\boldsymbol{A}$. Both eigendecomposition and Cholesky decomposition of $\boldsymbol{A}$ give a solution for Problem 1.3.[2] Banks et al [Ban+22] showed how to compute with high accuracy an eigendecomposition of a psd matrix in $O(n^{\omega+\eta})$ time for any constant $\eta > 0$, where $\omega > 2.37$ is the matrix multiplication exponent. Hence, a vector realization of $\boldsymbol{A}$ can be found in the same time. Observe that, when $\delta < \kappa$, Problem 1.3 is *at least as hard as certifying that a given matrix $\boldsymbol{A}$ is a psd matrix.* It is a notorious open problem whether certifying the psd-ness of a matrix can be done faster than $o(n^\omega)$ even when the condition number is polynomial in n. Therefore, we view Problem 1.3 as a significant barrier and formalize it as follows.

Hypothesis 1.4 (PSDness Checking Barrier)**.** *There is a constant $\eta > 0$ such that, every randomized algorithm for solving Problem 1.3 for instances with $\frac{\kappa}{\delta} \leq \text{poly}(n)$, correctly with probability at least $2/3$ requires $n^{2+\eta}$ time.*

Our negative result states that, assuming Hypothesis 1.4, there is no algorithm for Problem 1.1 against adaptive adversaries with sub-polynomial update time that maintains $\boldsymbol{w}_t$ satisfying an additional property stated below.

Property 1.5. *For every t let $\boldsymbol{u}_i(\boldsymbol{A}_t)$ denote the eigenvectors of $\boldsymbol{A}_t$ and $\lambda_i(\boldsymbol{A}_t)$ denote the eigenvalues. For all i such that $\lambda_i(\boldsymbol{A}_t) \leq \lambda_1(\boldsymbol{A}_t)/2$,*

$$\left(\boldsymbol{w}_t^\top \boldsymbol{u}_i(\boldsymbol{A}_t)\right)^2 \leq \frac{1}{n^2} \cdot \frac{\lambda_i(\boldsymbol{A}_t)}{\lambda_1(\boldsymbol{A}_t)}.$$

Theorem 1.6. *Assuming Hypothesis 1.4, there is no algorithm for Problem 1.1 that maintains $\boldsymbol{w}_t$'s additionally satisfying Property 1.5, and given parameters n and $\epsilon = \min\left\{1 - \frac{1}{n^{o(1)}}, \frac{1-\delta}{1+\delta}\right\}$ as input, works against an adaptive adversary in time $n^{o(1)} \cdot \left(nnz(\boldsymbol{A}_0) + \sum_{t=1}^{T} nnz(\boldsymbol{v}_i)\right)$.*

[2]Given an eigendecomposition $\boldsymbol{A} = \boldsymbol{Q}\Lambda\boldsymbol{Q}^\top$ where Λ is a non-negative diagonal matrix and $\boldsymbol{Q}$ is an orthogonal matrix, we set $\boldsymbol{X} = \boldsymbol{Q}\Lambda^{1/2}$. Given a Cholesky decomposition $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^\top$ where $\boldsymbol{L}$ is a lower triangular matrix, we set $\boldsymbol{X} = \boldsymbol{L}$.

Let us motivate Property 1.5 from several aspects below. First, in the static setting, this property can be easily satisfied since the static power method strongly satisfies it (see Lemma 3.3). Second, the statement of the property itself is a natural property that we might expect from an algorithm. Consider a "bad" eigenvector $\boldsymbol{u}_i$ whose corresponding eigenvalue is very small, i.e., less than half of the maximum one. It states that the projection of the output $\boldsymbol{w}_t$ along $\boldsymbol{u}_i$ should be very small, i.e., a polynomial factor smaller than the projection along the maximum eigenvector. This is intuitively useful because we do not want the output $\boldsymbol{w}_t$ to direct to the "bad" direction. It should mainly direct along the approximately maximum eigenvector. Third, we formalize this intuition and crucially exploit Property 1.5 to prove a reduction in Theorem 1.6. Specifically, Property 1.5 allows us to decrease eigenvalues of a psd matrix while maintaining the psd-ness, which is crucial for us. See Section 4 for more details. Lastly, our current algorithm from Theorem 1.2, a dynamic version of the power method, actually maintains $\boldsymbol{w}_t$'s that satisfy this property for certain snapshots, but not at every step. Unfortunately, we do not see how to strengthen our algorithm to satisfy Property 1.5 nor how to remove it from the requirement of Theorem 1.6. We leave both possibilities as open problems.

Understanding the power and limitations of dynamic algorithms against an oblivious adversary vs. an adaptive adversary has become one of the main research programs in the area of dynamic algorithms. However, finding a natural dynamic problem that separates oblivious and adaptive adversaries is still a wide-open problem.[3] In this paper, we suggest the problem of maintaining approximate maximum eigenvectors as a natural candidate for the separation and give some preliminary evidence for this.

Organization. In the following sections, we begin with some preliminaries required to prove our results in Section 2, followed by our algorithm and its analysis in Section 3, and our conditional lower bound in Section 4. In Section A we show connections with positive semi-definite programs, and finally, in Section 5, we present some open problems.

2 Preliminaries

Let $\boldsymbol{A}_0$ denote the initial matrix. Let $\lambda_0 = \lambda_{\max}(\boldsymbol{A}_0) = \|\boldsymbol{A}_0\|$ denote the maximum eigenvalue of the initial matrix $\boldsymbol{A}_0$. The following are the key definitions we use in our analysis.

Definition 2.1 (ϵ-max span and dimension)**.** *We define* $\mathrm{span}(\epsilon, \boldsymbol{A})$ *to denote the space spanned by all eigenvectors of* $\boldsymbol{A}$ *corresponding to eigenvalues* λ *satisfying* $\lambda \geq (1-\epsilon)\lambda_0$*. Let* $\dim(\epsilon, \boldsymbol{A})$ *to denote the dimension of the space* $\mathrm{span}(\epsilon, \boldsymbol{A})$.

We emphasize that λ_0 depends only on $\boldsymbol{A}_0$. So, it is a static value that does not change through time. We will use the following linear algebraic notations.

Definition 2.2. *Let* S_1 *and* S_2 *be two subspaces of a vector space* S*. The sum* $S_1 + S_2$ *is the space,*

$$S_1 + S_2 = \{s = s_1 + s_2 : s_1 \in S_1, s_2 \in S_2\}.$$

[3]Beimel et al. [Bei+22] gave the first separations for artificial problems assuming a strong cryptographic assumption. Bateni et al. [Bat+23] shows a separation between the adversaries for the *k-center clustering problem*, however, their results are based on the existence of a black box which the adaptive adversary can control.

The complement $\overline{S}$ of S is the vector space such that $S + \overline{S} = \mathbb{R}^n$, and $S \cap \overline{S} = \{0\}$. The difference, $S_1 - S_2$ is defined as,

$$S_1 - S_2 = S_1 \cap \overline{S_2}.$$

Next, we list standard facts about high-dimensional probability needed in our analysis.

Lemma 2.3 (Chernoff Bound)**.** *Let $x_1, \cdots x_m$ be independent random variables such that $a \leq x_i \leq b$ for all i. Let $x = \sum_i x_i$ and let $\mu = \mathbb{E}[x]$. Then for all $\delta > 0$,*

$$\Pr[x \geq (1+\delta)\mu] \leq \exp\left(-\frac{2\delta^2\mu^2}{m(b-a)^2}\right)$$

$$\Pr[x \leq (1-\delta)\mu] \leq \exp\left(-\frac{\delta^2\mu^2}{m(b-a)^2}\right)$$

Lemma 2.4 (Norm of Gaussian Vector)**.** *A random vector $\boldsymbol{v} \in \mathbb{R}^n$ with every coordinate chosen from a normal distribution, $N(0,1)$ satisfies,*

$$\Pr[|\|\boldsymbol{v}\|^2 - n| \leq 2(1+\delta)\delta \cdot n] \geq 1 - e^{-\delta^2 n}.$$

Proof. TOPROVE 0 □

Lemma 2.5 (Distribution of χ^2 Variable)**.** *Let $x \sim N(0,1)$ be a gaussian random variable. Then,*

$$\Pr\left[x^2 \geq \frac{1}{n^4}\right] \geq 1 - \frac{1}{n^2}.$$

Proof. TOPROVE 1 □

3 Algorithms against an Oblivious Adversary

To prove Theorem 1.2, we first reduce Problem 1.1 to solving a normalized threshold version of the problem where we assume that initially, the maximum eigenvalue is not much bigger than one. Then we want to maintain a certificate that the maximum eigenvalue is not much less than one until no such certificate exists. This is formalized below.

Problem 3.1 (DecMaxEV$(\epsilon, \boldsymbol{A}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_T)$)**.** *Let $\boldsymbol{A}_0$ be an $n \times n$ symmetric PSD matrix such that $\lambda_{\max}(\boldsymbol{A}_0) \leq 1 + \frac{\epsilon}{\log n}$. The* DecMaxEV*$(\epsilon, \boldsymbol{A}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_T)$ problem asks to find for every t, a vector $\boldsymbol{w}_t$ such that*

$$\|\boldsymbol{w}_t\| = 1 \quad \text{and} \quad \boldsymbol{w}_t^\top \boldsymbol{A}_t \boldsymbol{w}_t \geq 1 - 40\epsilon,$$

or return False *indicating that $\lambda_{\max}(\boldsymbol{A}_t) \leq 1 - \frac{\epsilon}{\log n}$.*

We defer the proof of the standard reduction stated below to the appendix.

Lemma 3.2. *Given an algorithm $\mathcal{A}$ that solves the decision problem* DecMaxEV*$(\epsilon, \boldsymbol{A}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_T)$ (Definition 3.1) for any $\epsilon > 0$, $\boldsymbol{A}_0 \succeq 0$ and vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_T$ in time $\mathcal{T}$, we can solve Problem 1.1 in total time $O\left(\frac{\log^2 n \log \frac{n}{\epsilon}}{\epsilon} \cdot nnz(\boldsymbol{A}_0) + \frac{\log n}{\epsilon} \log \frac{\lambda_{\max}(\boldsymbol{A}_0)}{\lambda_{\max}(\boldsymbol{A}_T)} \mathcal{T}\right)$.*

Algorithm 1 DecMaxEV with no update

```
1:  procedure PowerMethod($\epsilon$, $\boldsymbol{A}$)
2:      $R \leftarrow 10 \log n, r_0 \leftarrow 1$
3:      $K \leftarrow \frac{4 \log \frac{n}{\epsilon}}{\epsilon}$
4:      for $r = 1, \cdots, R$ do
5:          $\boldsymbol{v}^{(0,r)} \leftarrow$ random vector with coordinates chosen from $N(0,1)$
6:          for $k = 1 : K$ do
7:              $\boldsymbol{v}^{(k,r)} \leftarrow \boldsymbol{A}\boldsymbol{v}^{(k-1,r)}$
8:          $\boldsymbol{w}^{(r)} \leftarrow \frac{\boldsymbol{v}^{(K,r)}}{\|\boldsymbol{v}^{(K,r)}\|}$
9:      $\boldsymbol{W} = [\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(R)}]$
10:     if $(\boldsymbol{w}^{(r)})^\top \boldsymbol{A}\boldsymbol{w}^{(r)} < 1 - \epsilon$ for all $r \leq R$ then
11:         return False
12:     else
13:         $r_0 \leftarrow$ smallest $r$ such that $(\boldsymbol{w}^{(r)})^\top \boldsymbol{A}\boldsymbol{w}^{(r)} \geq 1 - 5\epsilon$
14:         return $[r_0, \boldsymbol{W}]$
```

Next, we describe Algorithm 1 which can be viewed as an algorithm for Problem 3.1 when there are no updates. Our algorithm essentially applies the *power iteration*, which is a standard algorithm used to find an approximate maximum eigenvalue and eigenvector of a matrix. In the algorithm, we make $R = O(\log n)$ copies to boost the probability. Below, we state the guarantees of the power method.

Lemma 3.3. *Let $\epsilon > 0$ and $\boldsymbol{A} \succeq 0$. Let $\boldsymbol{W}$ be as defined in Line 9 in the execution of* PowerMethod*$(\epsilon, \boldsymbol{A})$. With probability at least $1 - 1/n^{10}$, for some $\boldsymbol{w} \in \boldsymbol{W}$, it holds that $\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w} \geq (1 - \frac{\epsilon}{2})\lambda_{\max}(\boldsymbol{A})$. The total time taken by the algorithm is at most $O\left(\frac{nnz(\boldsymbol{A}) \log n \log \frac{n}{\epsilon}}{\epsilon}\right)$.*

Furthermore, let λ_i and $\boldsymbol{u}_i$ denote the eigenvalues and eigenvectors of $\boldsymbol{A}$. For all i such that $\lambda_i(\boldsymbol{A}) \leq \frac{\lambda_{\max}(\boldsymbol{A})}{2}$, with probability at least $1 - 2/n^{10}$, $\left[\boldsymbol{w}^\top \boldsymbol{u}_i\right]^2 \leq \frac{1}{n^8} \cdot \frac{\lambda_i}{\lambda_1}$.

We note that the last line of the above lemma is saying that the vectors returned by the power method satisfy Property 1.5, which we state for completeness but is not required by our algorithm. The following result is a direct consequence of Lemma 3.3.

Corollary 3.4. *Let $\epsilon > 0$, $\boldsymbol{A} \succeq 0$. Let $\boldsymbol{W}$ be as defined in Line 9 in the execution of* PowerMethod*$(\epsilon, \boldsymbol{A})$. If $\lambda_{\max}(\boldsymbol{A}) \geq 1 - \epsilon$, then with probability at least $1 - 1/n^{10}$, $\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w} \geq 1 - 5\epsilon$ for some $\boldsymbol{w} \in \boldsymbol{W}$. Furthermore, if $\lambda_{\max}(\boldsymbol{A}) \geq 1 - \epsilon/\log n$, then with probability at least $1 - 1/n^{10}$, $\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w} \geq 1 - \epsilon$ for some $\boldsymbol{w} \in \boldsymbol{W}$. The total time taken by the algorithm is at most $O\left(\frac{nnz(\boldsymbol{A}) \log n \log \frac{n}{\epsilon}}{\epsilon}\right)$.*

Observe that, if the algorithm returns $[r_0, \boldsymbol{W}]$, then $(\boldsymbol{w}^{(r)})^\top \boldsymbol{A}\boldsymbol{w}^{(r)} \geq 1 - 5\epsilon$ for $r = r_0$, and $\boldsymbol{w}^{(r_0)}$ is therefore a solution to Problem 3.1 when there is no update. The power method and its analysis are standard, and we thus defer the proof of Lemma 3.3 to the appendix.

Next, in Algorithms 2 and 3 we describe an algorithm for Problem 3.1 when we have an online sequence of updates $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_T$. The algorithm starts by initializing $R = O(\log n)$ copies of the approximate maximum eigenvectors from the power method. Given a sequence of updates, as long as one of the copies is the witness that the current matrix $\boldsymbol{A}_t$ still has a large eigenvalue, i.e., there exists r where $(\boldsymbol{w}^{(r)})_t^\top \boldsymbol{A}_t \boldsymbol{w}_t^{(r)} \geq 1 - 40\epsilon$, we can just return $\boldsymbol{w}^{(r)}$ as the solution to Problem 3.1. Otherwise, $(\boldsymbol{w}^{(r)})_t^\top \boldsymbol{A}_t \boldsymbol{w}_t^{(r)} < 1 - 40\epsilon$ for all $r \leq R$ and none of the vectors from the previous call to the power method are a witness of large eigenvalues anymore. In this case, we simply recompute these vectors by calling the power method again. If the power method returns that there is no large eigenvector, then we return FALSE from now. Otherwise, we continue in the same manner. Note that our algorithm is very simple, but as we will see, the analysis is not straightforward.

Algorithm 2 Initialization

```
1: procedure INIT($\epsilon$, $\boldsymbol{A}_0$)
2:     $\boldsymbol{W} \leftarrow$ POWERMETHOD($\epsilon$, $\boldsymbol{A}_0$)
3:     return $\boldsymbol{W}$
```

Algorithm 3 Update algorithm at time t (A_{t-1}, r_t, $\boldsymbol{W}_{t-1} = [w_{t-1}^{(r)} : r = 1, \cdots R]$, ϵ are maintained)

```
1: procedure UPDATE($\boldsymbol{v}_t$)
2:     $\boldsymbol{A}_t \leftarrow \boldsymbol{A}_{t-1} - \boldsymbol{v}_t \boldsymbol{v}_t^\top$
3:     if $(\boldsymbol{w}_{t-1}^{(r)})^\top \boldsymbol{A}_t \boldsymbol{w}_{t-1}^{(r)} < 1 - 40\epsilon$ for all $r \leq R$ then
4:         $[r_t, \boldsymbol{W}_t] \leftarrow$ POWERMETHOD($\epsilon$, $\boldsymbol{A}_t$)
5:         if POWERMETHOD($\epsilon$, $\boldsymbol{A}_t$) returns FALSE then
6:             return FALSE for all further updates
7:     else
8:         $r_t \leftarrow$ smallest $r$ such that $(\boldsymbol{w}_{t-1}^{(r)})^\top \boldsymbol{A}_t \boldsymbol{w}_{t-1}^{(r)} \geq 1 - 40\epsilon$
9:         $\boldsymbol{W}_t \leftarrow \boldsymbol{W}_{t-1}$
10:    return $[r_t, \boldsymbol{W}_t]$
```

3.1 Proof Overview

The overall proof of Theorem 1.2, including the proof of correctness and the runtime depends on the number of executions in Line 4 in Algorithm 3. If the number of executions of Line 4 is bounded by $\mathrm{poly}(\log n/\epsilon)$, then the remaining analysis is straightforward. Therefore, the majority of our analysis is dedicated to proving this key lemma, i.e., $\mathrm{poly}(\log n/\epsilon)$ bound on the number of calls to the power method:

Lemma 3.5 (Key Lemma)**.** *The number of executions of Line 4 over all updates is bounded by* $O(\log n \log^5 \frac{n}{\epsilon}/\epsilon^2)$ *with probability at least* $1 - \frac{1}{n}$.

Given the key lemma, the correctness and runtime analyses are quite straightforward and are presented in Section 3.2. We now give an overview of the proof of Lemma 3.5.

Let us consider what happens between two consecutive calls to Line 4, say at $\boldsymbol{A}$ and $\widetilde{\boldsymbol{A}} = \boldsymbol{A} - \sum_{i=1}^k \boldsymbol{v}_i \boldsymbol{v}_i^\top$. We first define the following subspaces of $\boldsymbol{A}$ and $\widetilde{\boldsymbol{A}}$. Recall Definition 2.2, which we use to define the following subspaces.

Definition 3.6 (Subspaces of $\boldsymbol{A}$ and $\widetilde{\boldsymbol{A}}$)**.** *Given $\epsilon > 0$, $\boldsymbol{A}$, and $\widetilde{\boldsymbol{A}}$ define for $\nu = 0, 1, \cdots, 15\log\frac{n}{\epsilon} - 1$:*

$$T_\nu = \operatorname{span}\left(\frac{(\nu+1)\epsilon}{5\log\frac{n}{\epsilon}}, \boldsymbol{A}\right) - \operatorname{span}\left(\frac{\nu\epsilon}{5\log\frac{n}{\epsilon}}, \boldsymbol{A}\right),$$

and,

$$\tilde{T}_\nu = \operatorname{span}\left(\frac{(\nu+1)\epsilon}{5\log\frac{n}{\epsilon}}, \widetilde{\boldsymbol{A}}\right) - \operatorname{span}\left(\frac{\nu\epsilon}{5\log\frac{n}{\epsilon}}, \widetilde{\boldsymbol{A}}\right).$$

That is, the space T_ν and $\tilde{T}_\nu$ are spanned by eigenvectors of $\boldsymbol{A}$ and $\widetilde{\boldsymbol{A}}$, respectively, corresponding to eigenvalues between $\left(1 - (\nu+1)\frac{\epsilon}{5\log\frac{n}{\epsilon}}\right)\lambda_0$ and $\left(1 - \nu\frac{\epsilon}{5\log\frac{n}{\epsilon}}\right)\lambda_0$.

Let $d_\nu = \dim(T_\nu)$ and $\tilde{d}_\nu = \dim(\tilde{T}_\nu)$. Also define,

$$\tilde{T} = \operatorname{span}(3\epsilon, \widetilde{\boldsymbol{A}}), \quad T = \operatorname{span}(3\epsilon, \boldsymbol{A}),$$

and let $d = \dim(T)$, $\tilde{d} = \dim(\tilde{T})$.

Observe that $T = \sum_{\nu=0}^{15\log\frac{n}{\epsilon}-1} T_\nu$ and similarly $\tilde{T} = \sum_{\nu=0}^{15\log\frac{n}{\epsilon}-1} \tilde{T}_\nu$. We next define some indices/levels corresponding to large subspaces, which we call "important levels".

Definition 3.7 (Important ν)**.** *We say a level ν is important if,*

$$d_\nu \geq \frac{\epsilon}{600\log^3\frac{n}{\epsilon}} \sum_{\nu' < \nu} d_{\nu'}.$$

We will use $\mathcal{I}$ to denote the set of ν that are important.

The main technical lemma that implies Lemma 3.5 is the following:

Lemma 3.8 (Measure of Progress)**.** *Let $\epsilon > 0$ and let $\boldsymbol{W} = [\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(R)}]$ be as defined in Line 9 in the execution of* PowerMethod*$(\epsilon, \boldsymbol{A})$. Let $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k$ be a sequence of updates generated by an oblivious adversary and define $\widetilde{\boldsymbol{A}} = \boldsymbol{A} - \sum_{i=1}^k \boldsymbol{v}_i\boldsymbol{v}_i^\top$.*

Suppose that $\lambda_{\max}(\boldsymbol{A}) \geq 1 - \epsilon$ and $\boldsymbol{w}^\top\widetilde{\boldsymbol{A}}\boldsymbol{w} < 1 - 40\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$. Then, with probability at least $1 - \frac{50\log\frac{n}{\epsilon}}{n^2}$, for some $\nu \in \mathcal{I}$,

- *$\dim(T_\nu - \tilde{T}) \geq \frac{\epsilon}{300\log\frac{n}{\epsilon}} d_\nu$ if $d_\nu \geq \frac{3000\log n\log\frac{n}{\epsilon}}{\epsilon}$, or*
- *$\dim(T_\nu - \tilde{T}) \geq 1$ if $d_\nu < \frac{3000\log n\log\frac{n}{\epsilon}}{\epsilon}$.*

We prove this lemma in Section 3.4. Intuitively speaking, it means that, whenever Line 4 of Algorithm 3 is executed, there is some important level ν such that an $\Omega(\epsilon/\operatorname{poly}\log(n/\epsilon))$-fraction of eigenvalues of $\boldsymbol{A}$ at level ν have decreased in value. This is the crucial place where we exploit an oblivious adversary.

Given Lemma 3.8, the remaining proof of Lemma 3.5 follows a potential function analysis which is presented in detail in Section 3.3. We consider potentials $\Phi_j = \sum_{\nu=0}^{j} d_\nu$ for $j = 0, \cdots, 15\log\frac{n}{\epsilon} - 1$. The main observation is that Φ_j is non-increasing over time for all j, and whenever there exists $\nu_0 \in \mathcal{I}$ that satisfies the condition of Lemma 3.8, Φ_{ν_0} decreases by $\dim(T_{\nu_0} - \tilde{T})$. Since $\dim(T_{\nu_0} -$

$\tilde{T}) \geq \Omega(\epsilon/\operatorname{poly}\log(n/\epsilon))d_{\nu_0}$ and $\nu_0 \in \mathcal{I}$, i.e., $\Phi_{\nu_0} = \sum_{\nu<\nu_0} d_\nu + d_{\nu_0} \leq d_{\nu_0}\left(\frac{O(\log^3 \frac{n}{\epsilon})}{\epsilon} + 1\right)$, we can prove that Φ_{ν_0} decreases by a multiplicative factor of $\Omega(1 - \epsilon^2/\operatorname{poly}\log(n/\epsilon))$. As a result, every time our algorithm executes Line 4, Φ_j decreases by a multiplicative factor for some j, and since we have at most $15 \log \frac{n}{\epsilon}$ values of j, we can only have $\operatorname{poly}(\log n/\epsilon)$ executions of Line 4.

It remains to describe how we prove Lemma 3.8 at a high level. We can write $\boldsymbol{w}^\top \widetilde{\boldsymbol{A}}\boldsymbol{w}$ for any $\boldsymbol{w} \in \boldsymbol{W}$ as

$$\boldsymbol{w}^\top \widetilde{\boldsymbol{A}}\boldsymbol{w} = \boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w} - \boldsymbol{w}^\top \boldsymbol{V}\boldsymbol{w},$$

for $\boldsymbol{V} = \sum_{i=1}^k \boldsymbol{v}_i\boldsymbol{v}_i^\top$. Our strategy is to show that:

If $\dim(T_\nu - \tilde{T})$ does not satisfies the inequalities in Lemma 3.8 for all $\nu \in \mathcal{I}$,
then $\boldsymbol{w}^\top \boldsymbol{V}\boldsymbol{w} \leq 35\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$. $(\star)$

Given $(\star)$ as formalized later in Claim 3.16, we can conclude Lemma 3.8 because, from the definition of $\boldsymbol{A}$ and $\widetilde{\boldsymbol{A}}$, we have that for some $\boldsymbol{w} \in \boldsymbol{W}$, $\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w} \geq 1-5\epsilon$ by Corollary 3.4 and $\boldsymbol{w}^\top \widetilde{\boldsymbol{A}}\boldsymbol{w} < 1-40\epsilon$. As a result for this $\boldsymbol{w}$, $\boldsymbol{w}^\top \boldsymbol{V}\boldsymbol{w} > 35\epsilon$. Now, by contra-position of $(\star)$, we have that $\dim(T_\nu - \tilde{T})$ is large for some $\nu \in \mathcal{I}$.

To prove $(\star)$, we further decompose $\boldsymbol{w}^\top \boldsymbol{V}\boldsymbol{w}$ as

$$\boldsymbol{w}^\top \boldsymbol{V}\boldsymbol{w} = \boldsymbol{w}^\top \boldsymbol{V}_{\tilde{T}}\boldsymbol{w} + \sum_{\nu=0}^{15\log\frac{n}{\epsilon}-1} \boldsymbol{w}^\top \boldsymbol{V}_{T_\nu - \tilde{T}}\boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{V}_{\overline{T}}\boldsymbol{w}.$$

In the above equation, $\boldsymbol{V}_{\tilde{T}} = \Pi_{\tilde{T}}\boldsymbol{V}\Pi_{\tilde{T}}$, $\boldsymbol{V}_{T_\nu-\tilde{T}} = \Pi_\nu \boldsymbol{V}\Pi_\nu$, and $\boldsymbol{V}_{\overline{T}} = \Pi_{\overline{T}}\boldsymbol{V}\Pi_{\overline{T}}$ where $\Pi_{\tilde{T}}, \Pi_\nu, \Pi_{\overline{T}}$ denote the projections matrices that project any vector onto the spaces $\tilde{T}$, $T_\nu - \tilde{T}$, and $\overline{T}$ respectively[4]. Refer to Section 3.4 for proof of why such a split is possible. Our proof of $(\star)$ then bounds the terms on the right-hand side. Let us consider each term separately.

1. $\boldsymbol{w}^\top \boldsymbol{V}_{\tilde{T}}\boldsymbol{w}$: We prove that this is always at most $10\epsilon(1+\epsilon)$ (Equation (**??**)). From the definition of $\boldsymbol{V}$,

$$\boldsymbol{w}^\top \boldsymbol{V}_{\tilde{T}}\boldsymbol{w} = \boldsymbol{w}^\top \Pi_{\tilde{T}}\boldsymbol{A}\Pi_{\tilde{T}}\boldsymbol{w} - \boldsymbol{w}^\top \Pi_{\tilde{T}}\widetilde{\boldsymbol{A}}\Pi_{\tilde{T}}\boldsymbol{w}.$$

   Since $\Pi_{\tilde{T}}\boldsymbol{w}$ is the projection of $\boldsymbol{w}$ along the large eigenspace of $\widetilde{\boldsymbol{A}}$, the second term on the right-hand side above is large, i.e. $\geq (1-10\epsilon)\lambda_0\|\Pi_{\tilde{T}}\boldsymbol{w}\|^2$. The first term on the right-hand side can be bounded as, $\boldsymbol{w}^\top \Pi_{\tilde{T}}\boldsymbol{A}\Pi_{\tilde{T}}\boldsymbol{w} \leq \|\boldsymbol{A}\|\|\Pi_{\tilde{T}}\boldsymbol{w}\|^2 \leq \lambda_0\|\Pi_{\tilde{T}}\boldsymbol{w}\|^2$. Therefore the difference on the right-hand side is at most $10\epsilon\lambda_0\|\Pi_{\tilde{T}}\boldsymbol{w}\|^2 \leq 10\epsilon\lambda_0\|\boldsymbol{w}\|^2 = 10\epsilon\lambda_0 \leq 10\epsilon(1+\epsilon)$.

2. $\boldsymbol{w}^\top \boldsymbol{V}_{\overline{T}}\boldsymbol{w}$: Observe that this term corresponds to the projection of $\boldsymbol{w}$ along the space spanned by the eigenvalues of $\boldsymbol{A}$ of size at most $1-3\epsilon$. Let $\boldsymbol{u}_i$ and λ_i denote an eigenvector and eigenvalue pair with $\lambda_i < 1-3\epsilon$. Since the power method can guarantee that $\boldsymbol{w}^\top \boldsymbol{u}_i \approx \lambda_i^{2K}$, we have $\lambda_i^{2K} \leq (1-3\epsilon)^{2K} \leq \operatorname{poly}\left(\frac{\epsilon}{n}\right)$ is tiny. So we have that $\boldsymbol{w}^\top \boldsymbol{V}_{\overline{T}}\boldsymbol{w} \leq \epsilon$ (Lemma 3.18).

[4]Suppose a subspace S is spanned by vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$. Let $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$. Recall that the projection matrix onto S is $\boldsymbol{U}(\boldsymbol{U}^\top\boldsymbol{U})^{-1}\boldsymbol{U}^\top$.

Before we look at the final case, we define a basis for the space T_ν.

Definition 3.9 (Basis for T_ν)**.** *Let T_ν be as defined in Definition 3.6. Define indices a_ν and b_ν with $b_\nu - a_\nu + 1 = d_\nu$ such that the basis of T_ν is given by $\boldsymbol{u}_{a_\nu}, \cdots, \boldsymbol{u}_{b_\nu}$, where $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_n$ are the eigenvectors of $\boldsymbol{A}$ in decreasing order of eigenvalues.*

3. $\boldsymbol{w}^\top \boldsymbol{V}_{T_\nu - \tilde{T}} \boldsymbol{w}$: For this discussion, we will ignore the constant factors and assume that the high probability events hold. Let Π_ν denote the projection matrix for the space $T_\nu - \tilde{T}$. Observe that $\boldsymbol{w}^\top \boldsymbol{V}_{T_\nu - \tilde{T}} \boldsymbol{w} = \boldsymbol{w}^\top \Pi_\nu \boldsymbol{V} \Pi_\nu \boldsymbol{w} \le \|\boldsymbol{V}\| \|\Pi_\nu \boldsymbol{w}\|^2 \le (1+\epsilon)\|\Pi_\nu \boldsymbol{w}\|^2$, where the last inequality is because $\widetilde{\boldsymbol{A}} = \boldsymbol{A} - \boldsymbol{V} \succeq 0$, and therefore, $\|\boldsymbol{V}\| \le \|\boldsymbol{A}\| \le (1+\epsilon)$. Hence, it suffices to bound $\|\Pi_\nu \boldsymbol{w}\|^2 = O(\epsilon)$.

We can write $\boldsymbol{w} = \frac{\sum_{i=1}^n \lambda_i^K \alpha_i \boldsymbol{u}_i}{\sqrt{\sum_{i=1}^n \lambda_i^{2K} \alpha_i^2}}$ where $\lambda_i, \boldsymbol{u}_i$'s are the eigenvalues and eigenvectors of $\boldsymbol{A}$ and $\alpha_i \sim N(0,1)$. Define $\boldsymbol{z} = \sum_{i=1}^n z_i \boldsymbol{u}_i$ where $z_i = \lambda_i^K \alpha_i$. That is, $\boldsymbol{w} = \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|}$. Since $\|\Pi_\nu \boldsymbol{w}\| = \|\Pi_\nu \boldsymbol{z}\|/\|\boldsymbol{z}\|$, it suffices to show that $\|\Pi_\nu \boldsymbol{z}\|^2 \le O(\epsilon)\|\boldsymbol{z}\|^2$. We show this in two separate cases. In both cases, we start with the following bound

$$\|\Pi_\nu \boldsymbol{z}\|^2 \le \lambda_{a_\nu}^{2K} \cdot \dim(T_\nu - \tilde{T}),$$

which holds with high probability. To see this, let $\boldsymbol{g}_\nu \sim N(0,1)$ be a gaussian vector in the space $T_\nu - \tilde{T}$. We can couple $\mathbf{g}_\nu$ with $\Pi_\nu \boldsymbol{z}$ so that $\Pi_\nu \boldsymbol{z}$ is dominated by $\lambda_{a_\nu}^K \cdot \boldsymbol{g}_\nu$. So $\|\Pi_\nu \boldsymbol{z}\|^2 \le \lambda_{a_\nu}^{2K} \|\boldsymbol{g}_\nu\|^2$. By Lemma 2.4, the norm square of gaussian vector is concentrated to its dimension so $\|\boldsymbol{g}_\nu\|^2 \le \dim(T_\nu - \tilde{T})$ with high probability, thus proving the inequality. Next, we will bound $\dim(T_\nu - \tilde{T})$ in terms of $\|\boldsymbol{z}\|$ in two cases.

When $\nu \notin \mathcal{I}$ (Lemma 3.21): From the definition of the important levels, we have

$$\dim(T_\nu - \tilde{T}) \le d_\nu \le \frac{O(\epsilon)}{\log^3 \frac{n}{\epsilon}} \sum_{\nu' < \nu} d_{\nu'}.$$

Now, we have $\sum_{\nu' < \nu} d_{\nu'} \approx \sum_{i=1}^{b_{\nu-1}} \alpha_i^2$ because $\alpha_i \sim N(0,1)$ is gaussian and the norm square of gaussian vector is concentrated to its dimension (Lemma 2.4). Since $\alpha_i = z_i / \lambda_i^K$, we have that

$$\sum_{\nu' < \nu} d_{\nu'} \approx \sum_{i=1}^{b_{\nu-1}} \alpha_i^2 = \sum_{i=1}^{b_{\nu-1}} \frac{z_i^2}{\lambda_i^{2K}} \le \|\boldsymbol{z}\|^2 / \lambda_{b_{\nu-1}}^{2K}.$$

Therefore, we have

$$\|\Pi_\nu \boldsymbol{z}\|^2 \le \lambda_{a_\nu}^{2K} \dim(T_\nu - \tilde{T}) \le \left(\frac{\lambda_{a_\nu}}{\lambda_{b_{\nu-1}}}\right)^{2K} \frac{O(\epsilon)}{\log^3 \frac{n}{\epsilon}} \|\boldsymbol{z}\|^2 \le O(\epsilon)\|\boldsymbol{z}\|^2$$

where the last inequality is trivial because $\lambda_{b_{\nu-1}} \ge \lambda_{a_\nu}$ by definition.

When $\nu \in \mathcal{I}$ (Lemma 3.20): In this case, according to $(\star)$, we can assume

$$\dim(T_\nu - \tilde{T}) \lesssim \epsilon d_\nu.$$

Again, by Lemma 2.4, we have that $d_\nu \approx \sum_{i=a_\nu}^{b_\nu} \alpha_i^2$ because $\alpha_i \sim N(0,1)$ is gaussian. Since $\alpha_i = z_i/\lambda_i^K$, we have

$$d_\nu \approx \sum_{i=a_\nu}^{b_\nu} \alpha_i^2 = \sum_{i=a_\nu}^{b_\nu} \frac{z_i^2}{\lambda_i^{2K}} \le \|\boldsymbol{z}\|^2/\lambda_{b_\nu}^{2K}.$$

Therefore,

$$\|\Pi_\nu \boldsymbol{z}\|^2 \le \lambda_{a_\nu}^{2K} \dim(T_\nu - \tilde{T}) \le \left(\frac{\lambda_{a_\nu}}{\lambda_{b_\nu}}\right)^{2K} \epsilon \|\boldsymbol{z}\|^2 \le O(\epsilon)\|z\|^2$$

where the last inequality is because $\left(\frac{\lambda_{a_\nu}}{\lambda_{b_\nu}}\right)^{2K} \le \left(\frac{1-\frac{\nu\epsilon}{5\log\frac{n}{\epsilon}}}{1-\frac{(\nu+1)\epsilon}{5\log\frac{n}{\epsilon}}}\right)^{2K} \approx \left(1+\frac{\epsilon}{2\log\frac{n}{\epsilon}}\right)^{2K} \approx O(1)$.

From these three cases, we can conclude that if $\dim(T_\nu - \tilde{T})$ is small for all $\nu \in \mathcal{I}$, then $\boldsymbol{w}^\top \boldsymbol{V} \boldsymbol{w} \le 35\epsilon$, proving our claim.

In the remaining sections, we give formal proofs of the claims made in this section. In Section 3.2, we prove the main result, Theorem 1.2, assuming the key lemma. In Section 3.3, we prove the key lemma, Lemma 3.5, assuming the Lemma 3.8. Finally, we prove Lemma 3.8 in Section 3.4

3.2 Proof of the Main Theorem 1.2 assuming the Key Lemma 3.5

Here, we formally prove Theorem 1.2 assuming the key Lemma 3.5. We will first prove the correctness and then bound the total runtime.

Correctness. The following formalizes the correctness guarantee of Algorithm 3.

Lemma 3.10. *Let $\epsilon > 1/n$. With probability at least $1-1/n$, the following holds for all time step $t \ge 1$: if the maximum eigenvalue of $\boldsymbol{A}_t$ is at least $1 - \frac{\epsilon}{\log n}$,* Update*$(\boldsymbol{v}_t)$ returns $[r_t, \boldsymbol{W}_t]$.*

Proof. TOPROVE 2 ☐

Runtime. Next, we bound the runtime of the various lines of Algorithm 3.

Lemma 3.11. *For a fixed $\boldsymbol{w}$ and any t, we can update $\boldsymbol{w}^\top \boldsymbol{A}_{t-1}\boldsymbol{w}$ to $\boldsymbol{w}^\top \boldsymbol{A}_t \boldsymbol{w}$ in time $O(nnz(\boldsymbol{v}_t))$.*

Proof. TOPROVE 3 ☐

Lemma 3.12. *Fix time t. Given $\boldsymbol{w}$ as input, we have that $\boldsymbol{w}^\top \boldsymbol{A}_t \boldsymbol{w}$ and $\boldsymbol{A}_t \boldsymbol{w}$ can be computed $O\Big(nnz(\boldsymbol{A}_0) + \sum_{i=1}^t nnz(\boldsymbol{v}_i)\Big)$ time.*

Proof. TOPROVE 4 ☐

Lemma 3.13. *For any time t, we can implement* PowerMethod*$(\epsilon, \boldsymbol{A}_t)$ in time at most*

$$O\left(\frac{\log n \log\frac{n}{\epsilon}}{\epsilon}\left(nnz(\boldsymbol{A}_0) + \sum_{i=1}^t nnz(\boldsymbol{v}_i)\right)\right).$$

Proof. TOPROVE 5 □

Given the above results, we can now prove Theorem 1.2.

Proof of Theorem 1.2

Proof. TOPROVE 6 □

3.3 Proof of the Key Lemma 3.5: Few Executions of Line 4

In this section, we prove Lemma 3.5 assuming the Progress Lemma 3.8. Let us first recall the precise definition of $\boldsymbol{A}$ and $\widetilde{\boldsymbol{A}}$. Suppose we execute Line 4 at update t_0. Now consider a sequence of updates, $\boldsymbol{v}_{t_0+1}, \cdots, \boldsymbol{v}_{t_0+k}$, and let $\boldsymbol{A}_{t_0+k} = \boldsymbol{A}_{t_0} - \sum_{i=1}^{k} \boldsymbol{v}_{t+i}\boldsymbol{v}_{t+i}^\top$. Suppose the next execution of Line 4 happens at t_0+k. For this to happen we must have that $\boldsymbol{w}_{t_0}^\top \boldsymbol{A}_{t_0+k}\boldsymbol{w}_{t_0} < 1-40\epsilon$ for all $\boldsymbol{w}_{t_0} \in \boldsymbol{W}_{t_0}$. We let $\boldsymbol{A} = \boldsymbol{A}_{t_0}$ and $\widetilde{\boldsymbol{A}} = \boldsymbol{A}_{t_0+k}$.

Next, recall Definition 3.6 and define $T_{\leq i} = \sum_{\nu=0}^{i} T_\nu$, $\tilde{T}_{\leq i} = \sum_{\nu=0}^{i} \tilde{T}_\nu$. The following observation will motivate our potential function analysis.

Lemma 3.14. *For all $i \leq 15\log\frac{n}{\epsilon}$,*

$$\dim(T_{\leq i}) \geq \dim(\tilde{T}_{\leq i}).$$

Let ν_0 be such that $\dim(T_{\nu_0} - \tilde{T}) > 0$. For any $i \geq \nu_0$, we have

$$\dim(T_{\leq i}) \geq \dim(\tilde{T}_{\leq i}) + \dim(T_{\nu_0} - \tilde{T}).$$

Proof. TOPROVE 7 □

The Potentials. The above lemma and Lemma 3.8 motivate the following potentials. For every $j \leq 15\log\frac{n}{\epsilon}$, we define the potentials $\Phi_j = \dim(T_{\leq j})$ and $\tilde{\Phi}_j = \dim(\tilde{T}_{\leq j})$. For all j, the potential may only decrease, i.e., $\tilde{\Phi}_j \leq \Phi_j$ by Lemma 3.14. Also, clearly, $\Phi_j \leq n$.

We will show that for each execution of Line 4, $\tilde{\Phi}_j$ decreases from Φ_j by a significant factor with high probability for some j. This will bound the number of executions.

Consider any important level $\nu_0 \in \mathcal{I}$. We have two observations:

1. $\tilde{\Phi}_{\nu_0} \leq \Phi_{\nu_0} - \dim(T_{\nu_0} - \tilde{T})$, and
2. $d_{\nu_0} \geq \Omega(1)\frac{\epsilon}{\log^3\frac{n}{\epsilon}}\Phi_{v_0}$.

The first point follows directly from the second part of Lemma 3.14. Moreover, since $\nu_0 \in \mathcal{I}$ is important, we have $d_{\nu_0} \geq \frac{\epsilon}{600\log^3\frac{n}{\epsilon}}\sum_{\nu'<\nu} d_{\nu'}$. Therefore,

$$\Phi_{\nu_0} = \sum_{\nu=1}^{\nu_0} d_\nu = \sum_{\nu'<\nu_0} d_{\nu'} + d_{\nu_0} \leq \left(\frac{600\log^3\frac{n}{\epsilon}}{\epsilon}+1\right)d_{\nu_0},$$

implying the second point. Combining these observations with the Progress Lemma 3.8, we have:

Lemma 3.15. *Suppose that $\lambda_{\max}(\boldsymbol{A}) \geq 1-\epsilon$ and $\boldsymbol{w}^\top\widetilde{\boldsymbol{A}}\boldsymbol{w} < 1-40\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$. With probability at least $1-50\log\frac{n}{\epsilon}/n^2$, there is a level ν_0 such that either*

- $\tilde{\Phi}_{\nu_0} \leq \left(1 - \frac{\Omega(\epsilon^2)}{\log^4 \frac{n}{\epsilon}}\right)\Phi_{\nu_0}$, *or*

- $\tilde{\Phi}_{\nu_0} \leq \Phi_{\nu_0} - 1$ *and* $\Phi_{\nu_0} \leq O(1)\frac{\log^4 \frac{n}{\epsilon} \log n}{\epsilon^2}$.

Proof. TOPROVE 8 $\square$

We are now ready to prove Lemma 3.5.

Proof of Lemma 3.5.

First, observe that whenever $\lambda_{\max}(\boldsymbol{A}) < 1-\epsilon$, by Line 10 of Algorithm 1, we will always return FALSE at the next execution of Line 4 of Algorithm 3.

Therefore, it suffices to bound the number of executions while $\lambda_{\max}(\boldsymbol{A}) \geq 1-\epsilon$. We execute Line 4 only if $\boldsymbol{w}^\top \widetilde{\boldsymbol{A}}\boldsymbol{w} < 1 - 40\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$. When this happens, there exists a level j where the potential Φ_j significantly decreases according to Lemma 3.15 with probability at least $1 - 50\log\frac{n}{\epsilon}/n^2$. For each level j, this can happen at most $L = O(\frac{\log n \log^4 \frac{n}{\epsilon}}{\epsilon^2})$ times because for every j, Φ_j is an integer that may only decrease and is bounded by n. Suppose for contradiction that there are more than $L \times 15 \log\frac{n}{\epsilon}$ executions of Line 4. So, with probability at least $1 - L\cdot 50 \log\frac{n}{\epsilon}/n^2 \geq 1/n$, there exists a level j where Φ_j decreases according to Lemma 3.15 strictly more than L times. This is a contradiction.

3.4 Proof of the Progress Lemma 3.8

It remains to prove the Progress Lemma. We first restate the lemma here.

Lemma 3.8 (Measure of Progress). *Let $\epsilon > 0$ and let $\boldsymbol{W} = [\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(R)}]$ be as defined in Line 9 in the execution of* POWERMETHOD*$(\epsilon, \boldsymbol{A})$. Let $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k$ be a sequence of updates generated by an oblivious adversary and define $\widetilde{\boldsymbol{A}} = \boldsymbol{A} - \sum_{i=1}^k \boldsymbol{v}_i\boldsymbol{v}_i^\top$.*

Suppose that $\lambda_{\max}(\boldsymbol{A}) \geq 1-\epsilon$ and $\boldsymbol{w}^\top\widetilde{\boldsymbol{A}}\boldsymbol{w} < 1-40\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$. Then, with probability at least $1 - \frac{50\log\frac{n}{\epsilon}}{n^2}$, for some $\nu \in \mathcal{I}$,

- $\dim(T_\nu - \tilde{T}) \geq \frac{\epsilon}{300\log\frac{n}{\epsilon}} d_\nu$ *if* $d_\nu \geq \frac{3000 \log n \log\frac{n}{\epsilon}}{\epsilon}$, *or*

- $\dim(T_\nu - \tilde{T}) \geq 1$ *if* $d_\nu < \frac{3000 \log n \log\frac{n}{\epsilon}}{\epsilon}$.

Recall the definitions of subspaces $T, T_\nu, \tilde{T}$ and $\overline{T}$ in Definition 3.6. We will first state the following claim and show that this is sufficient to prove Lemma 3.8. After concluding the proof of Lemma 3.8, we would prove the claim.

Claim 3.16. *Suppose that $\lambda_{\max}(\boldsymbol{A}) \geq 1-\epsilon$. If for every $\nu \in \mathcal{I}$,*

- $\dim(T_\nu - \tilde{T}) < \frac{\epsilon}{300\log\frac{n}{\epsilon}} d_\nu$ *if* $d_\nu \geq \frac{3000 \log n \log\frac{n}{\epsilon}}{\epsilon}$, *and*

- $\dim(T_\nu - \tilde{T}) < 1$ *if* $d_\nu < \frac{3000 \log n \log\frac{n}{\epsilon}}{\epsilon}$.

Then, with probability at least $1 - \frac{40\log\frac{n}{\epsilon}}{n^2}$, $\boldsymbol{w}^\top\boldsymbol{V}\boldsymbol{w} \leq 35\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$.

Proof of Lemma 3.8 using Claim 3.16

Suppose for contradiction that Lemma 3.8 does not hold, i.e., the conditions on $\dim(T_\nu - \tilde{T})$ of Claim 3.16 hold for all $\nu \in \mathcal{I}$. On one hand, since $\lambda_{\max}(\boldsymbol{A}) \geq 1 - \epsilon$, Claim 3.16 says that $\boldsymbol{w}^\top \boldsymbol{V} \boldsymbol{w} \leq 35\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$ with probability at least $1 - \frac{40 \log \frac{n}{\epsilon}}{n^2}$. From the assumption of Lemma 3.8, we have $\boldsymbol{w}^\top \widetilde{\boldsymbol{A}} \boldsymbol{w} < 1 - 40\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$ as well, this implies that, for all $\boldsymbol{w} \in \boldsymbol{W}$,

$$\boldsymbol{w}^\top \boldsymbol{A} \boldsymbol{w} < \boldsymbol{w}^\top \widetilde{\boldsymbol{A}} \boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{V} \boldsymbol{w} < 1 - 5\epsilon.$$

On the other hand, since $\lambda_{\max}(\boldsymbol{A}) \geq 1 - \epsilon$, we must have $\boldsymbol{w}^\top \boldsymbol{A} \boldsymbol{w} \geq 1 - 5\epsilon$ for some $\boldsymbol{w} \in \boldsymbol{W}$ with probability at least $1 - 1/n^2$ by Corollary 3.4. This gives a contradiction.

Therefore, we conclude that, with probability at least $1 - \frac{40 \log \frac{n}{\epsilon} + 1}{n^2}$, the conditions of Claim 3.16 must be false for some $\nu \in \mathcal{I}$. That is, we have

- $\dim(T_\nu - \tilde{T}) \geq \frac{\epsilon}{300 \log \frac{n}{\epsilon}} d_\nu$ if $d_\nu \geq \frac{3000 \log n \log \frac{n}{\epsilon}}{\epsilon}$, and
- $\dim(T_\nu - \tilde{T}) \geq 1$ if $d_\nu < \frac{3000 \log n \log \frac{n}{\epsilon}}{\epsilon}$.

This concludes the proof of Lemma 3.8.

Setting Up for the Proof of Claim 3.16

Recall the definitions of subspaces T, T_ν, $\tilde{T}$ and $\overline{T}$ in Definition 3.6.

Proposition 3.17. *We can cover the entire space with the following subspaces*

$$\mathbb{R}^n = T + \overline{T} = \tilde{T} + \sum_{\nu=0}^{15 \log \frac{n}{\epsilon} - 1} (T_\nu - \tilde{T}) + \overline{T} \tag{3}$$

where all subspaces in the sum are mutually orthogonal.

Proof. TOPROVE 9 □

Notation. The goal of Claim 3.16 is to bound $\boldsymbol{w}^\top \boldsymbol{V} \boldsymbol{w} \leq 35\epsilon$ for all $\boldsymbol{w} \in \boldsymbol{W}$. We will use the following notations.

- Let $\Pi_{\tilde{T}}, \Pi_\nu, \Pi_{\overline{T}}$ denote projection matrices to the subspaces $\tilde{T}$, $T_\nu - \tilde{T}$, and $\overline{T}$ respectively.
- Define $\boldsymbol{V}_{\tilde{T}} = \Pi_{\tilde{T}} \boldsymbol{V} \Pi_{\tilde{T}}$, $\boldsymbol{V}_{T_\nu - \tilde{T}} = \Pi_\nu \boldsymbol{V} \Pi_\nu$, and $\boldsymbol{V}_{\overline{T}} = \Pi_{\overline{T}} \boldsymbol{V} \Pi_{\overline{T}}$.

By Proposition 3.17, for any $\boldsymbol{w} \in \boldsymbol{W}$, we can decompose $\boldsymbol{w}^\top \boldsymbol{V} \boldsymbol{w}$ as

$$\boldsymbol{w}^\top \boldsymbol{V} \boldsymbol{w} = \boldsymbol{w}^\top \boldsymbol{V}_{\tilde{T}} \boldsymbol{w} + \sum_{\nu=0}^{15 \log \frac{n}{\epsilon} - 1} \boldsymbol{w}^\top \boldsymbol{V}_{T_\nu - \tilde{T}} \boldsymbol{w} + \boldsymbol{w}^\top \boldsymbol{V}_{\overline{T}} \boldsymbol{w}.$$

Our strategy is to upper bound each term one by one. Bounding $\boldsymbol{w}^\top \boldsymbol{V}_{\tilde{T}} \boldsymbol{w}$ is straightforward, but bounding other terms requires technical helper lemmas. Lemma 3.18 is needed for bounding $\boldsymbol{w}^\top \boldsymbol{V}_{\overline{T}} \boldsymbol{w}$. Lemmas 3.20 and 3.21 are helpful for bounding $\boldsymbol{w}^\top \boldsymbol{V}_{T_\nu - \tilde{T}} \boldsymbol{w}$ when $\nu \in \mathcal{I}$ and when $\nu \notin \mathcal{I}$, respectively.

Helper Lemmas for Claim 3.16

In all the statements of the helper lemmas below. Let $\boldsymbol{W}$ be as defined in Line 9 in the execution of PowerMethod$(\epsilon, \boldsymbol{A})$. Consider any fixed $\boldsymbol{w} \in \boldsymbol{W}$. Observe that we can write

$$\boldsymbol{w} = \sum_{i=1}^{n} \frac{\lambda_i^K \alpha_i \boldsymbol{u}_i}{\sqrt{\sum_j \lambda_j^{2K} \alpha_j^2}} \tag{4}$$

where $K = \frac{4 \log \frac{n}{\epsilon}}{\epsilon}$, $\alpha_i \sim N(0,1)$ are gaussian random variables, and λ_i and $\boldsymbol{u}_i$ are the i-th eigenvalue and eigenvector of $\boldsymbol{A}$, respectively.

The following lemma shows that the projection of $\boldsymbol{w}$ on $\overline{T}$ is always small. At a high level, since $\overline{T}$ is spanned by the eigenvectors with the small eigenvalues, the power method guarantees with high probability that the direction of $\boldsymbol{w}$ along these eigenvectors will be exponentially small in the number of iterations K. Recall that $\Pi_{\overline{T}}$ is the projection matrix to the space $\overline{T}$.

Lemma 3.18. *If $\lambda_{\max}(\boldsymbol{A}) \geq 1 - \epsilon$, then*

$$\Pr\left[\|\Pi_{\overline{T}} \boldsymbol{w}\|^2 \leq \frac{\epsilon^2}{4} \right] \geq 1 - \frac{3}{n^2}.$$

Proof. TOPROVE 10 □

The next two helper lemmas are to show that the projection of $\boldsymbol{w}$ to $T_\nu - \tilde{T}$ is small. To do this, we introduce some more notations and one proposition. For any level ν, we will use q_ν to denote,

$$q_\nu \overset{\text{def}}{=} \dim(T_\nu - \tilde{T}). \tag{5}$$

Let

$$\boldsymbol{z} = \sum_i z_i \boldsymbol{u}_i \text{ where } z_i = \lambda_i^K \alpha_i. \tag{6}$$

Therefore, $\boldsymbol{w} = \boldsymbol{z}/\|\boldsymbol{z}\|$.

We now bound the norm of the projection of $\boldsymbol{z}$ to $T_\nu - \tilde{T}$. The proof is based on the fact that $\boldsymbol{z}$ is a *scaled* gaussian random vector, and the projection of a gaussian on a q_ν-dimensional subspace should have norm proportional to q_ν.

Proposition 3.19. *For $\boldsymbol{z}$ as defined in (6), we have*

$$\|\Pi_\nu \boldsymbol{z}\|^2 \leq \lambda_{a_\nu}^{2K} \cdot \sum_{j=a_\nu}^{b_\nu} \alpha_j^2.$$

Furthermore, if $q_\nu \geq 10 \log n$, then with probability at least $1 - 1/n^2$,

$$\|\Pi_\nu \boldsymbol{z}\|^2 \leq 2 q_\nu \cdot \lambda_{a_\nu}^{2K}.$$

Proof. TOPROVE 11 □

The following lemma shows that the projection of $\boldsymbol{w}$ on $T_\nu - \tilde{T}$ is small when $\dim(T_\nu - \tilde{T}) := q_\nu$ is roughly at most an ϵ-factor of $\dim(T_\nu) := d_\nu$, and q_ν is still at least logarithmic. We will use this lemma to characterize the projection of $\boldsymbol{w}$ on T_ν for $\nu \in \mathcal{I}$. Recall that Π_ν is a projection matrix that projects any vector to the space $T_\nu - \tilde{T}$.

Lemma 3.20. *If* $10 \log n \le q_\nu \le \frac{\epsilon}{300 \log \frac{n}{\epsilon}} d_\nu$, *then*

$$\Pr\left[\left\|\Pi_\nu \boldsymbol{w}\right\|^2 \le \frac{\epsilon}{\log \frac{n}{\epsilon}}\right] \ge 1 - \frac{2}{n^2}.$$

Proof. TOPROVE 12 □

We next prove that for all $\nu \notin \mathcal{I}$, arbitrary projections of $\boldsymbol{w}$ on $T_\nu - \tilde{T}$ are always small. This proof uses a similar idea as that of the previous lemma and the main difference is that we can use the fact that d_ν is small for $\nu \notin \mathcal{I}$ to additionally show that the projection of $\boldsymbol{w}$ is small even for small dimensional arbitrary subspaces of T_ν. Recall that Π_ν is a projection matrix to the space $T_\nu - \tilde{T}$.

Lemma 3.21. *For any non-important level,* $\nu \notin \mathcal{I}$,

$$\Pr\left[\left\|\Pi_\nu \boldsymbol{w}\right\|^2 \le \frac{\epsilon}{\log \frac{n}{\epsilon}}\right] \ge 1 - \frac{1}{n^2}.$$

Proof. TOPROVE 13 □

Proof of Claim 3.16. We are now ready to finally prove Claim 3.16.

Proof. TOPROVE 14 □

4 Conditional Lower Bounds for an Adaptive Adversary

In this section, we will prove a conditional hardness result for algorithms against adaptive adversaries. In particular, we will prove Theorem 1.6. Consider Algorithm 4 for solving Problem 1.3. The only step in Algorithm 4 whose implementation is not specified is Line 8. We will implement this step using an algorithm for Problem 1.1.

High-level idea. Overall for our hardness result, we use the idea that an adaptive adversary can use the maximum eigenvectors returned to perform an update. This can happen n times and in the process, we would recover the entire eigen-decomposition of the matrix, which is hard. Now consider Algorithm 4. We claim that Algorithm 4 solves Problem 1.3. At the first glance, this claim looks suspicious because the input matrix for Problem 1.3 might not be PSD, but the dynamic algorithm for Problem 1.1 at Line 8 has any guarantees only when the matrices remain PSD. However, the reduction does work by crucially exploiting Property 1.5. The high-level idea is as follows.

Algorithm 4 Algorithm for Checking PSDness

```
1: procedure CheckPSD($\delta, \kappa, \boldsymbol{A}$)
2:     $\epsilon \leftarrow \min\{1 - n^{-o(1)}, (1-\delta)/(1+\delta)\}$
3:     $T \leftarrow \frac{2n}{\epsilon(1-\epsilon)^2} \log \frac{\kappa}{\delta}$
4:     $\boldsymbol{A}_0 \leftarrow \boldsymbol{A}$
5:     $\mu_0 = 0, \boldsymbol{w}_0 = 0$
6:     for $t = 1, 2, \cdots, T$ do
7:         $\boldsymbol{A}_t = \boldsymbol{A}_{t-1} - \frac{\mu_{t-1}}{10} \boldsymbol{w}_{t-1} \boldsymbol{w}_{t-1}^\top$
8:         $(\mu_t, \boldsymbol{w}_t) \leftarrow \epsilon$-approximate maximum eigenvalue and eigenvector of $\boldsymbol{A}_t$ (Equations (1),(2))
9:         if $\mu_t < 0$ then
10:            return False:$\boldsymbol{A}$ is not PSD
11:    $\sigma^2 \leftarrow$PowerMethod$(\epsilon, \boldsymbol{A}_T^\top \boldsymbol{A}_T)$
12:    if $0 \leq \sigma \leq \frac{(1+\epsilon)\mu_1 \delta}{\kappa}$ then
13:        return $\boldsymbol{X} = \frac{1}{\sqrt{10}} \begin{bmatrix} \sqrt{\mu_1} \boldsymbol{w}_1 & \sqrt{\mu_2} \boldsymbol{w}_2 & \cdots & \sqrt{\mu_T} \boldsymbol{w}_T \end{bmatrix}$
14:    else
15:        return False:$\boldsymbol{A}$ is not PSD
```

- If the input matrix $\boldsymbol{A}$ is initially PSD, then we can show that $\boldsymbol{A}_t$ remains PSD for all t by exploiting Property 1.5, (see Lemma 4.1). So, the approximation guarantee of the algorithm at Line 8 is valid at all steps. From this guarantee, $\|\boldsymbol{A}_T\|$ must be tiny since we keep decreasing the approximately maximum eigenvalues (see Lemma 4.2). At the end, the reduction will return $\boldsymbol{X}$.

- If the input matrix $\boldsymbol{A}$ is initially *not* PSD, there must exist a direction $\boldsymbol{v}$ such that $\boldsymbol{v}^\top \boldsymbol{A} \boldsymbol{v} < 0$. Since in the reduction, we update $\boldsymbol{A}_T = \boldsymbol{A} - \boldsymbol{W}$ for some $\boldsymbol{W} \succeq 0$, we must have that $\boldsymbol{v}^\top \boldsymbol{A}_T \boldsymbol{v} < \boldsymbol{v}^\top \boldsymbol{A} \boldsymbol{v}$. That is, this negative direction remains negative or gets even more negative. It does not matter at all what guarantees the algorithm at Line 8 has. We still have that $\|\boldsymbol{A}_T\|$ cannot be tiny. We can distinguish whether $\|\boldsymbol{A}_T\|$ is tiny or not using the static power method at Line 11, and, hence, we will return False in this case (see Lemma 4.2).

Analysis. We prove the guarantees of the output of Algorithm 4 when $\boldsymbol{w}_t$'s satisfy Property 1.5 for all t.

Lemma 4.1. *In Algorithm 4, let $\boldsymbol{w}_t$'s, $t = 1, \cdots, T$ be generated such that they additionally satisfy Property 1.5. If $\boldsymbol{A}_0 \succeq 0$, then $\boldsymbol{A}_t \succeq 0$ for all t.*

We would like to point out that our parameter ϵ is quite large. This just implies that our reduction can work even if we find crude approximations to the maximum eigenvector as long as this is along the directions with large eigenvalue, since $\boldsymbol{w}$ also has to satisfy Property 1.5.

Proof. TOPROVE 15 □

Lemma 4.2. *In Algorithm 4, let $\boldsymbol{w}_t$'s, $t = 1, \cdots, T$ be generated such that they additionally satisfy Property 1.5.*

- *If $\boldsymbol{A} \succeq 0$, then Algorithm 4 returns $\boldsymbol{X}$ such that $\|\boldsymbol{A} - \boldsymbol{X}\boldsymbol{X}^\top\| \leq \delta \min_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|$.*

- *If $\boldsymbol{A}$ is not psd, then Algorithm 4 returns* False.

Proof. TOPROVE 16 □

Proof of Theorem 1.6. We are now ready to prove our conditional lower bound.

Proof. TOPROVE 17 □

5 Conclusion and Open Problems

Upper Bounds. We have presented a novel extension of the power method to the dynamic setting. Our algorithm from Theorem 1.2 maintains a multiplicative $(1+\epsilon)$-approximate maximum eigenvalue and eigenvector of a positive semi-definite matrix that undergoes decremental updates from an oblivious adversary. The algorithm has polylogarithmic amortized update time per non-zeros in the updates.

Our algorithm is simple, but our analysis is quite involved. While we believe a tighter analysis that improves our logarithmic factors is possible, it is an interesting open problem to give a simpler analysis for our algorithm. Other natural questions are whether we can get similar algorithms in incremental or fully dynamic settings and whether one can get a worst-case update time.

Lower Bounds. We have shown a conditional lower bound for a class of algorithms against an adaptive adversary in Theorem 1.6. It would also be very exciting to generalize our lower bound to hold for any algorithm against an adaptive adversary, as that would imply an oblivious-vs-adaptive separation for a natural dynamic problem.

Incremental Updates. We believe that the corresponding incremental updates problem, i.e., we update the matrix as $\boldsymbol{A}_t \leftarrow \boldsymbol{A}_{t-1} + \boldsymbol{v}_t \boldsymbol{v}_t^\top$ cannot be solved in polylogarithmic amortized update time, even when the update sequence $\boldsymbol{v}_t$'s are from an oblivious adversary. At a high level, the incremental version of our problem seems significantly harder for the following reasons. When we perform decremental updates to a matrix, the new maximum eigenvector must be a part of the eigenspace spanned by the original maximum eigenvectors. Furthermore, it is easy to detect whether the maximum eigenvalue has gone down as we have shown in our paper. For the incremental setting, it is possible that after an update the maximum eigenvalue has gone up and the new maximum eigenvector is a direction that was not the previous one or the update direction and in such cases we cannot really detect this quickly with known information on previous eigenvectors and update directions. This can also happen n times and in every such case, we have to compute the eigenvalue and eigenvector from scratch. Therefore, we leave lower bounds and algorithms for incremental setting as an open problem.

Dynamic SDPs. As discussed in Appendix A, Theorem 1.2 can be viewed as a starting point towards a dynamic algorithm for general positive semi-definite programs. Can we make further progress? The dynamic semi-definite program problem, even with just two matrix constraints, already seems to be challenging.

One promising approach to attack this problem is to dynamize the matrix multiplicative weight update (MMWU) method for solving a packing/covering SDP [PTZ12] since the corresponding approach was successful for linear programs – the near-optimal algorithms of [BKS23] are essentially dynamized multiplicative weight update methods for positive linear programs. However, in our preliminary study exploring this approach, we could only obtain an algorithm that solves Problem A.2, which has a single matrix constraint, and solves Problem 1.1 partially, i.e., maintains an approximate eigenvalue only. The main barrier in this approach is that the algorithm requires maintaining the exponential of the sum of the constraint matrices, and to do this fast, we require that for any two constraint matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, $e^{\boldsymbol{A}+\boldsymbol{B}} = e^{\boldsymbol{A}}e^{\boldsymbol{B}}$ which only holds when $\boldsymbol{A}$ and $\boldsymbol{B}$ commute i.e., $\boldsymbol{A}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{A}$. Note that when $\boldsymbol{A}$ and $\boldsymbol{B}$ are diagonal, this is true; therefore, we can obtain the required algorithms for positive LPs. Even when we have just two constraint matrices where one of them is a diagonal matrix, this remains an issue as the matrices still do not commute.

References

[AGG88] P. Arbenz, W. Gander, and G. H. Golub. “Restricted rank modification of the symmetric eigenvalue problem: Theoretical considerations”. In: *Linear Algebra and its Applications* 104 (1988), pp. 75–95 (cit. on p. 1).

[AK07] S. Arora and S. Kale. “A combinatorial, primal-dual approach to semidefinite programs”. In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 227–236 (cit. on p. 24).

[AL17] Z. Allen-Zhu and Y. Li. “First efficient convergence for streaming k-pca: a global, gap-free, and near-optimal rate”. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 487–492 (cit. on p. 1).

[ALO16] Z. Allen-Zhu, Y. T. Lee, and L. Orecchia. “Using optimization to obtain a width-independent, parallel, simpler, and faster positive SDP solver”. In: *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2016, pp. 1824–1831 (cit. on p. 23).

[AO15] Z. Allen-Zhu and L. Orecchia. “Nearly-linear time positive LP solver with faster convergence rate”. In: *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*. 2015, pp. 229–236 (cit. on p. 23).

[AO19] Z. Allen-Zhu and L. Orecchia. “Nearly linear-time packing and covering lp solvers: Achieving width-independence and-convergence”. In: *Mathematical Programming* 175 (2019), pp. 307–353 (cit. on p. 23).

[Ban+22] J. Banks, J. Garza-Vargas, A. Kulkarni, and N. Srivastava. “Pseudospectral shattering, the sign function, and diagonalization in nearly matrix multiplication time”. In: *Foundations of Computational Mathematics* (2022), pp. 1–89 (cit. on p. 3).

[Bat+23] M. Bateni, H. Esfandiari, H. Fichtenberger, M. Henzinger, R. Jayaram, V. Mirrokni, and A. Wiese. “Optimal Fully Dynamic k-Center Clustering for Adaptive and Oblivious Adversaries”. In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2023, pp. 2677–2727 (cit. on p. 4).

[Bei+22] A. Beimel, H. Kaplan, Y. Mansour, K. Nissim, T. Saranurak, and U. Stemmer. “Dynamic algorithms against an adaptive adversary: Generic constructions and lower bounds”. In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 1671–1684 (cit. on p. 4).

[Bha+23] R. Bhattacharjee, G. Dexter, C. Musco, A. Ray, and D. P. Woodruff. “Universal Matrix Sparsifiers and Fast Deterministic Algorithms for Linear Algebra”. In: *arXiv preprint arXiv:2305.05826* (2023) (cit. on p. 1).

[BKS23] S. Bhattacharya, P. Kiss, and T. Saranurak. “Dynamic algorithms for packing-covering LPs via multiplicative weight updates”. In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2023, pp. 1–47 (cit. on pp. 2, 18, 23).

[BNS19] J. van den Brand, D. Nanongkai, and T. Saranurak. “Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds”. In: *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2019, pp. 456–480 (cit. on p. 1).

[BNS78] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen. “Rank-one modification of the symmetric eigenproblem”. In: *Numerische Mathematik* 31.1 (1978), pp. 31–48 (cit. on p. 1).

[Gol73] G. H. Golub. “Some modified matrix eigenvalue problems”. In: *SIAM review* 15.2 (1973), pp. 318–334 (cit. on p. 1).

[IPS10] G. Iyengar, D. J. Phillips, and C. Stein. “Feasible and accurate algorithms for covering semidefinite programs”. In: *Scandinavian Workshop on Algorithm Theory*. Springer. 2010, pp. 150–162 (cit. on p. 23).

[IPS11] G. Iyengar, D. J. Phillips, and C. Stein. “Approximating semidefinite packing programs”. In: *SIAM Journal on Optimization* 21.1 (2011), pp. 231–268 (cit. on p. 23).

[Jam+21] A. Jambulapati, Y. T. Lee, J. Li, S. Padmanabhan, and K. Tian. *Positive Semidefinite Programming: Mixed, Parallel, and Width-Independent*. 2021. arXiv: 2002.04830 [cs.DS] (cit. on pp. 23, 24).

[KL96] P. Klein and H.-I. Lu. “Efficient approximation algorithms for semidefinite programs arising from MAX CUT and COLORING”. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 338–347 (cit. on p. 23).

[LM00] B. Laurent and P. Massart. “Adaptive estimation of a quadratic functional by model selection”. In: *Annals of statistics* (2000), pp. 1302–1338.

[LN93] M. Luby and N. Nisan. “A parallel approximation algorithm for positive linear programming”. In: *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*. 1993, pp. 448–457 (cit. on p. 23).

[LS17] Y. T. Lee and H. Sun. “An sdp-based algorithm for linear-sized spectral sparsification”. In: *Proceedings of the 49th annual acm sigact symposium on theory of computing*. 2017, pp. 678–687 (cit. on p. 24).

[MSS19] R. Mitz, N. Sharon, and Y. Shkolnisky. “Symmetric rank-one updates from partial spectrum with an application to out-of-sample extension”. In: *SIAM Journal on Matrix Analysis and Applications* 40.3 (2019), pp. 973–997 (cit. on p. 1).

[OSV12] L. Orecchia, S. Sachdeva, and N. K. Vishnoi. “Approximating the exponential, the Lanczos method and an O (m)-time spectral algorithm for balanced separator”. In: *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 2012, pp. 1141–1160 (cit. on p. 23).

[PST95] S. A. Plotkin, D. B. Shmoys, and É. Tardos. “Fast approximation algorithms for fractional packing and covering problems”. In: *Mathematics of Operations Research* 20.2 (1995), pp. 257–301 (cit. on p. 23).

[PTZ12] R. Peng, K. Tangwongsan, and P. Zhang. “Faster and simpler width-independent parallel algorithms for positive semidefinite programming”. In: *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*. 2012, pp. 101–108 (cit. on pp. 18, 23).

[Qua20] K. Quanrud. “Nearly linear time approximations for mixed packing and covering problems without data structures or randomization”. In: *Symposium on Simplicity in Algorithms*. SIAM. 2020, pp. 69–80 (cit. on p. 23).

[San04] P. Sankowski. "Dynamic transitive closure via dynamic matrix inverse". In: *45th Annual IEEE Symposium on Foundations of Computer Science*. IEEE. 2004, pp. 509–517 (cit. on p. 1).

[Sta08] P. Stange. "On the efficient update of the singular value decomposition". In: *PAMM: Proceedings in Applied Mathematics and Mechanics*. Vol. 8. 1. Wiley Online Library. 2008, pp. 10827–10828 (cit. on p. 1).

[SW23] W. Swartworth and D. P. Woodruff. "Optimal Eigenvalue Approximation via Sketching". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 145–155 (cit. on p. 1).

[TB22] L. N. Trefethen and D. Bau. *Numerical linear algebra*. Vol. 181. Siam, 2022 (cit. on p. 2).

[Tre09] L. Trevisan. "Max cut and the smallest eigenvalue". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 263–272 (cit. on p. 24).

[Tre98] L. Trevisan. "Parallel approximation algorithms by positive linear programming". In: *Algorithmica* 21.1 (1998), pp. 72–88 (cit. on p. 23).

[WRM16] D. Wang, S. Rao, and M. W. Mahoney. "Unified Acceleration Method for Packing and Covering Problems via Diameter Reduction". In: *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2016 (cit. on p. 23).

A Connections to Dynamic Positive Semi-definite Programs

This section discusses connections between our Problem 1.1 and the dynamic versions of positive semi-definite programs. Using this connection, we conclude that Theorem 1.2 implies a dynamic algorithm for a special case of the dynamic covering SDP problem.

We first define packing and covering semi-definite programs (SDPs).[5]

Definition A.1 (Packing/Covering SDP)**.** *Let $\boldsymbol{C}$, $\boldsymbol{A}_i$'s for $i = 1, 2, ..., m$ be $n \times n$ symmetric PSD matrices and b_i's denote positive real numbers. The packing SDP problem asks to find*

$$\max_{\boldsymbol{Y} \succeq 0} \mathrm{Tr}[\boldsymbol{C}\boldsymbol{Y}] \quad s.t. \quad \mathrm{Tr}[\boldsymbol{A}_i \boldsymbol{Y}] \leq b_i, \quad \forall i = 1, \cdots, m.$$

The covering SDP problem asks to find

$$\min_{\boldsymbol{Y} \succeq 0} \mathrm{Tr}[\boldsymbol{C}\boldsymbol{Y}] \quad s.t. \quad \mathrm{Tr}[\boldsymbol{A}_i \boldsymbol{Y}] \geq b_i, \quad \forall i = 1, \cdots, m.$$

Note that when the matrices $\boldsymbol{C}$ and $\boldsymbol{A}_i$'s are all diagonal matrices, the packing and covering SDP problems above are precisely the well-studied packing and covering LP problems, respectively. Near-linear time algorithms for $(1+\epsilon)$-approximately solving packing and covering LPs are very well-studied in a long line of work, some of which are [AO15; AO19; WRM16; Qua20], and these problems have many applications, such as in graph embedding [PST95], approximation algorithms [LN93; Tre98], scheduling [PST95], to name a few.

Dynamic LPs. Near-optimal dynamic algorithms for packing and covering LPs were shown in [BKS23]. The paper studies two kinds of updates – *restricting* and *relaxing* updates. Restricting updates can only shrink the feasible region. In contrast, relaxing updates can only grow the feasible region. In [BKS23], the authors gave a deterministic algorithm that can maintain a $(1+\epsilon)$-approximate solution to either packing and covering LPs that undergo only restricting updates or only relaxing updates in total time $\widetilde{O}(N/\epsilon^3 + t/\epsilon)$, where N is the total number of nonzeros in the initial input and the updates, and t is the number of updates. Hence, this is optimal up to logarithmic factors.

A natural question is whether one can generalize the near-optimal dynamic LP algorithms with polylogarithmic overhead by [BKS23] to work with SDPs since SDPs capture many further applications such as maximum cuts [IPS11; KL96], Sparse PCA [IPS11], sparsest cuts [IPS10], and balanced separators [OSV12], among many others.

Static SDPs. Unfortunately, the algorithms for solving packing and covering SDPs are much more limited, even in the static setting. Near-linear time algorithms are known only for *covering* SDPs when the cost matrix $\boldsymbol{C} = \boldsymbol{I}$ is the identity [PTZ12; ALO16].

The fundamental barrier to working with general psd matrix $\boldsymbol{C}$ in covering SDPs is that it is as hard as approximating the minimum eigenvalue of $\boldsymbol{C}$ (consider the program $\max_{\boldsymbol{Y} \succeq 0} \mathrm{Tr}[\boldsymbol{C}\boldsymbol{Y}]$ such that $\mathrm{Tr}[\boldsymbol{Y}] \leq 1$). To the best of our knowledge, near-linear time algorithms for approximating the minimum eigenvalue assume a near-linear-time solver for $\boldsymbol{C}$, i.e., to compute $\boldsymbol{C}^{-1}\boldsymbol{x}$ in the near-linear time given $\boldsymbol{x}$. This can be done, for example, by applying the power method to $\boldsymbol{C}^{-1}$.

[5]Some papers [Jam+21] flip our definition of packing and covering SDPs by considering their dual form.

When $\boldsymbol{C}^{-1}$ admits a fast solver, sometimes one can approximately solve a covering SDP fast, such as for spectral sparsification of graphs [LS17], and the max-cut problem [AK07; Tre09].

For packing SDPs, there is simply no near-linear time algorithm known. An algorithm for approximately solving packing SDPs and even the generalization to mixed packing-covering SDPs was claimed in [Jam+21], but there is an issue in the convergence analysis even for pure packing SDPs. Fast algorithms for this problem, hence, remain open.

Dynamic SDPs. Since near-linear time static algorithms are prerequisites for dynamic algorithms with polylogarithmic overhead, we can only hope for a dynamic algorithm for covering SDPs when $\boldsymbol{C}$ is an identity. Below, we will show that our algorithm for Theorem 1.2 implies a dynamic algorithm for maintaining the covering SDP solution when there is a single constraint and the updates are restricting. This follows because this problem is equivalence to Problem 1.1.

We first define the dynamic covering problem with a single constraint under restricting updates.

Problem A.2 (Covering SDP with a Single Matrix Constraint under Restricting Updates)**.** *Given $\boldsymbol{A}_0 \succeq 0$, an accuracy parameter $\epsilon > 0$, and an online sequence of vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_T$ that update $\boldsymbol{A}_t \leftarrow \boldsymbol{A}_t - \boldsymbol{v}_t \boldsymbol{v}_t^\top$ such that $\boldsymbol{A}_t \succeq 0$. The problem asks to explicitly maintain, for all t, an $(1+\epsilon)$-approximate optimal value ν_t of the SDP, i.e.,*

$$\nu_t \leq (1+\epsilon) OPT_t \overset{\text{def}}{=} \min_{\boldsymbol{Y} \succeq 0, \operatorname{Tr}[\boldsymbol{A}_t \boldsymbol{Y}] \geq 1} \operatorname{Tr}[\boldsymbol{Y}].$$

Furthermore, given a query request, return a matrix $\boldsymbol{Q}^{(t)}$ where $\boldsymbol{Y}^{(t)} = \boldsymbol{Q}^{(t)} \boldsymbol{Q}^{(t)\top} \in \mathbb{R}^{n \times n}$ is a $(1+\epsilon)$-approximate optimal solution, i.e.,

$$\operatorname{Tr}[\boldsymbol{A}_t \boldsymbol{Y}^{(t)}] \geq 1 \text{ and } \operatorname{Tr}[\boldsymbol{Y}^{(t)}] \leq (1+\epsilon) OPT_t.$$

Problem A.2 is equivalent to Problem 1.1 in the following sense: given an algorithm for Problem 1.1, we can obtain an algorithm for Problem A.2 with the same total update time and optimal query time. Conversely, given an algorithm for Problem A.2, we can obtain an algorithm for Problem 1.1 in the *eigenvalue-only* version with the same total update time.

Proposition A.3. *The following holds,*

1. *Given an algorithm for Problem 1.1 with update time $\mathcal{T}$, there is an algorithm for Problem A.2 with update time $\mathcal{T}$ and query time $O(n)$, i.e., the time required to query $\boldsymbol{Q}^{(t)}$ at any time t.*
2. *Given an algorithm for Problem A.2 with update time $\widetilde{\mathcal{T}}$, there is an algorithm for Problem 1.1 that only maintains the approximate eigenvalues with update time at most $\widetilde{\mathcal{T}}$.*

Proof. TOPROVE 18 □

By plugging Theorem 1.2 into Proposition A.3, we conclude the following.

Corollary A.4. *There is a randomized algorithm for Problem A.2 under a sequence of T restricting updates, that given $n, \boldsymbol{A}_0$ and $\epsilon > 1/n$ as input, with probability at least $1 - 1/n$ works against an oblivious adversary in total update time*

$$O\left(\frac{\log^3 n \log^6 \frac{n}{\epsilon} \log \frac{\lambda_{\max}(\boldsymbol{A}_0)}{\lambda_{\max}(\boldsymbol{A}_T)}}{\epsilon^4} \left(nnz(\boldsymbol{A}_0) + \sum_{i=1}^{T} nnz(\boldsymbol{v}_i) \right) \right),$$

and query time $O(n)$.

B Omitted Proofs

Lemma 3.2. *Given an algorithm $\mathcal{A}$ that solves the decision problem* $\textsc{DecMaxEV}(\epsilon, \boldsymbol{A}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_T)$ *(Definition 3.1) for any $\epsilon > 0$, $\boldsymbol{A}_0 \succeq 0$ and vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_T$ in time $\mathcal{T}$, we can solve Problem 1.1 in total time $O\left(\frac{\log^2 n \log \frac{n}{\epsilon}}{\epsilon} \cdot nnz(\boldsymbol{A}_0) + \frac{\log n}{\epsilon} \log \frac{\lambda_{\max}(\boldsymbol{A}_0)}{\lambda_{\max}(\boldsymbol{A}_T)} \mathcal{T}\right)$.*

Proof. TOPROVE 19 □

Power Method

The following lemma proves that at least one initial random vector has a component along the eigenspace of the maximum eigenvalue.

Lemma B.1. *Let $\boldsymbol{u}_1, \cdots \boldsymbol{u}_l$ denote the maximum eigenvectors of $\boldsymbol{A}$ and let $\boldsymbol{v}^{(0)} \in \mathbb{R}^n$ denote the vector with entries sampled indenpendently from $N(0,1)$, i.e., $\boldsymbol{v}^{(0)} = \sum_{i=1}^{n} \alpha_i \boldsymbol{u}_i$, where $\alpha_i \sim N(0,1)$, $\boldsymbol{u}_i$'s are eigenvectors of $\boldsymbol{A}$. Then, with probability at least $3/4$, $\sum_{i=1}^{l} \alpha_i^2 \geq \frac{1}{25}$.*

Proof. TOPROVE 20 □

Lemma 3.3. *Let $\epsilon > 0$ and $\boldsymbol{A} \succeq 0$. Let $\boldsymbol{W}$ be as defined in Line 9 in the execution of* $\textsc{PowerMethod}(\epsilon, \boldsymbol{A})$*. With probability at least $1 - 1/n^{10}$, for some $\boldsymbol{w} \in \boldsymbol{W}$, it holds that $\boldsymbol{w}^\top \boldsymbol{A} \boldsymbol{w} \geq (1 - \frac{\epsilon}{2})\lambda_{\max}(\boldsymbol{A})$. The total time taken by the algorithm is at most $O\left(\frac{nnz(\boldsymbol{A}) \log n \log \frac{n}{\epsilon}}{\epsilon}\right)$.*

Furthermore, let λ_i and $\boldsymbol{u}_i$ denote the eigenvalues and eigenvectors of $\boldsymbol{A}$. For all i such that $\lambda_i(\boldsymbol{A}) \leq \frac{\lambda_{\max}(\boldsymbol{A})}{2}$, with probability at least $1 - 2/n^{10}$, $\left[\boldsymbol{w}^\top \boldsymbol{u}_i\right]^2 \leq \frac{1}{n^8} \cdot \frac{\lambda_i}{\lambda_1}$.

Proof. TOPROVE 21 □