Adversarially Robust Streaming via Dense–Sparse Trade-offs

Omri Ben-Eliezer*	Talya Eden	Krzysztof Onak
MIT	MIT	Boston University
omrib@mit.edu	talyaa01@gmail.com	krzysztof@onak.pl

Abstract

A streaming algorithm is adversarially robust if it is guaranteed to perform correctly even in the presence of an adaptive adversary. The development and analysis of such algorithms have been a very active topic recently, and several sophisticated frameworks for robustification of classical streaming algorithms have been developed. One of the main open questions in this area is whether efficient adversarially robust algorithms exist for moment estimation problems (e.g., F_2-estimation) under the turnstile streaming model, where both insertions and deletions are allowed. So far, the best known space complexity for streams of length m, achieved using differential privacy (DP) based techniques, is of order $\tilde{O}(m^{1/2})$ for computing a constant-factor approximation with high constant probability (the $\tilde{O}$ notation hides here terms polynomial in $\log m$ and $\log n$, where n is the universe size). In this work, we propose a new simple approach to tracking moments by alternating between two different regimes: a sparse regime, in which we can explicitly maintain the current frequency vector and use standard sparse recovery techniques, and a dense regime, in which we make use of existing DP-based robustification frameworks. The results obtained using our technique break the previous $m^{1/2}$ barrier for any fixed p. More specifically, our space complexity for F_2-estimation is $\tilde{O}(m^{2/5})$ and for F_0-estimation, i.e., counting the number of distinct elements, it is $\tilde{O}(m^{1/3})$.

All existing robustness frameworks have their space complexity depend multiplicatively on a parameter λ called the *flip number* of the streaming problem, where $\lambda = m$ in turnstile moment estimation. The best known dependence in these frameworks (for constant factor approximation) is of order $\tilde{O}(\lambda^{1/2})$, and it is known to be tight for certain problems. Again, our approach breaks this barrier, achieving a dependence of order $\tilde{O}(\lambda^{1/2-c(p)})$ for F_p-estimation, where $c(p) > 0$ depends only on p.

*Work partially conducted while the author was at Harvard University.

1 Introduction

Streaming algorithms are an integral part of the modern toolbox for large-scale data analysis. A streaming algorithm observes a stream of data updates that arrive one by one, and is required to compute some global function of the data using a small amount of space (memory) and with an efficient running time.

Most of the literature on streaming algorithms implicitly assumes that the stream updates do not depend on previous outputs of the algorithm or on the randomness produced by the algorithm. This assumption may not be realistic in many situations: for example, when the data is chosen by a malicious adversary in response to previous outputs, or when data characteristics change based on previous outcomes in some complicated or unpredictable way. As a result, the last couple of years have seen substantial progress in the systematic investigation of *adversarially robust* streaming algorithms [BEY20, BEJWY20, HKM+20, WZ20, ABED+21, KMNS21, BHM+21, ACSS21], which preserve their correctness guarantees even for adaptively chosen data and are thus especially suitable for these interactive settings.

There is already a wide range of problems and settings for which the best known adversarially robust streaming algorithms are almost as efficient as their classical, non-robust counterparts. The *flip number* [BEJWY20] of a streaming problem, an algorithmic stability parameter which counts how many times the output value may change by a multiplicative factor of $1+\alpha$ as the stream progresses, plays a central role in many of these results [HKM+20, WZ20, KMNS21, ACSS21]. When the flip number λ is small, the generic methods developed in these works can turn a classical streaming algorithm into an adversarially robust one with only a small overhead (linear in λ or better) in the space complexity. This is especially useful in the *insertion only* streaming model, where elements are only added to the stream, but may not be deleted from it. Many important streaming problems, such as F_p-estimation, distinct elements, entropy estimation, and various others, all have flip number of $\lambda = O(\alpha^{-1} \log m)$ for insertion-only streams of length m. Under the standard assumption that $m = \mathrm{poly}(n)$, where n is the size of the universe of all possible data elements, and building on additional known results from the streaming literature, one can then obtain adversarially robust insertion-only $(1 \pm \alpha)$-approximation algorithms with space complexity $\mathrm{poly}(1/\alpha, \log n)$.

The situation in the *turnstile streaming* model, which allows both insertions and deletions, is more complicated. The most popular technique for turnstile streams in the classical regime, linear sketching, is provably not adversarially robust [HW13]. Furthermore, the flip number can be very large, potentially even $\Theta(m)$. The best known robustification methods in this regime [HKM+20, ACSS21], based on differential privacy, have a multiplicative $O(\sqrt{\lambda})$ dependence in the flip number (for constant ϵ). Therefore, they induce a space overhead of $\tilde{\Omega}(\sqrt{m})$ compared to the best non-robust algorithms.

A separation result of Kaplan, Mansour, Nissim and Stemmer [KMNS21] shows that indeed the $\sqrt{\lambda}$-type dependence in the flip number is tight for some streaming problems; specifically, they show this for a variant of the Adaptive Data Analysis problem in the context of bounded-space computation. We note, however, that the lower bound of [KMNS21] does not apply to many core problems in the streaming literature, for which no separation between the classical oblivious and adversarially robust settings is known. In particular, this is the case for F_p-estimation, in which the goal is to approximate $\sum_i |v_i|^p$, the p-th moment of a frequency vector $v \in \mathbb{Z}^n$. This gives rise to the following question, widely regarded as one of the central open questions on adversarially

robust streaming.[1]

What is the adversarially robust space complexity of F_p-estimation in the turnstile streaming model?

In this work we show that a combination of existing building blocks from the literature (with slight modifications and simplifications) can yield a substantially improved space complexity for the above problem. Our results hold when deletions are allowed, as long as each update increases or decreases the frequency of a single element by 1 (or more generally, by a bounded integer amount). We also allow frequencies of elements to become negative in the process, which is known as the *general turnstile streaming model*.

2 Overview of Our Contribution

2.1 Our results

We give an F_p-estimation algorithm that breaks the $\sqrt{m}$ (or $\sqrt{\lambda}$) space barrier. We now state a *simplified* version of the main result, focusing just on on the dependence on the stream length m and domain size n, whenever it is polynomial. For the full statement of our results, see Theorem 20.

Theorem 1 (Simplified main result)**.** *For any fixed $p \in [0, \infty)$ and $\alpha > 0$, there is an adversarially robust F_p-estimation streaming algorithm that computes a $(1 \pm \alpha)$-approximation, using:*

- $\tilde{O}(m^{1/3})$ *space if* $p \in [0, 1]$,
- $\tilde{O}(m^{p/(2p+1)})$ *if* $p \in [1, 2]$,
- $\tilde{O}(m^{p/(2p+1)} \cdot n^{1-5/(2p+1)})$ *if* $p \in (2, \infty)$,

where the $\tilde{O}$ notation suppresses factors that are polynomial in α^{-1}, $\log m$, and $\log n$. The algorithm gives correct estimates throughout the entire stream with probability $1 - o(1)$.

We note that since the flip number for the moment estimation problem is $\lambda = \Theta(m)$ (see Section 3.3), the dependency of the space complexity of our approach in λ is $\widetilde{O}(\lambda^{1/3})$ for $p \in [0, 1]$ and $\widetilde{O}(\lambda^{\frac{1}{2} - \frac{1}{4p+2}})$ for $p > 1$. This improves polynomially upon the currently best known $\widetilde{O}(\sqrt{\lambda})$ bound, obtained using the aforementioned differential privacy based robustness frameworks [HKM+20, ACSS21]. Together with the separation of [KMNS21], our result (see also [Jay21]) suggests that a paradigm shift may be required in order to achieve improved space complexity for turnstile streams: rather than developing very widely applicable robustness frameworks that suffer from the $\sqrt{\lambda}$-type lower bound due to their wide applicability, it may make sense to look for methods that are perhaps somewhat less generic, and exploit other properties of the problem, beyond just the flip number.

[1] To the best of our knowledge, the first explicit appearance of this question in the literature is in Jayaram's Ph.D. thesis [Jay21, page 26]. See also a talk by Stemmer [Ste21] at 54:45 and the third question on the list of open questions from the STOC 2021 Workshop on Robust Streaming, Sketching, and Sampling [Rob21a].

2.2 Our techniques

Our result relies on a straightforward combination of known techniques from the literature. The bottleneck of the previous best result for moment estimation for general turnstile streaming is the direct reliance on the flip number λ, which for general streams can be of order $\Omega(m)$. As mentioned above, methods that take only the flip number into account (and do not use any other characteristics of the problem at hand) cannot get space complexity much smaller than $\sqrt{\lambda}$ (that is, $\sqrt{m}$ for norm estimation). Thus, we exploit a specific characteristic of F_p-estimation: the actual number of significant changes to the p-th moment can only be large if the moment remains small. This can only be the case if the underlying vector is sparse, i.e., has relatively few non-zero coordinates. We therefore divide the current state of the frequency vector into two regimes: sparse and dense, using a threshold T. If the vector has at most T non-zero coordinates, it is considered sparse. If it has more than $4T$ non-zero coordinates, it is considered dense. For densities in between, the state of the vector may be temporarily classified as either dense or sparse.

In the sparse regime, we take the simplest possible approach, which is storing the input explicitly, using a sparse representation, which requires only $O(T)$ space. In this form, it is easy to maintain the current moment, since we know the current frequency vector exactly. In the dense regime, we apply the technique from the paper of Hassidim, Kaplan, Mansour, Matias, and Stemmer [HKM+20], which uses differential privacy to protect not the input data, but rather the internal randomness of estimators it uses. At a high level, their framework consists of invoking a set of k estimators, which upon each query provide an updated estimate. Given the stream of updates, they use differentially private methods to detect whenever the current estimate is no longer relevant, at which point they query the set of estimators to get an updated estimate. Their technique in general increases the space requirement by a factor of the square root of the flip number, compared to that of oblivious streaming algorithms. In particular, applying their method for the moment estimation problems, requires invoking $k = \widetilde{O}(\sqrt{\lambda})$ instances of oblivious F_p-estimation algorithms. We improve on the above, by taking advantage of the fact that the estimated value of the p-th moment cannot change too rapidly for dense vectors. For instance, for $p = 0$ (the distinct element count) or $p = 1$, if the vector has at least T non-zero coordinates, at least $\Omega(T)$ insertions or deletions are required to change it by a constant factor. Similarly, for $p = 2$, at least $\Omega(\sqrt{T})$ insertions or deletions are needed. Hence, the flip number for the dense regime is much lower, and we can make significantly fewer queries to a set of oblivious F_p-estimation algorithms. This in turn implies that we can significantly reduce the number of required estimators.

The missing component in our description so far is how the transition between the regimes happens. If we are transitioning from the sparse regime to the dense one, we have all the information needed about the current state of the input vector that we are tracking. If we are transitioning from the dense regime to the sparse regime, we use off-the-shelf sparse recovery techniques (also known as compressed sensing) to recover the frequency vector exactly. To know when to do this, we run in parallel an adversarially robust streaming algorithm for distinct element counting, which we also know has to be queried only every $\Omega(T)$ steps.

2.3 Pseudocode

We present the pseudocode of our approach as Algorithm 1. We maintain adversarially robust estimators, $\mathcal{A}_{\text{approx}}$ and $\mathcal{A}_{\text{density}}$, that are queried significantly less frequently than m times throughout the entire execution of the algorithm. We query them at regular intervals, knowing that their values

Algorithm 1: Adversarially Robust Streaming of Moments

Parameters: $p \in [0, \infty)$ describing the moment, dimension $n \in \mathbb{Z}_+$, stream length bound $m \in \mathbb{Z}_+$, threshold $T \in \mathbb{R}_+$, approximation quality parameter $\alpha \in (0, 1)$, error parameter δ

$\mathit{regime} \leftarrow$ SPARSE; $v \leftarrow (0, \ldots, 0)$; $\mathit{count} \leftarrow 0$

$M_{\text{exact}} \leftarrow 0$; $M_{\text{approx}} \leftarrow 0$; $k_{\text{approx}} \leftarrow 0$

$\mathcal{A}_{\text{sparse}} \leftarrow$ the sparse recovery algorithm (Theorem 9) with sparsity parameter $k = \lceil 4T \rceil$

$\mathit{interval} \leftarrow \begin{cases} \alpha T/4 & \text{for } p \in [0, 1] \\ \frac{\alpha}{32p} \left(\frac{\alpha T}{16}\right)^{1/p} & \text{for } p \in (1, \infty) \end{cases}$

$\mathit{interval} \leftarrow \max\{\lfloor \mathit{interval} \rfloor, 1\}$

$\mathcal{A}_{\text{density}} \leftarrow (m/\lfloor T/10 \rfloor)$-query adversarially robust streaming algorithm (Algorithm 2) for $(1 \pm .25)$-approximation of number of distinct elements with error parameter $\delta/2$

$\mathcal{A}_{\text{approx}} \leftarrow (m/\mathit{interval})$-query adversarially robust streaming algorithm (Algorithm 2) for $(1 \pm \alpha/4)$-approximation of the p-th moment with error parameter $\delta/2$

foreach update (i, Δ) **do**

- $\mathit{count} \leftarrow \mathit{count} + 1$
- Process the update (i, Δ) by $\mathcal{A}_{\text{sparse}}$, $\mathcal{A}_{\text{density}}$, $\mathcal{A}_{\text{approx}}$
- **if** *count* is a multiple of $\lfloor T/10 \rfloor$ **then** $k_{\text{approx}} \leftarrow$ estimate from $\mathcal{A}_{\text{density}}$
- **if** *count* is a multiple of *interval* **then** $M_{\text{approx}} \leftarrow$ estimate from $\mathcal{A}_{\text{approx}}$
- **if** *regime* = SPARSE **then**
  - Update v and M_{exact}
  - Output M_{exact}
  - **if** $\|v\|_0^0 \geq 4T$ **then** *regime* $\leftarrow$ DENSE
- **else**
  - Output M_{approx}
  - **if** $k_{\text{approx}} \leq 2T$ **then**
    - Use $\mathcal{A}_{\text{sparse}}$ to recover v
    - $M_{\text{exact}} \leftarrow \|v\|_p^p$
    - *regime* $\leftarrow$ SPARSE

cannot change too rapidly, when the vector is dense. We note that we do not use them when the vector is sparse, as in that regime their readouts may be inaccurate.

We present the pseudocode for an adversarially robust algorithm that has to answer only a limited number of queries in Algorithm 2. This algorithm is a simplified and adjusted version of the algorithm that appeared in the work of Hassidim et al. [HKM+20]. In the introduction of their paper, they note that constructing a bounded-query variant of their algorithm is possible, but do not give any details beyond that. As this variant is crucial for our purposes, we present this construction in full detail for completeness.

Algorithm 2: Bounded Query Adversarially Robust Streaming
Parameters: number q of queries, probability δ of error, bound τ on the size of the range of possible values, desired approximation parameter α
Subroutine: oblivious $(1 \pm \alpha/3)$-approximation streaming algorithm $\mathcal{A}$ as described in the statement of Lemma 11

$\epsilon \leftarrow 1/100$
$\delta' \leftarrow \epsilon\delta/(10q)$
$\epsilon' \leftarrow \epsilon/\sqrt{8q\ln(1/\delta')}$
$k \leftarrow \Theta(\epsilon'^{-1}\log\frac{2q\log 2\tau}{\alpha\delta})$
Initialize k independent instances $\mathcal{A}_1, \ldots, \mathcal{A}_k$ of $\mathcal{A}$
foreach update (i, Δ) **do**
    Process the update (i, Δ) by all of $\mathcal{A}_1, \ldots, \mathcal{A}_k$
    if the adversary is querying for the current output **then**
        foreach $j \in [k]$ **do**
            $\gamma_j \leftarrow$ estimate from $\mathcal{A}_j$
            $\gamma'_j \leftarrow \gamma_j$ rounded up to a multiple of $1 + \frac{\alpha}{3}$ and truncated if not in $\{0\} \cup [1, \tau)$.
        Output an $(\epsilon', 0)$-DP estimate of the median of $\{\gamma'_1, \ldots, \gamma'_k\}$ as in Theorem 13 with parameters set to $\delta \leftarrow \frac{\delta}{2q}$ and $\epsilon \leftarrow \epsilon'$

3 Preliminaries

3.1 Basic notation and terms

For any $k \in \mathbb{N}$, we write $[k]$ to denote $\{1, \ldots, k\}$, the set of k smallest positive integers.

Definition 2 (The p^{th}-moment)**.** The p^{th}-moment of a vector $v \in \mathbb{R}^n$ is $\|v\|_p^p = \sum_{i=1}^n |v_i|^p$, for any $p \in [0, \infty)$. We interpret the 0^{th}-moment as $\|v\|_0^0 = |\{i \in [k] : v_i \neq 0\}|$, i.e., the number of non-zero coordinates of v, by assuming in this context that $0^0 = 0$ and $x^0 = 1$ for any $x \neq 0$.

Definition 3 (Vector density)**.** Let $k \in \mathbb{R}$ and $n \in \mathbb{N}$. We say that a vector $v \in \mathbb{R}^n$ is *k-dense* if at least k of its coordinates are non-zero, and *k-sparse* if at most k of them are non-zero.

Definition 4 ($(1 \pm \alpha)$-approximation)**.** For any $Q, Q' \in [0, \infty)$ and $\alpha \in [0, 1]$, we say that Q' is a *$(1 \pm \alpha)$-approximation* to Q if

$$(1 - \alpha)Q \leq Q' \leq (1 + \alpha)Q.$$

3.2 Streaming algorithms

A *streaming algorithm* receives, one by one, a stream of updates $u_1, u_2, \ldots, u_m$ that modify the data, and is typically required to compute or approximate some function f of the data over the stream of updates. In this paper, we fully focus on the setting in which the input is a *frequency vector* $v \in \mathbb{Z}^n$ for some integer n, known to the algorithm in advance. Initially, at the beginning of the stream, this vector is the all-zero vector, i.e., $v = (0, \ldots, 0)$. The stream consists of updates of the form $u_j = (i_j, \Delta_j)$ in which $i_j \in [n]$ and $\Delta_j \in \{-1, 1\}$. The interpretation of each update is

that Δ_j is added to the i_j-th coordinate of v, i.e., each update increases ("insertion") or decreases ("deletion") a select coordinate by 1.[2]

For any fixed stream of updates, the streaming algorithm is required to output $f(v)$ or a good approximation to $f(v)$ (e.g., a $(1 \pm \alpha)$-approximation if $f(v)$ is a non-negative real number) after seeing the stream with probability $1 - \delta$, for some parameter δ. We refer to $1 - \delta$ in this context, as *success probability*. We sometimes refer to streaming algorithms in this model, in which the stream is independent of the actions of the streaming algorithm, as *oblivious* or *non-robust* to distinguish them from adversarially robust streaming algorithms, which we design in this paper.

Intermediate approximations. We assume that the streaming algorithm does not know the exact length of the stream in advance, and only knows an upper bound on it. Because of that, we assume that the algorithm can be asked to output its approximation of $f(v)$ at any time throughout the stream. This is true for a large majority of streaming algorithms, and in particular, to the best of our knowledge, applies to all general moment streaming algorithms. For this type of streaming algorithm, if its success probability is $1 - \delta$, this means that it can output a desired approximation with probability at least $1 - \delta$ at any fixed prefix of the stream.

Streaming related notation and assumptions. Throughout the paper, we consistently write n to denote the dimension of the vector on which the streaming algorithm operates. We use m to denote an upper bound on the length of the input and we assume that $m = O(\text{poly}(n))$. We assume that machine words are large enough to represent n and m, i.e., the number of bits in them is at least $\Omega(\log \max\{m, n\})$ and we express the complexity of algorithms in words.

3.3 Adversarially robust streaming algorithms

In this paper, we design streaming algorithms in the adversarially robust streaming model of Ben-Eliezer et al. [BEJWY20], which we now introduce in the context of computing a $(1 \pm \alpha)$-approximation to values of a function f.

Definition 5 (Adversarially robust streaming)**.** Fix a function $f : \mathbb{Z}^n \to [0, \infty)$ and let $\alpha > 0$. The robust streaming model is defined as a game between two players, Adversary and Algorithm, where f, α, and the stream length m are known to both players. In each round $j \in [m]$:

- First, Adversary picks (i_j, Δ_j) for $i_j \in [n]$ and $\Delta_j \in \{-1, 1\}$ and sends them to Algorithm. The choice of i_j and Δ_j may depend on all previous updates sent by Adversary as well as all previous outputs of Algorithm.
- Algorithm outputs y_j, which is required to be a $(1 \pm \alpha)$-approximation to $f(v^{(j)})$, where $v^{(j)}$ is the vector aggregating all updates so far (that is, $v_i^{(j)} = \sum_{j' \le j : i_{j'} = i} \Delta_{j'}$ for all $i \in [n]$). Algorithm sends y_j to Adversary.

[2] As mentioned, the update values Δ_j are always ± 1 in the model we consider here. In the most general setting for turnstile streaming, Δ_j may be unbounded; however, for our arguments to hold, it is important that Δ_j are bounded in some way (e.g., satisfy $\Delta_j \in [-C, -C^{-1}] \cup \{0\} \cup [C^{-1}, C]$ for some constant $C \ge 1$). This assumption is not necessary for F_0-estimation—i.e., counting the number of distinct elements—for which updates of arbitrary magnitude are allowed as long as they can be handled by a non-robust streaming algorithm on which we build.

Algorithm's goal is to return correct outputs at all times. That is, y_j is required to be a $(1 \pm \alpha)$-approximation to $f(v^{(j)})$ for all $j \in [m]$. Conversely, Adversary's goal is to have Algorithm's output y_j that is not a $(1 \pm \alpha)$-approximation to $f(v^{(j)})$ for some $j \in [m]$.

We say that a streaming algorithm is *adversarially robust* and has success probability $1 - \delta$ for some $\delta \in [0, 1]$ if it can provide all correct outputs as Algorithm with probability at least $1 - \delta$ for any Adversary.

We now introduce the notion of an *q-query* adversarially robust streaming algorithm, which has to provide no more than q outputs.

Definition 6 (q-query robust streaming algorithm)**.** Let $q \in \mathbb{N}$. A *q-query adversarially robust streaming algorithm* is defined similarly to a standard adversarially robust streaming algorithm, with the following modification: Adversary may perform at most q queries for outputs y_j from Algorithm, and only receives the output y_j in these time steps where queries are made. Adversary may pick these time steps adaptively as a function of all previous interactions. We say that a q-query adversarially robust streaming algorithm has a probability of success $1-\delta$, for some $\delta \in [0, 1]$, if for any Adversary that makes at most q queries, with probability at least $1-\delta$, it correctly answers all of them.

Note on the tracking property. Oblivious streaming algorithms are not required to have the "tracking" property, i.e., they have to provide a good approximation at the end of the stream (or at any fixed point), but their definition does not require any type of consistent behavior throughout the stream. This is required, however, for adversarially robust streaming algorithms. We build our adversarially robust streaming algorithms from oblivious streaming algorithms that do not have a tracking property.

3.4 Frequency moment estimation

The main focus of this paper is designing streaming algorithms for the problem of F_p-estimation, i.e., estimation of the p^{th}-moment, in the turnstile streaming model. To build our adversarially robust algorithms, we use classical, non-robust, turnstile streaming algorithms for F_p-estimation.

Theorem 7 (Previously known F_p-estimation results)**.** *There exist turnstile streaming algorithms that with probability of success $\frac{9}{10}$, return a $(1 \pm \alpha)$-approximation to the p^{th} moment of a frequency vector in $\mathbb{Z}^n$ on the stream of length m with the following space complexity:*

Value of p	**Space**	**Reference**
$p = 0$	$O(\alpha^{-2} \cdot \log n \cdot (\log(1/\alpha) + \text{loglog}(m)))$	[KNW10b]
$p \in (0, 2]$	$O(\alpha^{-2} \cdot \log m)$	[KNW10a]
$p > 2$	$O\left(n^{1-2/p} \cdot \left(\alpha^{-2} + \alpha^{-4/p} \log n\right)\right)$	[GW18]

3.5 Flip number

The flip number, defined in [BEJWY20], plays a prominent role in many of the previous results on adversarially robust streaming. For completeness, we next provide its definition suited for our context.

Definition 8 (Flip number)**.** Fix a function $f\colon \mathbb{Z}^n \to [0,\infty)$ and $\alpha > 0$. Let $u_1 = (i_1, \Delta_1)$, $\ldots$, $u_m = (i_m, \Delta_m)$ be a sequence of updates to some vector v whose initial value is $v^{(0)}$, and let $v^{(j)}$ be the value of the vector after j updates have been received. The *flip number* $\lambda_\alpha(f, (u_1, \ldots, u_m))$ of f with respect to the above sequence is the size t of the largest subsequence $0 \le j_1 < \ldots < j_t \le m$ for which $f\left(v^{(j_l)}\right)$ is not a $(1 \pm \alpha)$-approximation of $f\left(v^{(j_{l+1})}\right)$ for any $l = 1, \ldots, t-1$. The flip number $\lambda_\alpha(f)$ of f is the maximum of $\lambda_\alpha(f, (u_1, \ldots, u_m))$ over all possible choices of the sequence $u_1, \ldots, u_m$.

It is easy to see that the flip number of F_p-estimation is $\Omega(m)$ for any p: indeed, consider the following pair of insertion-deletion updates $(i, 1), (i, -1)$, repeated $m/2$ times. In such a stream, the value of the p^{th} moment alternates m times between 0 and 1.

3.6 Sparse recovery

In our algorithm, we use sparse recovery to reconstruct the current frequency vector when it becomes sparse, which is possible even if it was arbitrarily dense in the meantime. For an introduction to the topic of sparse recovery, see the survey of Gilbert and Indyk [GI10]. Here we use the following streaming subroutine introduced by Gilbert, Strauss, Tropp, and Vershynin [GSTV07].

Theorem 9 (Sparse recovery [GSTV07])**.** *There is a streaming algorithm that takes a parameter k, operates on a vector $v \in \mathbb{Z}^n$, and has the following properties. It uses $O(k \operatorname{polylog} n)$ words of space and handles each coordinate update in $O(\operatorname{polylog} n)$ time. Whenever the input vector is k-sparse, the algorithm can reconstruct it exactly in $O(k \operatorname{polylog} n)$ time.*

With probability $1 - O(n^{-3})$, taken over the initial selection of randomness, the algorithm can correctly recover all *k-sparse vectors in all parts of the process (even when they are constructed in an adaptive manner).*

3.7 Differential privacy

Differential privacy [DMNS06] is by now a standard formal notion of privacy for individual data items in large datasets. The formal definition is as follows.

Definition 10 (Differential Privacy)**.** Let A be a randomized algorithm operating on databases. A is (ϵ, δ)-differentially private (in short (ϵ, δ)-DP) if for any two databases S and S' that differ on one row, and any event T, it holds that

$$\Pr[A(S) \in T] \le e^\epsilon \cdot \Pr[A(S') \in T] + \delta$$

In the framework of Hassidim et al. [HKM+20], which we use here, DP is used in a somewhat non-standard way to protect the internal randomness of instances of a static algorithm.

4 Bounded Query Adversarially Robust Streaming

In this section, we present a q-query adversarially robust streaming algorithm for approximating a function $f\colon \mathbb{Z}^n \to \mathbb{R}$. In the case that $q \ll m$, and for problems where the flip number is $\lambda = \Theta(m)$, such as turnstile F_p-estimation, it obtains significant gains in the space complexity compared to the general algorithm introduced by Hassidim et al. [HKM+20]. The space overhead of the q-query

robust algorithm over an oblivious streaming algorithm is roughly $\sqrt{q}$, independently of how much the function changes in the meantime. Informally, this is because the flip number of the output observed by the adversary decreases from a (worst case) $\Theta(m)$ factor to a $\Theta(q)$ one. The algorithm is a simplified and adjusted version of the algorithm of Hassidim et al. [HKM+20]. Their algorithm builds on two important primitives: a DP procedure for detecting when a set of functions exceeds a certain threshold, and a DP procedure for computing the median of a set of values. Their algorithm works by invoking the threshold detection procedure after each update, in order to detect whether the estimate of the computed function should be re-evaluated. If this is the case, then the median procedure is used to compute a private updated estimation. Compared to their algorithm, we do not need the first primitive, i.e., the differentially private threshold detection. We only recompute a private median when the algorithm is replying to a query from the adversary.[3]

Lemma 11 (q-query adversarially robust algorithm)**.** *Let $\alpha, \delta \in (0,1)$ and $q \in \mathbb{Z}_+$. Let $\mathcal{A}$ be an oblivious streaming algorithm for computing a $(1\pm\alpha/3)$-approximation to a function $f : \mathbb{Z}^n \to [0,\infty)$ that uses S space and is correct with probability $9/10$ when queried once. Additionally, let $\{0\}\cup[1,\tau]$ be the range of possible correct values of f on the stream.*

There is a q-query adversarially robust streaming algorithm, Algorithm 2, that uses $\chi \cdot S$ space to provide a $(1\pm\alpha)$-approximation to f with probability $1-\delta$, where

$$\chi \stackrel{\text{def}}{=} O\left(\sqrt{q\log(2q/\delta)}\cdot\log\frac{2q\log 2\tau}{\alpha\delta}\right).$$

4.1 Tools from differential privacy

In order to prove Lemma 11, we use the following set of tools from the differential privacy literature. First, the following theorem allows for composing multiple applications of a DP mechanism.

Theorem 12 ([DRV10])**.** *Let $\epsilon, \delta' \in (0,1]$ and let $\delta \in [0,1]$. An algorithm that allows for q adaptive interactions with an (ϵ,δ)-DP mechanism is $(\epsilon', q\delta+\delta')$-DP for $\epsilon' \stackrel{\text{def}}{=} \sqrt{2q\ln(1/\delta')}\cdot\epsilon + 2q\epsilon^2$.*

At the heart of the algorithm is a DP mechanism for computing a median of a set of items. While sublinear-space algorithms for DP median estimation are known to exist [ABEC21], for our purposes it suffices to use a simple approach with near-linear space complexity.

Theorem 13 ([HKM+20, Theorem 2.6])**.** *For every $\epsilon, \delta \in (0,1)$, there exists an $(\epsilon,0)$-DP algorithm for databases $S \in X^*$ of size $\Omega\left(\frac{1}{\epsilon}\log\left(|X|/\delta\right)\right)$ that outputs an element $x \in X$ such that with probability at least $1-\delta$, there are at least $|S|/2-\Gamma$ elements in S that are bigger or equal to x and at least $|S|/2-\Gamma$ elements in S that are smaller or equal to x, where $\Gamma \stackrel{\text{def}}{=} O\left(\frac{1}{\epsilon}\log\left(|X|/\delta\right)\right)$. The algorithm uses $O(|S|)$ space.*

We note that the original statement of the theorem did not mention the space complexity, but such space can be obtained using standard approaches in the DP literature, e.g., by applying the exponential mechanism with the Gumbel trick [ABEC21, MT07].

Finally, we use a known generalization theorem that shows that a differentially private mechanism cannot heavily bias its computation on a sufficiently large set of random samples.

[3] We note that the private thresholds procedure is crucial for Hassidim et al. [HKM+20] to improve their space overhead from roughly $\sqrt{m}$, which strongly depends on the stream length, to roughly $\sqrt{\lambda}$, which depends only on the flip number. In the problems we consider here, this is not essential as $\lambda = \Theta(m)$ anyway.

Theorem 14 ([BNS+21, DFH+15]). *Let $\epsilon \in (0, 1/3)$, $\delta \in (0, \epsilon/4)$, and $t \geq \epsilon^{-2} \log(2\epsilon/\delta)$. Let $\mathcal{A} : X^t \to 2^X$ be an (ϵ, δ)-DP algorithm that operates on a database of size t and outputs a predicate $h : X \to \{0, 1\}$. Let $\mathcal{D}$ be a distribution on X, let S be a database containing t elements drawn independently from $\mathcal{D}$, and let $h \leftarrow \mathcal{A}(S)$ be an output of $\mathcal{A}$ on S. Then*

$$\Pr_{\substack{S \sim \mathcal{D} \\ h \leftarrow \mathcal{A}(S)}} \left[\left| \frac{1}{|S|} \sum_{x \in S} h(x) - \mathop{\mathbb{E}}_{x \sim \mathcal{D}}[h(x)] \right| \geq 10\epsilon \right] \leq \frac{\delta}{\epsilon}.$$

4.2 Proof of Lemma 11

Recall that Algorithm 2 runs multiple copies $\mathcal{A}_1, \ldots, \mathcal{A}_k$ of an oblivious streaming algorithm $\mathcal{A}$, where each $\mathcal{A}_i$ uses an independent random string $r_i \in \{0, 1\}^*$, selected from the same distribution as the randomness of $\mathcal{A}$. Let $R \stackrel{\text{def}}{=} \{r_1, \ldots, r_k\}$ be the collection of random strings used by the copies of $\mathcal{A}$. We can view R as a database, in which each r_i is a row, and Algorithm 2 as a mechanism that operates on it. We now show that Algorithm 2 does not reveal much about the collection of random strings it uses.

Lemma 15. *Algorithm 2 is (ϵ, δ')-DP with respect to R, the collection of random strings used by copies of $\mathcal{A}$, where ϵ and δ' are as defined in the algorithm.*

Proof. The only way in which the algorithm reveals anything about the strings r_i is by outputting the private median of current estimates of all algorithms. Note that the set of possible values of estimates γ'_j is of size at most $1 + \lceil \log_{1+\alpha/3}(\tau) \rceil = O(\alpha^{-1} \log(2\tau))$. Let ϵ' be as defined in Line 3. It follows from Theorem 13 that each application of the median algorithm is $(\epsilon', 0)$-DP with respect to R and errs with probability at most $\delta/(2q)$ when the constant hidden by the asymptotic notation in the definition of k in Line 4 is large enough. Applying Theorem 12, we conclude that the entire algorithm is (ϵ'', δ)-DP with respect to R, where

$$\epsilon'' \stackrel{\text{def}}{=} \sqrt{2q \ln(1/\delta')} \cdot \epsilon' + 2q\epsilon'^2 \leq \frac{\epsilon}{2} + \frac{\epsilon^2}{4} \leq \epsilon. \qquad \square$$

We now proceed to prove Lemma 11, i.e., that Algorithm 2 has the desired properties.

Proof of Lemma 11. Recall that by Lemma 15, Algorithm 2 is (ϵ, δ')-DP with respect to the collection of random strings that copies of $\mathcal{A}$ use, where ϵ is defined in Line 1 and δ' is defined in Line 2. For any random string $r \in \{0, 1\}^*$ that an instance of $\mathcal{A}$ may use and for any $i \in [q]$, let $h_i(r) : \{0, 1\}^* \to \{0, 1\}$ equal 1 if $\mathcal{A}$ outputs a $(1 \pm \alpha/3)$-approximation to the function being computed on the prefix of the stream, when asked the i-th query and using r as its randomness. Let r_j be the randomness that $\mathcal{A}_j$, the j-th instance of $\mathcal{A}$, uses. Since the adversary can be seen as an (ϵ, δ')-DP mechanism, and this includes all generated queries and updates to the stream, by Theorem 14 and the union bound, we get that

$$\left| \mathop{\mathbb{E}}_{r}[h_i(r)] - \frac{1}{k} \sum_{j=1}^{k} h_i(r_j) \right| \leq 10\epsilon = \frac{1}{10}$$

for all $i \in [q]$, with probability at least $1 - q \cdot \frac{\delta'}{\epsilon} \geq 1 - \delta/10$ as long as $k \geq \epsilon^{-2} \log(2\epsilon/\delta')$. We now show that this condition holds for a sufficiently large constant hidden by the asymptotic notation

in the definition of k in Line 4. First, observe that ϵ is defined to be a positive constant, and hence ϵ^{-2} is a constant as well and can easily be bounded by a sufficiently large constant hidden in the definition of k. It remains to bound $\log(2\epsilon/\delta') = \log(q/(5\delta))$. To this end, observe that the definition of k also has two multiplicative terms. The first one is

$$\frac{1}{\epsilon'} = \frac{\sqrt{8q\ln(1/\delta')}}{\epsilon} \geq 100\sqrt{8\ln 10} \geq 1,$$

and the second one is

$$\log\frac{2q\log 2\tau}{\alpha\delta} \geq \log\frac{2q}{\delta} = \log 2 + \log\frac{q}{\delta}.$$

Since $\frac{q}{\delta} \geq 1$, and, therefore, $\log\frac{q}{\delta} \geq 0$, their product multiplied by a sufficiently large constant—which again can be hidden in the asymptotic notation in the definition of k—is greater than $\log\frac{20q}{\delta} = \log 20 + \log\frac{q}{\delta}$. This finishes the proof that $k \geq \epsilon^{-2}\log(2\epsilon/\delta')$ can easily be achieved by properly adjusting constants.

Since $\mathcal{A}$ is correct with probability at least $9/10$ on any fixed data stream, this implies that for each $i \in [q]$, at least $\frac{4}{5}k$ predicates $h_i(r_j)$ are 1. In other words, with probability at least $1 - \delta/10$, for each query from the adversary, at least $\frac{4}{5}k$ of the collected estimates γ_j of the current value of f are its $(1 \pm \alpha/3)$-approximations. Note that the rounding step can only increase the approximation error by a factor of at most $(1 + \alpha/3)$, which means that at least $\frac{4}{5}k$ of estimates γ'_j are $(1 \pm \alpha)$-approximation of the current value of f because $(1 + \alpha/3)^2 < 1 + \alpha$.

Now note that the private median algorithm returns an estimate that is greater than or equal to at least $2/5$ of estimates γ'_j and also smaller than or equal to at least $2/5$ of the same estimates with probability at least $1 - \frac{\delta}{2q}$. As long as this algorithm outputs such an estimate, and as long as the fraction of bad estimates is at most $1/5$, this means that the algorithm outputs a $(1 \pm \alpha)$-approximation to the value of f at the query point. By the union bound this occurs for all queries with probability at least $1 - \delta/2 - \delta/10 \geq 1 - \delta$.

The space complexity of the algorithm is dominated by the space to store the instances of $\mathcal{A}$. Note that there are $k = O(\sqrt{q\log(2q/\delta)} \cdot \log\frac{2q\log 2\tau}{\alpha\delta})$ of them. □

5 Bounded Change

As discussed in the introduction, our frequency moment estimation algorithm gains from the fact that when the vector v is k-dense for some value of k, the value of $\|v\|_p^p$ cannot change too rapidly. We formally state and prove this below.

5.1 Moments $p \in [0, 1]$

Lemma 16. *Let $v, v' \in \mathbb{Z}^n$, $p \in [0, 1]$, $\alpha \in (0, 1)$, and $k \in \mathbb{Z}_+$. If v is k-dense and $\|v - v'\|_1 \leq \alpha k$, then $\|v'\|_p^p$ is a $(1 \pm \alpha)$-approximation to $\|v\|_p^p$.*

Proof. Consider a function $f : \mathbb{R} \to \mathbb{R}$ defined as $f(t) \stackrel{\text{def}}{=} |t|^p$. We claim that for any $t \in \mathbb{Z}$, $|f(t) - f(t+1)| \leq 1$. This is easy to verify for $t \in \{-1, 0\}$, because $f(-1) = f(1) = 1$ and $f(0) = 0$. Since f is differentiable in $(-\infty, -1] \cup [1, \infty)$ with the absolute value of the derivative bounded by 1, the claim holds in that range as well, i.e., for other $t \in \mathbb{Z} \setminus \{-1, 0\}$. This implies that for any $t, t' \in \mathbb{Z}$, $|f(t) - f(t')| \leq |t - t'|$.

We have

$$\left| \|v\|_p^p - \|v'\|_p^p \right| \leq \sum_{i=1}^{n} \left| f(v_i) - f(v_i') \right| \leq \sum_{i=1}^{n} |v_i - v_i'| = \|v - v'\|_1 \leq \alpha k.$$

Since v is k-dense, for at least k of its coordinates i, $|v_i| \geq 1$, and hence $\|v\|_p^p \geq k$. We therefore have $\|v'\|_p^p = \|v\|_p^p + \left(\|v'\|_p^p - \|v\|_p^p \right) \leq \|v\|_p^p + \alpha k \leq (1+\alpha) \|v\|_p^p$. Analogously, $\|v'\|_p^p \geq \|v\|_p^p - \alpha k \geq (1-\alpha) \|v\|_p^p$. □

5.2 Moments $p \in [1, \infty)$

We use the following two well-known facts, which are easy to verify via basic calculus.

Fact 17. *For $p \geq 1$ and $\alpha \in (0,1)$, $(1-\alpha)^p \geq 1 - \alpha p$.*

Fact 18. *For $\alpha \in (0,1)$, $e^{\alpha} \leq (1 + 2\alpha)$.*

Lemma 19. *Let $v, v' \in \mathbb{Z}^n$, $p \in [1, \infty)$, $\alpha \in (0,1)$, and $k \in \mathbb{Z}_+$. If v is k-dense and $\|v - v'\|_1 \leq \frac{\alpha}{8p}(\frac{\alpha k}{4})^{1/p}$, then $\|v'\|_p^p$ is a $(1 \pm \alpha)$-approximation to $\|v\|_p^p$.*

Proof. Let $\Delta \stackrel{\text{def}}{=} v - v'$. We partition the set of indices, $[n]$, into two sets, $\mathcal{I}_{\text{small}}$ and $\mathcal{I}_{\text{large}}$, based on how $|\Delta_i|$ compares to $|v_i|$. We have $\mathcal{I}_{\text{small}} \stackrel{\text{def}}{=} \left\{ i \in [n] : |\Delta_i| \leq \frac{\alpha}{4p} |v_i| \right\}$ and $\mathcal{I}_{\text{large}} \stackrel{\text{def}}{=} [n] \setminus \mathcal{I}_{\text{small}}$.

For $i \in \mathcal{I}_{\text{small}}$, we have

$$\left((1 - \frac{\alpha}{4p}) |v_i| \right)^p \leq |v_i'|^p \leq \left((1 + \frac{\alpha}{4p}) |v_i| \right)^p.$$

The left–hand side can be bounded from below by $(1 - \alpha/4)|v_i|^p$, using Fact 17. The right–hand side is at most $\left(e^{\alpha/4p}\right)^p |v_i|^p \leq e^{\alpha/4} |v_i|^p \leq (1 + \frac{\alpha}{2}) |v_i|^p$, where the last inequality uses Fact 18. This implies that $\left| |v_i|^p - |v_i'|^p \right| \leq \frac{\alpha}{2} |v_i|^p$. As a corollary, we obtain

$$\sum_{i \in \mathcal{I}_{\text{small}}} \left| |v_i|^p - |v_i'|^p \right| \leq \frac{\alpha}{2} \sum_{i \in \mathcal{I}_{\text{small}}} |v_i|^p \leq \frac{\alpha}{2} \|v\|_p^p .$$

For $i \in \mathcal{I}_{\text{large}}$, we have

$$\sum_{i \in \mathcal{I}_{\text{large}}} |v_i'| \leq \sum_{i \in \mathcal{I}_{\text{large}}} |v_i| + |\Delta_i| \leq \sum_{i \in \mathcal{I}_{\text{large}}} \left(1 + \frac{4p}{\alpha} \right) |\Delta_i| \leq \sum_{i \in \mathcal{I}_{\text{large}}} \frac{8p}{\alpha} |\Delta_i| \leq \frac{8p}{\alpha} \|\Delta\|_1 \leq \left(\frac{\alpha k}{4} \right)^{1/p}.$$

This implies that, due to the convexity of the function $f(x) \stackrel{\text{def}}{=} x^p$ for $p \geq 1$,

$$\sum_{i \in \mathcal{I}_{\text{large}}} |v_i'|^p \leq \left(\sum_{i \in \mathcal{I}_{\text{large}}} |v_i'| \right)^p \leq \frac{\alpha k}{4}.$$

The same bound holds for v, i.e.,

$$\sum_{i \in \mathcal{I}_{\text{large}}} |v_i|^p \leq \left(\sum_{i \in \mathcal{I}_{\text{large}}} |v_i'| \right)^p \leq \frac{\alpha k}{4}.$$

Combining these bounds, we get a bound on the sum of differences in coordinates in $\mathcal{I}_{\text{large}}$

$$\sum_{i\in\mathcal{I}_{\text{large}}} \left||v_i|^p - |v'_i|^p\right| \leq \sum_{i\in\mathcal{I}_{\text{large}}} |v_i|^p + |v'_i|^p \leq \frac{\alpha k}{4} + \frac{\alpha k}{4} = \frac{\alpha k}{2} \leq \frac{\alpha}{2}\|v\|_p^p,$$

where the last inequality follows from the fact that v is k-dense, and therefore, $\|v\|_p^p \geq k$.

Overall, combining our knowledge for both $\mathcal{I}_{\text{small}}$ and $\mathcal{I}_{\text{large}}$,

$$\begin{aligned}\left|\|v\|_p^p - \|v'\|_p^p\right| = \left|\sum_{i=1}^n |v_i|^p - |v'_i|^p\right| &\leq \sum_{i=1}^n \left||v_i|^p - |v'_i|^p\right| \\ &\leq \sum_{i\in\mathcal{I}_{\text{small}}} \left||v_i|^p - |v'_i|^p\right| + \sum_{i\in\mathcal{I}_{\text{large}}} \left||v_i|^p - |v'_i|^p\right| \\ &\leq \alpha\|v\|_p^p.\end{aligned}$$

This immediately implies our main claim. □

6 Proof of the Main Result

We start by restating our main result. This version has more details than the simplified version, which was presented as Theorem 1 in the introduction.

Theorem 20 (Adversarially robust moment estimation algorithm, full version of Theorem 1)**.** *Algorithm 1 is a $(1\pm\alpha)$-approximation adversarially robust streaming algorithm for the p^{th}-moment with success probability $1-\delta-O(n^{-3})$ for streams of length m. The space complexity of the algorithm for different values of p is specified in Table 1.*

value of p	**$\widetilde{O}(m^\mu n^\rho)$ space**	**detailed space complexity**
$p\in[0,1]$	$\mu=\frac{1}{3}$, $\rho=0$	$O(m^{1/3}\cdot\alpha^{-5/3}\cdot\log^{5/3}(m/\alpha\delta))\cdot\text{polylog}(n)$
$p\in(1,2]$	$\mu=\frac{p}{2p+1}$, $\rho=0$	$O\left(m^{p/(2p+1)}\cdot\alpha^{-(5p+1)/(2p+1)}\cdot\log^{5p/(2p+1)}(m/(\alpha\delta))\right)$ $\cdot\text{polylog}(n)$
$p=2$	$\mu=\frac{2}{5}$, $\rho=0$	$O\left(m^{2/5}\cdot\alpha^{-11/5}\log^{4/3}(m/\alpha\delta)\right)\cdot\text{polylog}(n)$
$p\in(2,\infty)$	$\mu=\frac{p}{2p+1}$, $\rho=1-\frac{5}{2p+1}$	$O\left((mp)^{p/(2p+1)}\cdot n^{1-5/(2p+1)}\cdot\alpha^{-(5p+1)/(2p+1)}\right)$ $\cdot\log^{10p/(6p+3)}(m/(\alpha\delta))\cdot\text{polylog}\,n$

Table 1: Space complexity of Algorithm 1. See Theorem 20.

Implementation notes for Algorithm 1. We start with a few implementation details. First, to ensure low memory usage, one has to maintain a sparse representation of v, i.e., store only the non-zero coordinates in an easily searchable data structure such as a balanced binary search tree. This is possible, because as soon as v becomes $4T$-dense, we stop maintaining it explicitly. Hence this part of the algorithm uses only $O(T)$ words of space.

We also avoid discussing numerical issues, and assume that for any integer $j \in [m]$, we can compute a good approximation to j^p in $O(1)$ time, and also that summing such sufficiently good approximations still yields a sufficiently good approximation. In order to efficiently update M_{exact}, while avoiding accumulating numerical errors (due to a sequence of additions and subtractions), one can create a balanced binary tree in which we sum approximations for $|v_i|^p$ for all non-zero coordinates v_i. Updating one of them then requires only updating the sums on the path to the root. This path is of length $O(\log T)$, and hence this requires updating at most $O(\log T)$ intermediate sums, each being a result of adding two values.

Proof of our main result. We are now ready to move on to the proof of our main result, which collects all the tools that we have developed throughout the paper.

Proof of Theorem 20. We first prove our algorithm's correctness conditioning on three assumptions, and then we prove that these assumptions hold with high probability. Finally, we analyze the space complexity of the algorithm. Throughout the proof $v^{(i)}$ denotes the value of v after the i^{th} update. Our assumptions are:

1. All invocations of $\mathcal{A}_{\text{sparse}}$, $\mathcal{A}_{\text{approx}}$, and $\mathcal{A}_{\text{density}}$ are successful. That is, the following events occur: $\mathcal{A}_{\text{sparse}}$ correctly recovers v (provided that v is $\lceil 4T \rceil$-sparse), $\mathcal{A}_{\text{approx}}$ returns a $(1 \pm \alpha/4)$-approximation to the p^{th}-moment of v whenever it is queried, and $\mathcal{A}_{\text{density}}$ returns a $(1 \pm .25)$-approximation to the number of non-zero coordinates in v whenever it is queried.

2. If v is T-sparse, then *regime* = SPARSE.

3. If *regime* = SPARSE, then v is $4T$-sparse.

Correctness under the assumptions. By Assumption 3, $\mathcal{A}_{\text{sparse}}$ is only invoked when v is $4T$-sparse. By the first item, each such invocation correctly recovers v. Hence, at the first time step of every time interval such that *regime* = SPARSE, the algorithm has a sparse representation of v (as discussed in Section 6) and this continues for the duration of the sparse interval. Therefore, for the duration of an interval where *regime* = SPARSE, M_{exact} correctly approximates $\|v\|_p^p$, and therefore all outputs of the algorithm are $(1 \pm \alpha)$-approximations to $\|v\|_p^p$.

Consider now a time interval where *regime* = DENSE. By the above discussion, at the first time step such that *regime* = DENSE, it holds that $\|v\|_p^p > 4T$ (since during SPARSE intervals the algorithm exactly knows $\|v\|_p^p$). We claim that at all time steps where *regime* = DENSE, k_{approx} is a $(1 \pm \alpha)$-approximation of $\|v\|_p^p$. Fix a maximal time interval $[t, t']$ such that *regime* = DENSE. Let $M_{\text{approx}}(t)$ denote the value of M_{approx} at time step t, and let i_0 denote the time step in which this value was computed (note that $i_0 \leq t$). Further let $i_1, \ldots, i_\ell$ denote all the time steps within $[t, t']$ in which $\mathcal{A}_{\text{approx}}$ was invoked. Now consider any two subsequent time steps i_j, i_{j+1} for $j \in [1, \ell - 1]$, and any time step $z \in [i_j, i_{j+1}]$. It holds that $i_{j+1} - i_j = \mathit{interval}$. For $p \in [0, 1]$, since $z - i_j \leq \mathit{interval} = \alpha T/4$, it holds that $\|v^{(i_j)} - v^{(z)}\|_1 \leq \alpha T/4$, and by Lemma 16 it follows that $\left\|v^{(z)}\right\|_p^p \in (1 \pm \alpha/4) \left\|v^{(i_j)}\right\|_p^p$. For $p \geq 1$, $\|v^{(i_j)} - v^{(z)}\|_1 \leq \mathit{interval} = \frac{\alpha}{32p}(\frac{\alpha T}{16})^{1/p}$, and Lemma 19 implies that $\left\|v^{(z)}\right\|_p^p \in (1 \pm \alpha/4) \left\|v^{(i_j)}\right\|_p^p$. By the assumption that all invocations of $\mathcal{A}_{\text{approx}}$ are successful, $M_{\text{approx}}(i_j) \in (1 \pm \alpha/4) \|v\|_p^p$. Hence, it follows that for both possible regimes of p, $M_{\text{approx}}(z) \in (1 \pm \alpha) \left\|v^{(z)}\right\|_p^p$ for any $z \in [i_1, t']$. Similar reasoning proves that at time step i_0, v was k-dense, and hence for any $z \in [t, i_1]$, it holds holds that $M_{\text{approx}}(z) \in (1 \pm \alpha) \left\|v^{(z)}\right\|_p^p$. Therefore,

for any $z \in [t, t']$ such that $[t, t']$ is a maximal time interval with *regime* = DENSE, it holds that the output of the algorithm is a $(1 \pm \alpha)$-approximation of $\|v\|_p^p$. Hence, it remains to prove that the assumptions hold with high probability.

The assumptions hold. For item 1, by Theorem 9, with probability $1 - O(n^{-3})$, $\mathcal{A}_{\text{sparse}}$ is successful on *all* invocations.[4] Algorithm $\mathcal{A}_{\text{density}}$ is queried $O(m/\lfloor T/10 \rfloor)$ times, so by the setting of q it holds that, with probability at least $1 - \delta/2$, all queries return a $(1 \pm .25)$-approximation of $\|v\|_0^0$. Similarly, $\mathcal{A}_{\text{approx}}$ is invoked $O(m/\textit{interval})$ times, and hence, with probability at least $1 - \delta/2$, all queries return a $(1 \pm \alpha/4)$-approximation of $\|v\|_p^p$. Hence, Assumption 1 holds with probability at least $1 - \delta - O(n^{-3})$. We henceforth condition on this event.

Now consider Assumption 2, that if v is T-sparse then *regime* = SPARSE. Clearly this holds from the beginning of the stream and until the first time that $\|v\|_0^0 > 4T$, since up to that point everything is deterministic and exact. Assume towards contradiction that there exists time steps such that v is T-sparse and *regime* = DENSE, and let t be the earliest one. Let i_0 be the closest step prior to t in which k_{approx} was recomputed (by invoking $\mathcal{A}_{\text{density}}$). By the conditioning on Assumption 1 holding, $k_{\text{approx}}(i_0) \in \left[.75 \left\|v^{(i_0)}\right\|_0^0, 1.25 \left\|v^{(i_0)}\right\|_0^0\right]$. Since at time step i_0, the regime was not changed to SPARSE, it also holds that $k_{\text{approx}}(i_0) > 2T$. Hence, $\left\|v^{(i_0)}\right\|_0^0 > (4/5) \cdot k_{\text{approx}}(i_0) > (8/5)T$. Clearly, in $\lfloor T/10 \rfloor$ updates $\|v\|_0^0$ cannot change by more than $T/10$. Hence, $\left\|v^{(t)}\right\|_0^0 > T$, implying that v is not T-sparse, and so we have reached a contradiction.

We turn to Assumption 3. By the conditioning on Assumption 1 holding, $\mathcal{A}_{\text{sparse}}$ always correctly recovers v, implying that as long as *regime* = SPARSE, v is exactly known to the algorithm. Therefore, it can be exactly detected when $\|v\|_0^0$ becomes greater than $4T$, at which point the regime is being set to DENSE. Hence, up until that point, $\|v\|_0^0 \leq 4T$ and v is $4T$-sparse.

Space complexity analysis. We now analyze the space complexity of our algorithm. By Theorem 7 and Lemma 11, for $q = m/\lfloor T/10 \rfloor$, Algorithm $\mathcal{A}_{\text{density}}$ requires $O(\sqrt{m/T} \cdot \alpha^{-2} \cdot \log^{5/2}(m/(\alpha\delta)) \cdot \log n = \widetilde{O}(\sqrt{m/T})$ space. This complexity is always bounded by the following terms. By Theorem 9, $\mathcal{A}_{\text{sparse}}$ requires $O(k \operatorname{polylog} n) = O(T \operatorname{polylog} n)$ space. We continue to analyze the space complexity due to $\mathcal{A}_{\text{approx}}$ separately for different regimes of p.

For $p \in [0, 1]$, by Theorem 7 and Lemma 11, Algorithm $\mathcal{A}_{\text{approx}}$ with $q = m/\textit{interval} = O(m/(\alpha T))$, requires $O(\sqrt{m/(\alpha T)} \cdot \log^{5/2}(m/(\alpha\delta)) \cdot \alpha^{-2} \cdot \log n = \widetilde{O}(\sqrt{m/T})$ space. Hence, setting T to balance between the space complexities of $\mathcal{A}_{\text{sparse}}$ and $\mathcal{A}_{\text{approx}}$ (and $\mathcal{A}_{\text{density}}$), we get that the space complexity of (α, δ)-approximating the p^{th} moment for $p \in [0, 1]$ is

$$O\left(m^{1/3} \cdot \alpha^{-5/3} \cdot \log^{5/3}(m/(\alpha\delta)\right) \cdot \operatorname{polylog}(n) = \widetilde{O}(m^{1/3}).$$

For $p \in (1, \infty]$, it holds that $\textit{interval} = \Theta((\alpha/p) \cdot (\alpha T)^{1/p})$. We again consider two separate regimes. First, $p \in (1, 2]$. By Lemma 11 and Theorem 7, since $q = \sqrt{m/\textit{interval}}$, $\mathcal{A}_{\text{approx}}$ takes $O\left(\sqrt{mp/(\alpha(\alpha T)^{1/p})} \cdot \log^{3/2}(m/(\alpha\delta)) \cdot \alpha^{-2} \cdot \log m \cdot \log(1/\delta)\right) = \widetilde{O}(\sqrt{m/T^{1/p}})$ space. Equating this term with the $O(T \cdot \operatorname{polylog} n)$ space required by $\mathcal{A}_{\text{sparse}}$, results in a space complexity of

$$O\left(m^{p/(2p+1)} \cdot \alpha^{-(5p+1)/(2p+1)} \cdot \log^{5p/(2p+1)}(m/(\alpha\delta))\right) \cdot \operatorname{polylog}(n) = \widetilde{O}(m^{p/(2p+1)})$$

for $p \in (1, 2]$. Finally, we consider the regime $p \in (2, \infty)$. In this regime, we again have $q = (m/\textit{interval}) = O(mp/(\alpha(\alpha T)^{1/p}))$, and so by Lemma 11 and Theorem 7, the space usage of $\mathcal{A}_{\text{approx}}$

[4] We note that with probability $1 - O(n^{-3})$ (over its set of initial random coins) Algorithm $\mathcal{A}_{\text{sparse}}$ correctly recovers *all* k-sparse vectors, and hence its output is correct for any (adversarial) input stream.

is $O(\sqrt{mp/(\alpha(\alpha T)^{1/p})} \cdot \log^{3/2}(m/(\alpha\delta))(\alpha^{-2} \cdot \log(1/\delta) \cdot n^{1-2/p})) = \widetilde{O}(\sqrt{m/T^{1/p}} \cdot n^{1-2/p})$. Hence, equating this with the $O(T \operatorname{polylog} n)$ space required by $\mathcal{A}_{\text{sparse}}$, we get a space complexity of

$$O\left((mp)^{p/(2p+1)} \cdot n^{1-5/(2p+1)} \alpha^{-(5p+1)/(2p+1)} \cdot \log^{5p/(2p+1)}(m/(\alpha\delta))\right) \cdot \operatorname{polylog} n$$
$$= \widetilde{O}((mp)^{p/(2p+1)} \cdot n^{1-5/(2p+1)}).$$

This concludes the proof. □

Acknowledgments

This work was inspired by the conversation of Cameron Musco and David Woodruff—after David Woodruff's talk at the STOC 2021 workshop on adversarially robust streaming [Rob21a, Rob21b]—about when tracking moments in the general turnstile model is difficult and involves a large flip number. The authors wish to thank Rajesh Jayaram and Uri Stemmer for useful discussions.

References

[ABEC21] Daniel Alabi, Omri Ben-Eliezer, and Anamay Chaturvedi. Bounded space differentially private quantiles, 2021. Preprint; Extended abstract appeared in Theory and Practice of Differential Privacy (TPDP) 2021.

[ABED+21] Noga Alon, Omri Ben-Eliezer, Yuval Dagan, Shay Moran, Moni Naor, and Eylon Yogev. Adversarial laws of large numbers and optimal regret in online classification. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 447–455, New York, NY, USA, 2021. ACM.

[ACSS21] Idan Attias, Edith Cohen, Moshe Shechner, and Uri Stemmer. A framework for adversarial streaming via differential privacy and difference estimators. *CoRR*, abs/2107.14527, 2021.

[BEJWY20] Omri Ben-Eliezer, Rajesh Jayaram, David P. Woodruff, and Eylon Yogev. A framework for adversarially robust streaming algorithms. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, page 63–80, New York, NY, USA, 2020. ACM.

[BEY20] Omri Ben-Eliezer and Eylon Yogev. The adversarial robustness of sampling. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, page 49–62, New York, NY, USA, 2020. ACM.

[BHM+21] Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. Adversarial robustness of streaming algorithms through importance sampling. *CoRR*, abs/2106.14952, 2021.

[BNS+21] Raef Bassily, Kobbi Nissim, Adam D. Smith, Thomas Steinke, Uri Stemmer, and Jonathan R. Ullman. Algorithmic stability for adaptive data analysis. *SIAM J. Comput.*, 50(3), 2021.

[DFH+15] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 117–126. ACM, 2015.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography (TCC)*, pages 265–284. Springer Berlin Heidelberg, 2006.

[DRV10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 51–60. IEEE Computer Society, 2010.

[GI10] Anna Gilbert and Piotr Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

[GSTV07] Anna C. Gilbert, Martin J. Strauss, Joel A. Tropp, and Roman Vershynin. One sketch for all: fast algorithms for compressed sensing. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 237–246. ACM, 2007.

[GW18] Sumit Ganguly and David P Woodruff. High probability frequency moment sketches. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[HKM+20] Avinatan Hassidim, Haim Kaplan, Yishay Mansour, Yossi Matias, and Uri Stemmer. Adversarially robust streaming algorithms via differential privacy. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[HW13] Moritz Hardt and David P. Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 121–130, New York, NY, USA, 2013. ACM.

[Jay21] Rajesh Jayaram. *Sketching and Sampling Algorithms for High-Dimensional Data*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2021.

[KMNS21] Haim Kaplan, Yishay Mansour, Kobbi Nissim, and Uri Stemmer. Separating adaptive streaming from oblivious streaming using the bounded storage model. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 94–121, Cham, 2021. Springer International Publishing.

[KNW10a] Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1161–1178. SIAM, 2010.

[KNW10b] Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 41–52, 2010.

[MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103, 2007.

[Rob21a] STOC 2021 workshop: Robust streaming, sketching, and sampling. https://rajeshjayaram.com/stoc-2021-robust-streaming-workshop.html, June 2021. Accessed: 2021-08-31.

[Rob21b] STOC 2021 workshop: Robust streaming, sketching, and sampling. https://youtu.be/svgv-xw9DZc, June 2021. Accessed: 2021-09-06.

[Ste21] Uri Stemmer. Adversarial streaming, differential privacy, and adaptive data analysis. https://youtu.be/Whu-6IVYFXc, March 2021. Accessed: 2021-08-31.

[WZ20] David P. Woodruff and Samson Zhou. Adversarially robust and sliding window streaming algorithms without the overhead. *CoRR*, abs/2011.07471, 2020. To appear in FOCS 2021.