Sublinear-Time Sparse Recovery, and Its Power in the Design of Exact Algorithms

Sublinear-Time Sparse Recovery, and its Power in the Design of Exact Algorithms

A dissertation presented

by

Vasileios Nakos

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

May 2019

Dissertation Advisor:
Jelani Nelson

Author:
Vasileios Nakos

Sublinear-Time Sparse Recovery, and its Power in the Design of Exact Algorithms

Abstract

In the sparse recovery problem one wants to reconstruct an approximatelly k-sparse vector $x \in \mathbb{R}^n$ using time and number of measurements that are sublinear, i.e. way less n, ideally nearly linear in k. Depending on the setting, measurements correspond to one of the following: linear combinations of the entries of x, a non-linear function of some linear function of x , Fourier coefficients, the logical OR of entries in x. In this thesis I describe several new contributions to the field of sparse recovery, as well as indicate how sparse recovery techniques can be of great significance in the design of exact algorithms, outside of the scope of the problems they first were created for.

- **Standard** Sparse Recovery:
  - The state of the art ℓ_2/ℓ_2 scheme: optimal measurements, $O(k \log^2(n/k))$ decoding time and $O(\log(n/k))$ column sparsity, via a new, non-iterative approach.
- **Non-linear** Sparse Recovery:
  - The first sublinear-time algorithm for one-bit compressed sensing.
  - A set of $O(k \log^c n)$-time algorithms for compressed sensing from intensity only measurements. The algorithms use $O(k \log n)$ measurements, being the first sublinear-time measurement-optimal algorithms for the problem.
- **Sparse Fourier Transform**

  - A nearly sample-optimal algorithm for ℓ_∞/ℓ_2 Sparse Fourier Transform in any dimension.
  - A nearly optimal sublinear-time deterministic algorithm for ℓ_∞/ℓ_1 Sparse Fourier Transform.

- **Design of Exact Algorithms**
  - A nearly optimal algorithm for sparse polynomial multiplication.
  - An almost optimal deterministic algorithm for the Modular-SubsetSum problem, running in time $m \cdot 2^{O(\sqrt{\log m \cdot \log\log m})}$.
  - A nearly optimal Las Vegas algorithm for the Modular-SubsetSum problem, running in time $\widetilde{O}(m)$.
  - An (almost) output-sensitive algorithm for the SubsetSum problem.

Contents

List of Tables

List of Figures

Acknowledgments

First of all, I would like to thank my advisor Jelani Nelson for his enormous generosity regarding travelling and funding, as well all the math he taught me during my Ph.D. years. Special thanks to all my collaborators: Karl Bringmann, Yi Li, Zhao Song, Zhengyu Wang and David Woodruff. I would also like to thank John Girash for answering numerous logistical questions. The rest you know who you are.

Introduction

0.1 Standard Compressed Sensing/Sparse Recovery

Compressed Sensing, or sparse recovery, is a powerful mathematical framework the goal of which is to reconstruct an approximately k-sparse vector $x \in \mathbb{R}^n$ from linear measurements $y = \Phi x$, where $\Phi \in \mathbb{R}^{m \times n}$. The most important goal is to reduce the number of measurements m needed to approximate the vector x, avoiding the linear dependence on n. In discrete signal processing, where this framework was initiated [CRT06, Don06], the core principle that the sparsity of a signal can be exploited to recover it using much fewer samples than the Shannon-Nyquist Theorem. We refer to the matrix Φ as the sketching or sensing matrix, and $y = \Phi x$ as the sketch of vector x.

Sparse recovery is the primary task of interest in a number of applications, such as image processing [TLW+06, LDP07, DDT+08], design pooling schemes for biological tests [ECG+09, DWG+13], pattern matching [CEPR07], combinatorial group testing [SAZ09, ESAZ09, KBG+10], localizing sources in sensor networks [ZBSG05, ZPB06], as well as neuroscience [GS12]. Furthermore, not surprisingly, tracking heavy hitters in data streams, also known as frequent items, can be captured by the sparse recovery framework [Mut05, CH09, KSZC03, Ind07]. In practice, streaming algorithms for detecting heavy hitters have been used to find popular destination addresses and heavy bandwidth users by AT&T [CJK+04] or answer “iceberg queries” in databases [FSGM+99].

Sparse recovery attracts researchers from different communities, from both theoretical and practical perspective. During the last ten years, hundreds of papers have been published by theoretical computer scientists, applied mathematicians and electrical engineers that specialize in compressed sensing. While numerous algorithms with running time linear in the universe size n are known, [Don06, CRT06, IR08, NT09a, BIR08, BD09a, SV16] to name a few, our goal is to obtain algorithms that are sublinear, something that is crucial in many applications.

The desirable quantities we want to optimize may vary depending on the application. For example, in network management, x_i could denote the total number of packets with destination i passing through a network router. In such an application, storing the sketching matrix explicitly is typically not a tenable solution, since this would lead to an enormous space consumption; the number of possible IP addresses is 2^{32}. Moreover, both the query and the update time should be very fast, in order to avoid congestion on the network. Incremental updates to x come rapidly, and the changes to the sketch should also be implemented very fast; we note that in this case, even poly-logarithmic factors might be prohibitive. Interested readers can refer to [KSZC03, EV03] for more information about streaming algorithms for network management applications.

Sparse recovery schemes that are optimal across all axis are a challenge and an important theoretical and practical problem. For most sparse recovery tasks, we have algorithms that achieve different trade-offs for the various parameters of interest. One exception is the ℓ_∞/ℓ_2 guarantee, for which the breakthrough work of Larsen, Nelson, Nguyên and Thorup [LNNT16] shows that this trade-off is unnecessary.

"The goal of that research is to obtain encoding and recovery schemes with good compression rate (i.e., short sketch lengths) as well as good algorithmic properties (i.e., low encoding, update and recovery times)." – *Anna Gilbert and Piotr Indyk* [GI10]

0.1.1 Previous work

Since compressed sensing has been extensively studied in the literature for more than a decade, different guarantees of interest have been suggested (x_{-k} is the vector that occurs after zeroing out every i that does not belong among the largest k coordinates). In what follows $x \in \mathbb{R}^n$ is the vector we want to sketch, x' is the approximation to x, k is the sparsity and ϵ is the fineness of the approximation. The most extensively studied error guarantees are the following.

- $\ell_2/\ell_2 : \|x - x'\|_2 \leq (1+\epsilon)\|x_{-k}\|_2$.
- $\ell_1/\ell_1 : \|x - x'\|_1 \leq (1+\epsilon)\|x_{-k}\|_1$.
- $\ell_\infty/\ell_2 : \|x - x'\|_\infty \leq \frac{1}{\sqrt{k}}\|x_{-k}\|_2$.
- $\ell_\infty/\ell_1 : \|x - x'\|_\infty \leq \frac{1}{k}\|x_{-k}\|_1$.
- $\ell_2/\ell_1 : \|x - x'\|_2 \leq (1+\epsilon)\frac{1}{\sqrt{k}}\|x_{-k}\|_1$.

Regarding the universality of the scheme, there are two different guarantees, one is the for-all guarantee and the other is the for-each guarantee. In the for-all guarantee, one wants to design a sketch that gives the desired result for all vectors $x \in \mathbb{R}^n$. In the for-each guarantee, one wants to design a distribution over sketches that gives the desired result for a fixed vector $x \in \mathbb{R}^n$. We note that ℓ_∞/ℓ_2, ℓ_2/ℓ_2 are impossible in the for-all model, unless $\Omega(n)$ measurements are used [CDD09]. The standard approach for the for-all guarantee is via RIP matrices, satisfying the so-called Restricted Isometry Property. In what follows, we will refer to the for-each model, unless stated otherwise.

The first set of schemes that initiated the research on compressed sensing are given in [CRT06, Don06]. There the authors show, for any $x \in \mathbb{R}^n$, given $y = \Phi x$, it is possible to satisfy the ℓ_2/ℓ_1 guarantee for all vectors, if Φ is a Gaussian matrix with $O(k\log(n/k))$ rows. The schemes in [CM06, CCF02] achieve the ℓ_∞/ℓ_2 guarantee with $O(k\log n)$ measurementz, matching known lower bounds [JST11], $O(n\log n)$ decoding time and $O(\log n)$ update time. The state of the art for ℓ_∞/ℓ_2 is [LNNT16], which gives optimal number of measurements, sublinear decoding time, $O(\log n)$ update time and $1/\operatorname{poly}(n)$ failure probability. Price and Woodruff [PW11] show that in order to get ℓ_2/ℓ_2 with constant failure probability $< 1/2$ with the output being exactly k-sparse output requires $\Omega(\epsilon^{-2}k)$ measurements. They also showed non-k-sparse output requires $\Omega(\epsilon^{-1}k\log(n/k))$ measurements in the regime $\epsilon > \sqrt{k\log n/n}$, and gave an upper bound of $O(\epsilon^{-1}k\log n)$ measurements, showing thus a separation in the measurement complexity between k-sparse and $O(k)$-sparse output. Later, in the breakthrough work of Gilbert, Li, Porat and Strauss [GLPS10] an algorithm that runs in sublinear time, and has $O(\log(n/k)\log^2 k)$ column sparsity, was devised. On generic norms, nearly optimal bounds have been given by Backurs, Indyk, Razenshteyn and Woodruff [BIRW16]. We note, however, that their schemes are not computationally efficient: they have exponential running time, except in the case of Earth-Mover-Distance, which has time polynomial in n and $\log^k n$.

Measurements. The number of measurements corresponds to physical resources: memory in monitoring devices of data streams, number of screens in biological applications, or number of filters in dynamic spectrum access (DSA) of radio signal [HMT+13].

In applications such as medical imaging, it is crucial to reduce the number of measurements, since the radiation used in CT scans could potentially increase cancer risks for patients. For instance, [PSL+12] showed that a positive association between radiation exposure from CT scans in childhood and subsequent risk of leukemia and brain tumors.

For more applications, we refer the readers to [QBI+13].

Encoding Time. Designing algorithms with fast update/encoding time is a well-motivated task for streaming algorithms, since the packets arrive at an extremely fast rate [TZ12]; even logarithmic factors are crucial in these applications. Also in digital signal processing applications, in the design of cameras or satellites which demand rapid imaging, when we observe a sequence of images that are close to each other, we may not need to encode the new signal from the beginning, rather than encode only that part which differs from the current signal; the delay is then defined by the update time of our scheme. Moreover, in Magnetic Resonance Imaging (MRI) update time or encoding time defines the time the patient waits for the scan to happen. Improvement of the runtime has benefits both for patients and for healthcare economics [LDSP08].

A natural question is the following: what are the time limitations of our data structures, regarding update time? Regarding the streaming setting, the first lower bounds are given in [LNN15] for non-adaptive algorithms. An algorithm is called non-adaptive if, during updates, the memory cells are written and read depend only on the index being updated and the random coins tossed before the stream is started to being processed. The lower bounds given concern both randomized and deterministic algorithms; the relevant bounds to sparse recovery are for ℓ_p/ℓ_q estimation. However, for constant failure probability their results do not give anything useful, since their lower bounds start to kick in when the failure probability becomes very small, namely $o(2^{-\sqrt{m \cdot \log n}})$.

For the column sparsity (which could be smaller than update time, and hence the lower bounds in [LNN15] might not apply[1]), the only known lower bounds are known for RIP matrices, which are used in the for-all setting. To the best of our knowledge, the first non-trivial lower bounds were given by Nachin [Nac10], and then extended by Indyk and Razenshteyn in [IR13] for RIP-1 model-based compressed sensing matrices. Lower bounds for the column sparsity of RIP-2 matrices were given in Nelson and Nguyên [NN13], and then to RIP-p matrices in Allen-Zhu, Gelashvili and Razenshteyn [AZGR16]. Roughly speaking, the lower bounds for ℓ_2 indicate that if one aims for optimal measurements, $m = k\log(n/k)$, in the regime $k < n/\log^3 n$, one cannot obtain column sparsity better than $\Omega(m)$. This indicates that the for-all case should be significantly worse, in terms of column sparsity, than the for-each case.

Decoding Time. Another very important quantity we want to minimize is the time needed to reconstruct the approximation of x from its compressed version. This quantity is of enormous significance in cases where the universe size is huge and we cannot afford to iterate over it. This is often the case in networking applications, where the universe size is the number of distinct IP addresses. In MRI applications the decoding time corresponds to the time needed to reconstruct the image after the scan has been performed. Decoding time is highly important also in satellite systems, modern radars and airspace surveillance, where compressed sensing have found extensive application [End10].

[1] the lower bounds in [LNN15] also depend heavily on the streaming model, so they do not transfer necessarily to all scenarios where sparse recovery finds application.

Our Contribution. We give the state of the art algorithm for ℓ_2/ℓ_2 compressed sensing, which achieves optimal sample complexity, has decoding better than any previous attempt and is always sublinear (as long as the number of measurements remain sublinear), and achieves $O(\log(n/k))$ column sparsity, significantly better than previous work. Previous work on sublinear-time compressed sensing employed an iterative procedure, recovering the heavy coordinates in phases. We completely depart from that framework, and give the first sublinear-time ℓ_2/ℓ_2 scheme which achieves the optimal number of measurements without iterating; this new approach is the key step to our progress. Towards that, we satisfy the ℓ_2/ℓ_2 guarantee by exploiting the heaviness of coordinates in a way that was not exploited in previous work. Via our techniques we obtain improved results for various sparse recovery tasks, and indicate possible further applications to problems in the field, to which the aforementioned iterative procedure creates significant obstructions.

0.2 Sparse Fourier Transform

When the measurements are not arbitrarily chosen, but have to be chosen from a structured ensemble, the most important subtopic is the sparse Fourier transform, where one desires to reconstruct a k-sparse vector from Fourier measurements. In Optics imaging [Goo05, Voe11] and Magnetic resonance imaging (MRI) [ASSN08], the physics [Rey89] of the underlying device restricts us to the Fourier ensemble, where the sparse Fourier problem becomes highly relevant. In fact, one of the initial motivations of Candes, Romberg and Tao came out due to the aforementioned applications. The number of samples plays a crucial role: they determine the amount of radiation a patient receives in CT scans, and taking fewer samples can reduce the amount of time the patient needs to stay in the machine. The framework has found its way in practical life-changing applications. Software includes the COMPRESSED SENSING GRAB-VIBE, CS SPACE, CS SEMAC and CS TOF by Siemens [Sie], as well as Compressed Sense by Phillips [Phi]. Its incorporation in the MRI technology allows faster acquisition rates, depiction of dynamic processes or moving organs, as well as acceleration of MRI scanning up to a factor of 40. On the webpage of SIEMENS Healthineers, for example, one can see the following, as well as numerous similar statements.

This allows bringing the advantages of Compressed Sensing GRASP-VIBE *to daily clinical routine.*

- *Perform push-button, free-breathing liver dynamics.*
- *Overcome timing challenges in dynamic imaging and respiratory artifacts.*
- *Expand the patient population eligible for abdominal* MRI*.*

The Fourier transform is in fact ubiquitous: image processing, audio processing, telecommunications, seismology, polynomial multiplication, SUBSET SUM and other textbook algorithms are a few of the examples where the Fast Fourier Transform finds applications. The Fast Fourier Transform by Cooley and Tukey [CT65] runs in $O(n \log n)$ time, and has far-reaching applications in all of the aforementioned cases. It is thus expected that algorithms which exploit sparsity assumptions about the input, and can outperform FFT in applications are of high practical value. More specifically, sparsity assumptions have given researchers the hope of defeating the FFT algorithm of Cooley and Tukey, in the special (but of high practical value) case where the signal is approximately sparse. Moreover, since FFT serves as an important computational primitive, and has been recognized as one of the 10 most important algorithms of the 20th century [Cip00], every place where it has found application can possibly be benefited from a faster algorithm. The main intuition and hope is that signals arising in practice often exhibit certain structures, such as concentration of energy in a small number of Fourier coefficients.

Generally, the two most important parameters one would like to optimize are the sample complexity, i.e. the numbers needed to obtain from the time domain, as well as the time needed to approximate the Fourier Transform.

Two different lines of research exist for the problem: the one focuses solely on sample complexity, while the other tries to achieve sublinear time while keeping the sample complexity as low as possible. The first line of research operates via the renowned Restricted Isometry Property (RIP), which proceeds by taking random samples and solving a linear/convex program, or an iterative thresholding procedure [CT06, DDTS06, TG07, BD08, DM08, RV08, BD09b, BD09a, NT09b, NV09, GK09, BD10, NV10, Fou11, Bou14, HR16]. The analysis of the algorithms is performed in the following way, in two steps. The first step ensures that, after sampling an appropriate number of points from the time domain, the inverse DFT matrix restricted on the rows indexed by those points acts as a near isometry on the space of k-sparse vectors. All of the state of the art results [CT06, RV08, Bou14, HR16] employ chaining arguments to make the analysis of this sampling procedure as tight as possible. The second part is how to exploit the aforementioned near-isometry property to find the best k-sparse approximation to the signal. There the approaches either follow an iterative procedure which gradually denoise the signal [BD08, NT09b, NV09], or perform ℓ_1 minimization [CT06], a method that promotes sparsity of solutions.

The second line of research tries to implement arbitrary linear measurements via sampling Fourier coefficients [GL89, Man92, KM93, GGI+02, AGS03, GMS05, Iwe08, Iwe10, HIKP12a, HIKP12b, LWC13, Iwe13, PR14, IKP14, IK14, Kap16, Kap17, CI17, BZI17, MZIC17, LN19] and use sparse functions (in the time domain) which behave like bandpass filters in the frequency domain. The seminal work of Kapralov [Kap17] achieves $O(k \log n)$ samples and running time that is some log factors away from the sample complexity. This would be the end of the story, apart from the fact that this algorithm does not scale well with dimension, since it has an exponential dependence on d. Indeed, in many applications, one is interested in higher dimensions, rather than the one-dimensional case. The main reason[2] why this curse of dimensionality appears is due to the lack of dimension-independent ways to construct functions that approximate the ℓ_∞ ball and are sufficiently sparse in the time domain. A very nice work of Kapralov, Velingker and Zandieh [KVZ19] tries to remedy that by combining the standard execution of FFT with careful aliasing, but their algorithm works in a noiseless setting, and has a polynomial, rather than linear, dependence on k; the running time is polynomial in $k, \log n$ and the exponential dependence is avoided. It is an

[2]But not the only one: pseudorandom permutations for sparse FT in high dimensions also incur an exponential loss, and it is not known whether this can be avoided.

important and challenging question whether a robust and more efficient algorithm can be found.

We note that in many applications, such as MRI or computed tomography (CT), the main focus is the sample complexity; the algorithms that have found their way to industry are, to the best of our knowledge, not concerned with sublinear running time, but with the number of measurements, which determine the acquisition time, or in CT the radiation dose the patient receives. Lastly, we bring to the readers' attention the recent work on sparse Fourier transform in the continuous setting, see [Iwe10, Iwe13, Iwe13, BCG+14, PS15, CKPS16, AKM+18].

Our Contribution (1). We give a randomized algorithm which uses $O(k \log k \log n)$ samples, it is dimension-free, it operates for any universe size, and achieves the strongest ℓ_∞/ℓ_2 guarantee, while running in time comparable to the Fast Fourier Transform. All previous algorithms proceed either via the Restricted Isometry Property or via filter functions. Our approach totally departs from the aforementioned techniques, and we believe is a fresh look to the sparse Fourier transform problem.

Our Contribution (2). We give a polynomial time algorithm to find a set of $O(k^2 \log^2 n)$ samples, which allow computing the best k-term approximation to the Sparse Fourier Transform of a signal in time $O(k^2 \log^3 n)$. Our approach also yields an algorithm with $O(k^2 \log n)$ sample complexity but $O(nk \log n)$ running time, as well a nearly optimal construction of an incoherent matrix, using rows of the DFT matrix.

0.3 Non-Linear Compressed Sensing

0.3.1 Compressed Sensing from Intensity-Only Measurements

In recent years a variant of the sparse recovery problem, called *compressive phase retrieval*, has become an active topic, which seeks to recover a sparse signal $x \in \mathbb{R}^n$ (or $\mathbb{C}^n$) from the *phaseless measurements* $y = |\Phi x|$ (or $y = |\Phi x| + \nu$ with post-measurement noise), where

$|z|$ denotes a vector formed by taking the absolute value of every coordinate of z. The primary goal remains the same, i.e. to use as fewer measurements as possible. Such type of measurements arises in various fields such as optical imaging [SEC+15] and speech signal processing [RJ93]. There has been rich research in geometric algorithms for this problem (see, e.g. [CSV13, CLS15b, CLS15a, GWX16, IPSV16, IVW17]) that run in at least polynomial time while there have been relatively few sublinear time algorithms – [CBJC14, IVW16, PYLR17, Nak17a] are the only algorithms to the best of our knowledge. Most existing algorithms consider sparse signals, and thus such sublinear time algorithms have a flavour of code design, akin to Prony's method. Among the sublinear-time algorithms, [CBJC14] considers sparse signals only, [PYLR17] considers sparse signals with random post-measurement noise, [IVW16] allows adversarial post-measurement noise but has poor recovery guarantee, [Nak17a] considers near-sparse real signals with no post-measurement noise but achieves constant-factor approximation and thus outperforms all other sublinear-time algorithms for real signals. The approach in [Nak17a] employs combinatorial techniques more widely used in the theoretical computer science literature for the classical sparse recovery problem. The later work of [LN18] has improved upon [Nak17a], giving a set of new algorithms that are sample-optimal and run in sublinear time.

More quantitatively, suppose that the decoding algorithm $\mathcal{R}$, given input $y = |\Phi x|$, outputs an approximation $\widehat{x}$ to x, with the guarantee that the approximation error $d(x, \widehat{x})$ is bounded from above. When $x \in \mathbb{R}^n$, both x and $-x$ yield the same measurements, the approximation error $d(x, \widehat{x})$ has therefore the form $d(x, \widehat{x}) := \min\{\|x - \widehat{x}\|, \|x + \widehat{x}\|\}$ for some norm $\|\cdot\|$. When $x \in \mathbb{C}^n$, the approximation error $d(x, \widehat{x}) = \min_{\theta \in [0,2\pi)} \|x - e^{i\theta}\widehat{x}\|$. Specifically we consider the following three types of error guarantee:

- (ℓ_∞/ℓ_2) $\min_{\theta \in [0,2\pi)} \|x - e^{i\theta}\widehat{x}\|_\infty \leq \frac{1}{\sqrt{k}}\|x_{-k}\|_2$ for $x \in \mathbb{C}^n$;
- (ℓ_2/ℓ_2) $\min_{\theta \in [0,2\pi)} \|x - e^{i\theta}\widehat{x}\|_2 \leq (1+\epsilon)\|x_{-k}\|_2$ for $x \in \mathbb{C}^n$;
- (ℓ_1/ℓ_1) $\min\{\|x - \widehat{x}\|_1, \|x + \widehat{x}\|_1\} \leq (1+\epsilon)\|x_{-k}\|_1$ for $x \in \mathbb{R}^n$,

where x_{-k} denotes the vector formed by zeroing out the largest k coordinates (in magnitude) of x. Note that when x is noiseless, that is, when $x_{-k} = 0$, all guarantees mean exact recovery of x, i.e., $\widehat{x} = x$.

Besides the error guarantees, the notions of for-all and for-each in the sparse recovery problems also extend to the compressive phase retrieval problem. In a for-all problem, the measurement matrix Φ is chosen in advance and will work for all input signals x, while in a for-each problem, the measurement matrix Φ is usually random such that for each input x, a random choice of Φ works with a good probability.

Our Contribution. We give ℓ_∞/ℓ_2 and ℓ_2/ℓ_2 schemes that achieve $O(k \log n)$ measurements and $O(k \log^c n)$ running time. Previous algorithms either assumed that the signal is sparse, either ran in significantly worse time and/or satisfied a weaker guarantee. Along the way, we also develop a new $O(k)$-measurement and $O(k \log k)$-time algorithm for exactly k-sparse signals.

0.3.2 One-Bit Compressed Sensing

In modern acquisition systems measurements need to be quantized: that it means that we have access only to $y = Q(Ax)$ for some $Q : \mathbb{R}^m \to \mathcal{A}^m$ [BB08]. In other words, Q maps every element of the encoded vector to an element to a finite alphabet $\mathcal{A}$. The most common paradigm is when $\mathcal{A} = \{-1, 1\}$ and

$$y = \text{sign}(Ax),$$

where the sign function is applied to any element of the vector. In hardware systems such as the analog-to-digital converter (ADC), quantization is the primary bottleneck limiting sample rates [Wal99, LRRB05]. Moreover, as indicated in [LRRB05], the sampling rate has to decrease exponentially in order for the number of bits to be increased linearly. Furthmore, power consumption is dominated by the quantizer, leading to increased ADC costs. Thus, the one-bit compressed sensing framework provides a way to disburden the quantization bottleneck by reducing the sampling rate, i.e. the total number of measurements [BB08].

Apart from having important applications, the problem of one-bit compressed sensing is also interesting from a theoretical perspective, as it is a natural and fundamental question on high-dimensional geometry. One can think of it in the following way: can we construct a set of hyperplanes $\mathcal{H}$ such that we can approximate the direction a k-sparse vector $x \in \mathbb{R}^n$ given $\text{sign}(\langle x, h \rangle)$, for all $h \in \mathcal{H}$? If we want a uniform guarantee, i.e. being able to approximate the direction of x for every x, this means that every region defined by the hyperplanes and the sphere must have "small' diameter. Othewise, if we want to reconstruct the direction of x with some target probability, then it suffices that most regions defined by the sphere and the hyperplane have small diameter. The latter formulation is very closely related to the problem of random hyperplane tesselations [PV14].

Previous Work

The problem of one-bit compressed sensing was introduced in [BB08], and has received a fair amount of attention till then; one can see [LXZL18] for details. Efficient algorithms, which proceed by by solving linear or convex programs when the sensing matrix consists of gaussians, appear in [PV13a, PV13b, GNJN13]. Algorithms that are based on iterative hard-thresholding have been suggested in [JDDV13, JLBB13]. Moreover, the paper of Plan and Vershyin [PV14] studies the very relevant problem of random hyperplane tesselations. The authors in [GNJN13, ABK17] give also combinatorial algorithms for support-recovery from one-bit measurements using combinatorial structures called union-free families.

The work of [BFN$^+$16] introduces schemes for one-bit compressed sensing for the scenario where the underlying singal is sparse with respect to an overcomplete dictionary rather than a basis; this scenario is common in practice. Researchers have also tried to reduce the reconstruction error by employing different techniques and under different models. One approach suggested is Sigma-Delta quantization [KSW16, GLP$^+$10]. If adaptivity is allowed and, moreover, the measurements take the form of threshold signs, the authors in [BFN$^+$17] show that the reconstruction error can be made exponentially small.

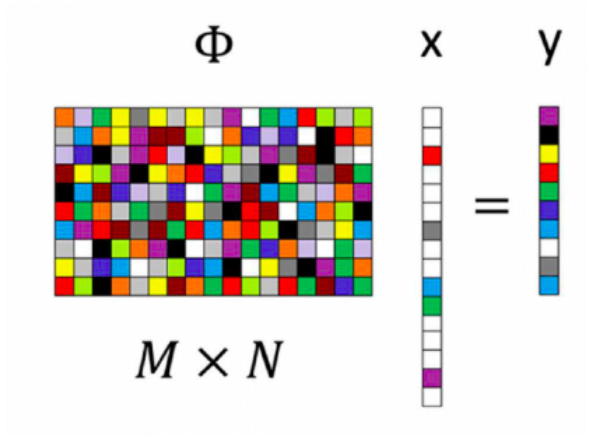


Figure 1: *An illustration of the standard sparse recovery problem in its simplest form. The vector x to be sensed has a few non-zero coordinates, and Φ is the sensing matrix, with much less rows than columns. Given y one wants to reconstruct x.*

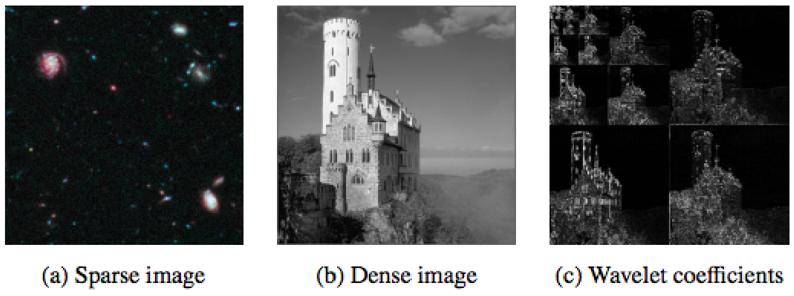


Figure 2: *Examples of sparsity. Subfigure (a) contains an image from the Hubble space telescope. The image is sparse because it contains a small number of bright pixels, which contain the important information in order to reconstruct the image. The castle in Subfigure (b) is not sparse, but its wavelet coefficients in Subfigure (c) give a much sparser representation.*

Our Contribution. We give an algorithm for one-bit compressed satisfying what we call $\delta - \ell_2/\ell_2$ guarantee, which uses $O(k \log n + \delta^{-2}k)$ measurements, and runs in $\text{poly}(k \cdot \log n)$ time. This is the first algorithm for the problem running in sublinear time, and even compares with the best super-linear time algorithm in terms of sample complexity; precisely, for $k \leq n^{1-\gamma}$ it is uses less measurements.

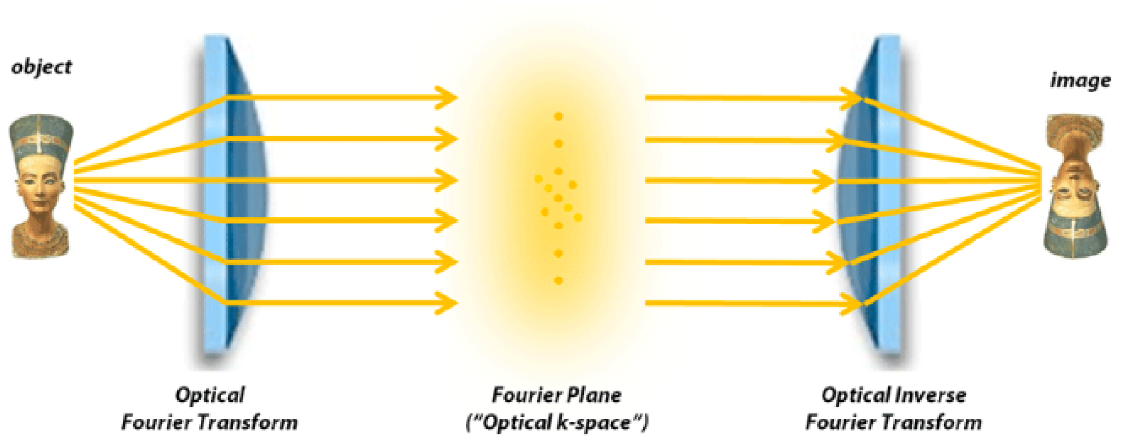


Figure 3: *Light waves reflected off the sculpture enter the first lens and are refracted (bent) as a function of their spatial frequencies. Lower frequency waves are only weakly refracted, passing straight through the center of the lens. Higher frequency waves are refracted more at the edges. The "output" of the first lens is a series of unfocused wave fronts with higher spatial frequencies toward the periphery and lower frequencies toward the center. These waves constructively and destructively interfere. The first lens has thus performed an "optical" Fourier transformation of the incident light rays. If you put your head midway between the two lenses (at the so-called Fourier plane) and looked back towards the Queen, you would see nothing except a vague diffuse glow representing the average intensity of light entering the first lens. The light waves are unfocused and would not form a picture on your retina. You are in "optical" k-space. The second lens reverses this procedure, reassembling the waves dispersed in optical k-space back to their original relationships. The second lens thus performs an inverse Fourier transform, allowing the creation of a focused image.*

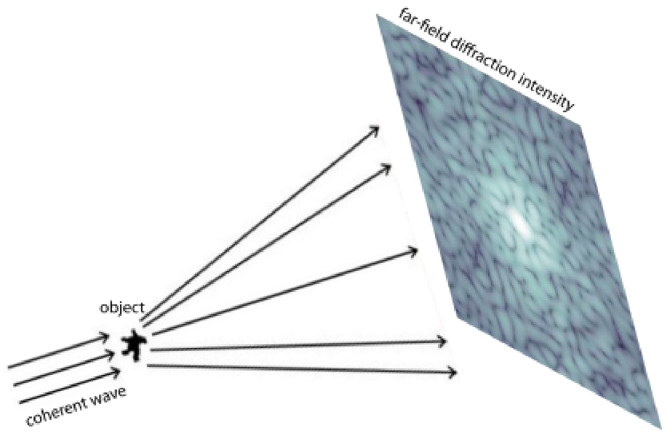


Figure 4: *A typical setup of Coherent Diffractive Imaging, which gives rise to a signal recovery problem from phaseless measurements. In the basic CDI setup (forward scattering), an object is illuminated by a quasi-monochromatic coherent wave, and the diffracted intensity is measured. When the object is small and the intensity is measured far away, the measured intensity is proportional to the magnitude of the Fourier transform of the wave at the object plane, with appropriate spatial scaling.*

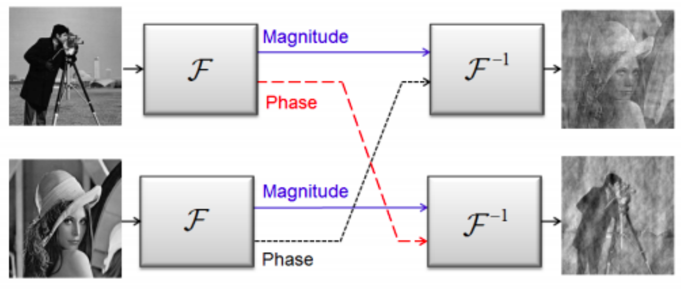


Figure 5: *A synthetic example demonstrating the importance of phase in reconstructing a signal from its Fourier transform.*

0.4 Sparse Polynomial Multiplication

Multiplying two polynomials is a fundamental computational primitive, with multiple applications in computer science. Using the Fast Fourier Transform, one can perform multiplication of polynomials stored as vectors of floating point numbers, in time $O(n \log n)$, where n is a bound on the largest degree.

An important and natural question is whether, and under which circumstances, a faster algorithm can be invented. Researchers have tried to obtain algorithms that beat the $O(n \log n)$-time bound, when the two polynomials are sparse, i.e. the number of non-zero terms in each polynomial is at most s. Interestingly, some ideas from the relevant literature have found applications in computer algebra packages such as Maple, Mathematica and Singular, including ways to represent and store polynomials [Maz01, MP14, MP15, GR16].

When two polynomials have at most k coefficients, the trivial algorithm gives $O(k^2 \log n \log s)$ time, which is already and improvement for $s \leq \sqrt{n}$. It is important though to obtain an algorithm that is output-sensitive, i.e. runs in nearly linear time with respect to k, the number of non-zero coefficients in the product. A result of Cole and Hariharan [CH02] obtains an algorithm that runs in $O(k \log^2 n)$ time, when the coefficients of the two polynomials are non-negative. A data structure for carefully allocating and de-allocating memory has been designed in [Yan98], trying to tackle the problem of memory handling can be the main bottleneck in complexity of sparse multiplication in practical scenarios. The aforementioned algorithm is based on a heap, an idea which was also lead to implementations developed in [MP07, MP09, MP14, MP15]. The authors in [MP09] develop a parallel algorithm for multiplying sparse distributed polynomials, where each core uses a heap of pointers to multiply parts of polynomials, exploiting its L3 cache. The same authors in [MP14] have created a data structure suitable for the Maple kernel, that allows for obtains significant performance in many Maple library routines.

When the support of the product is known or structured, work in [Roc08, Roc11, VDHL12, VDHL13] indicates how to perform the multiplication fast. Using techniques from spare interpolation, Aarnold and Roche [AR15] have given an algorithm that runs in time that is nearly linear in the "structural sparsity" of the product, i.e. the sumset of the supports of the two polynomials. When there are not "too many" cancellations, this is roughly the same as the size of the support of the product, and the above algorithm is quite efficient. However, in the presence of a considerable amount of cancellations in the product, the aforementioned algorithm becomes sub-optimal. Removing this obstacle seems to be the final step, and has been posed as an open problem in the excellent survey of [Roc18].

In this thesis, we resolve the aforementioned open question, giving an algorithm that is nearly optimal in the size of the input plus the size of the output. Due to its small computational complexity and simplicity, we expect our algorithm to be implemented in modern computer algebra software.

We note that one can use the rather heavy hammer of the sparse Fourier transform [GMS05, HIKP12a, HIKP12b, Kap16, Kap17, KVZ19] to obtain nearly optimal algorithms for sparse polynomial multiplication, but these algorithms come at a cost, since they are way more complicated, and invoke the filter functions; functions which are sparse in the time domain and approximate the ℓ_∞ box in the frequency domain.

Our Contribution. We give a clean, nearly optimal for multiplying two sparse polynomials with integer coefficients. Our algorithm runs in time which is proportional to the size of the input plus the size of the output.

0.5 Subset Sum

SubsetSum is one of the most important problems in computer science, and is taught in almost every undregraduate course. In this problem we are given a set S of n integers and a target t, and are asked to find a subset of S that sums up to t. The problem has been established as NP-complete by Karp [Kar72], and belonged to his initial list of NP-complete problems. Belmman's classical algorithm from 1957 [Bel57] solves the problem in pseudopolynomial $O(n \cdot t)$ time, and it's an archetypical example of algorithm design via dynamic programming. Another classical algorithm is the "meet-in-the-middle" algorithm of [HS74] , which gives $O(2^{n/2})$ time, regardless of the value of t. Using RAM parallelism Pisinger showed how to shave a $\log t$ factor from the pseudopolynomial algorithm [Pis03], thus being the first improvement over Bellman's classical approach. If all elements are bounded by M, Pisinger has also showed how to obtain a pseudopolynomial solution in $O(nM)$ time. Apart from being a cornerstone in algorithm design, SubsetSum has also played an important role in cryptography: Merkel and Hellman [MH78] based their cryptosystem on it, something that initiated work in cryptostystems based on Knapskack, see [Sha84, BO88, CR88, Odl90, IN96].

The SubsetSum problem has given rise to a plethora of algorithmic techniques, some of which have been recorded in books devoted to the specific problem [KPP04, MT90]. The problem still attracts a lot of researches, with important algorithmic contributions happening the last 10 years [O'B11, LMS11, BCJ11, DDKS12, AKKM13, GS17, AKKN15, AKKN16, LWWW16, BGNV17, Ned17, Bri17, KX17, JW18, ABHS19, ABJ+19]. The work of [BGNV17] shows that with polynomial space one can beat the trivial bound of 2^n, while in Merlin-Arthur setting Nederlof showed that the running time can become $T^{1/2}n^{O(1)}$. Very recently, the work of Koiliaris and Xu [KX17] gave the first deterministic algorithm for SubsetSum that runs in time $\widetilde{O}(\min\{\sqrt{n}t, t^{4/3}\})$, while the almost independent work of Bringmann [Bri17] gave a randomized $\widetilde{O}(t+n)$-time algorithm. Two years later, another $\widetilde{O}(t+n)$ algorithm [JW18] showed up, which proceeds by cleverly manipulating formal series. The running time of [Bri17] and [JW18] is optimal under the Strong Exponential

Time Hypothesis, as proved in [ABHS19].

Possibly the most important generalization of Subset Sum is the ModularSubsetSum problem, where the addition operation is performed over $\mathbb{Z}_m$. The structural properties of the problem have been investigated in the field of Additive Combinatorics for more than half a century[EGZ61, Ols68, Ols75, Alo87, NSV08, Sze70, Vu08]. The dynamic programming approach to ModularSubsetSum runs in time $O(mn)$, but, interestingly enough, going beyond this barrier requires new ideas, as well as different techniques from the latest improvements in the non-modular case. The fastest deterministic algorithm runs in time $\widetilde{O}(\min\{\sqrt{n}m, m^{5/4}\})$ [KX17], while the fastest randomized algorithm due to Axiotis et.al. [ABJ+19] runs in time $\widetilde{O}(m+n)$; the latter algorithm matches a conditional lower bound from from [ABHS19], similar to the standard Subset Sum problem. The deterministic approach of [KX17] carefully partitions the input into sets that can be covered by arithmetic progressions, then solves each arithmetic progression separately and combines the results with multiple FFTs. The algorithm of [ABJ+19] takes a totally different route, and manages to improve Bellman's dynamic programming approach by using a sophisticated hashing approach to compute the new attainable subset sums after insertion of every element; interestingly enough, their approach crucially exploits the modularity of the problem, and does not extend to Subset Sum.

Our contribution (1). We give the first deterministic algorithm for Modular Subset Sum that runs in $O(m^{1+o(1)})$ time; in specific our algorithm runs in time that is proportional to the number of all attainable sums times an $m^{o(1)}$ factor. This almost matches the randomized algorithm of [ABJ+19] and the lower bound of [ABHS19], and is polynomially better than the previous deterministic algorithm by Koiliaris and Xu [KX17]. Along the way we obtain a state of the art deterministic algorithm for sumset computation, which should be of independent interst.

Our approach also yields state of the art result or the randomized version of the problem. We obtain a Las Vegas algorithm running in $\widetilde{O}(m)$ time, a mild improvement over the algorithm of [ABJ+19] which was Monte Carlo. We also give a novel, nearly optimal Las

Vegas algorithm for sumset computation, which we believe is significantly simpler than the previous Las Vegas algorithm for the problem [CH02].

Our improvements are obtained by delving deeper into the additive structure of Modular Subset Sum than previous work. Rougly speaking, our algorithms proceed by computing all attainable subset sums in a bottom-up fashion and carefully define a "terminating condition"; when the condition is satisfied, this forces the solution space to have a specific additive structure, which we then exploit.

Our Contribution (2). We give an almost output sensitive randomized algorithm for the classical Subset Sum problem. Precisely, let $\mathcal{S}(S,t)$ be the set of all attainable subset sums of S which are at most t. Our algorithm runs in time proportional to (ignoring logarithmic factors) $|\mathcal{S}(S, t + t/\operatorname{poly}(\log t))|$. The textbook algorithm of Bellman runs in time $O(|S| \cdot \mathcal{S}(S,t))$, while the algorithm of Bringmann [Bri17] runs in time $\widetilde{\Theta}(t)$. Thus, although the latter matches a conditional lower bound from the k-Clique problem [ABHS19], in many interesting cases the textbook dynamic programming algorithm can be much better. Moreover, both algorithms proceed by computing $\mathcal{S}(S,t)$, the set of all subset sums. In light of the above, our algorithm should be viewed as making considerable progress in obtaining the best of both worlds.

Chapter 1

Standard Compressed Sensing

1.1 ℓ_2/ℓ_2 Compressed Sensing; Without Iterating

1.1.1 Our result

Our main result is a novel scheme for ℓ_2/ℓ_2 sparse recovery. Our contribution lies in obtaining better decoding time, and $O(\log(n/k))$ column sparsity via new techniques. The problem of improving the column sparsity to $O(\log(n/k))$ was explicitly stated in [GLPS10] as an open problem. Moreover, as an important technical contribution, we introduce a different approach for sublinear-time optimal-measurement sparse recovery tasks. Since this iterative loop is a crucial component of almost all algorithms in sublinear-time compressed sensing [IPW11, PS12, HIKP12a, GNP+13, IKP14, Kap16, GLPS17, CKSZ17, Kap17, LNW18, NSWZ18], we believe our new approach and ideas will appear useful in the relevant literature, as well as be a starting point for re-examining sparse recovery tasks under a different lens, and obtaining improved bounds.

1.1.2 Notation

For $x \in \mathbb{R}^n$ we let $H(x,k)$ to be the set of the largest k in magnitude coordinates of x. We also write x_S for the vector obtained after zeroing out every $x_i, \notin S$, and $x_{-k} = x_{[n]\setminus H(x,k)}$. We use $\|\cdot\|_p$ to denote the ℓ_p norm of a vector, i.e. $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$.

Reference	Measurements	Decoding Time	Encoding Time
[Don06, CRT06]	$k\log(n/k)$	LP	$k\log(n/k)$
[CCF02, CM06]	$\epsilon^{-2}k\log n$	$\epsilon^{-1}n\log n$	$\log n$
[NT09a]	$k\log(n/k)$	$nk\log(n/k)$	$\log(n/k)$
[CM04]	$\epsilon^{-2}k\log^2 n$	$\epsilon^{-1}k\log^c n$	$\log^2 n$
[CCF02, CM06]	$\epsilon^{-2}k\log^c n$	$\epsilon^{-1}k\log^2 n$	$\log^c n$
[GLPS10]	$\epsilon^{-1}k\log(n/k)$	$\epsilon^{-1}k\log^c n$	$\log(n/k)\cdot\log^2 k$
Our result	$\epsilon^{-1}k\log(n/k)$	$\epsilon^{-1}k\log^2(n/k)$	$\log(n/k)$

Table 1.1: *(A list of ℓ_2/ℓ_2-sparse recovery results). We ignore the "O" for simplicity. LP denotes the time of solving Linear Programs [CLS19], and the state-of-the-art algorithm takes n^ω time where ω is the exponent of matrix multiplication. The results in [Don06, CRT06, NT09a] do not explicitly state the ℓ_2/ℓ_2 guarantee, but their approach obtains it by an application of the Johnson-Lindenstrauss Lemma; they also cannot facilitate $\epsilon < 1$, obtaining thus only a 2-approximation. The c in previous work is a sufficiently large constant, not explicitly stated, which is defined by probabilistically picking an error-correcting code of short length and iterating over all codewords. We estimate $c \geq 4$. We note that our runtime is (almost) achieved by [HIKP12a], but our running time is always sublinear in n, in contrast to [HIKP12a] which can be up to $n\log n$.*

1.1.3 Technical statements

We proceed with the definition of the ℓ_2/ℓ_2 sparse recovery problem.

Problem 1.1.1 (ℓ_2/ℓ_2 sparse recovery)**.** *Given parameters ϵ, k, n, and a vector $x \in \mathbb{R}^n$. The goal is to design some matrix $\Phi \in \mathbb{R}^{m\times n}$ and a recovery algorithm $\mathcal{A}$ such that for $y = \Phi x$, $x' = A(\Phi, y))$ x' satisfies*

$$\|x' - x\|_2 \leq (1+\epsilon) \min_{k\text{-sparse } z\in\mathbb{R}^n} \|z - x\|_2.$$

We primarily want to minimize m (which is the number of measurements), the running time of $\mathcal{A}$ (which is the decoding time) and column sparsity of Φ.

In table 1.1, we provide a list of the previous results and compare with ours. Here, we formally present our main result.

Theorem 1.1.2 (stronger ℓ_2/ℓ_2 sparse recovery)**.** *There exists a randomized construction of a linear sketch $\Phi \in \mathbb{R}^{m\times n}$ with $m = O(\epsilon^{-1}k\log(n/k))$ and column sparsity $O(\log(n/k))$, such* *that given $y = \Phi x$, we can find an $O(k)$-sparse vector $x' \in \mathbb{R}^n$ in $O(m \cdot \log(n/k))$ time such that*

$$\|x' - x\|_2 \leq (1+\epsilon) \min_{k\text{-sparse } z \in \mathbb{R}^n} \|z - x\|_2.$$

holds with $9/10$ probability.

Remark 1.1.3. *In the regime where k is very close to n, for example $k = n/\mathrm{poly}(\log n)$, we get an exponential improvement on the column sparsity over [GLPS10]. In many applications of compressed sensing, this is the desired regime of interest, check for example Figure 8 from [BI08]: $n = 71,542$ while $m \geq 10,000$, which corresponds to k being very close to n.*

Remark 1.1.4. *As can be inferred from the proof, our algorithms runs in time $O((k/\epsilon)\log^2(\epsilon n/k) + (k/\epsilon)\log(1/\epsilon))$, which is slightly better than the one stated in Theorem 1.1. The algorithm in [HIKP12a] achieves also the slightly worse running time of $O((k/\epsilon)\log n \log(n/k))$. That algorithm was the first algorithm that achieved running time $O(n \log n)$ for all values of k, ϵ for which the measurement complexity remained sublinear, smaller than γn, for some absolute constant γ. A careful inspection shows that our algorithm achieves running time that is **always** sublinear, as long as the measurement complexity remains smaller than γn.*

1.1.4 Overview of techniques and difference with previous work

This subsection is devoted to highlighting the difference between our approach and the approach of [GLPS10]. We first give a brief high-level description of the state of the art algorithm before our work, then discuss our techniques, and try to highlight why the previous approach could not obtain the stronger result we present in this paper. Lastly, we show how our ideas can be possibly applied to other contexts.

Summary of [GLPS10].

The algorithm of [GLPS10] consists of $O(\log k)$ rounds: in the r-th round the algorithm finds a constant fraction of the remaining heavy hitters. Beyond this iterative loop lies the following idea about achieving the ℓ_2/ℓ_2 guarantee: in order to achieve it, you can find all but $\frac{k}{3^r}$ heavy hitters i such that $|x_i|^2 = \Omega(\frac{2^r \epsilon}{k}\|x_{-k}\|_2^2)$. This means that the algorithm is allowed to "miss" a small fraction of the heavy hitters, depending on their magnitude. For example, if all heavy hitters are as small as $\Theta(\sqrt{\epsilon/k}\|x_{-k}\|_2)$, a correct algorithm may even not find any of them. This crucial observation leads naturally to the main iterative loop of combinatorial compressed sensing, which, as said before, loop proceeds in $O(\log k)$ rounds. Every round consists of an identification and an estimation step: in the identification step most heavy hitters are recognized, while in the estimation step most of them are estimated correctly. Although in the estimation step some coordinates might have completely incorrect estimates, this is guaranteed (with some probability) be fixed in a later round. The reason why this will be fixed is the following. If a coordinate i is badly estimated, then it will appear very large in the residual signal and hence will be identified in later rounds, till it is estimated correctly. One can observe that the correct estimation of that round for coordinate i cancels out (remedies) the mistakes of previous rounds on coordinate i. Thus, the identification and estimation procedure, which are interleaved, work complementary to each other. The authors of [GLPS10] were the first that carefully managed to argue that identifying and estimating a constant fraction of heavy hitters per iteration, gives the optimal number of measurements.

More formally, the authors prove the following iterative loop invariant, where $k_r = k3^{-r}, \epsilon_r = \epsilon_r 2^{-r}$ for $r \in [R]$ with $R = \log k$: Given $x \in \mathbb{R}^n$ there exists a sequence of vectors $\{x^{(r)}\}_{r\in[R]}$, such that $x^{(r+1)} = x^{(r)} - \widehat{x}^{(r)}$ and

$$\|(x - \widehat{x})_{-k_r}\|_2^2 \leq (1 + \epsilon_r)\|x_{-k_r}\|_2^2. \tag{1.1}$$

In the end, one can apply the above inequality inductively to show that

$$\|x - \sum_{r=1}^{R} \widehat{x}^{(r)}\|_2^2 \leq (1 + \epsilon)\|x_{-k}\|_2^2.$$

We now proceed by briefly describing the implementations of the identification and the estimation part of [GLPS10]. In the identification part, in which lies the main technical contribution of that work, every coordinate $i \in [n]$ is hashed to $O(k/\epsilon)$ buckets and in each bucket $O(\log(\epsilon n/k))$-measurement scheme based on error-correcting codes is used to identify a heavy hitter; the authors carefully use a random error-correcting code of length $O(\log\log(\epsilon n/k))$, so they afford to iterate over all codewords and employ a more sophisticated approach and use nearest-neighbor decoding. This difference is one of the main technical ideas that allow them to obtain $O(\epsilon^{-1}k\log(n/k))$ measurements, beating previous work, but it is also the main reason why they obtain $k \cdot \text{poly}(\log n)$ decoding time[1]: performing nearest neighbor decoding and storing the code per bucket incurs additional $\text{poly}(\log n)$ factors. Moreover, for every iteration r they need to repeat the identification scheme r times in order to bring down the failure probability to 2^{-r}, so that they afford a union-bound over all iterations. This leads to an additional $O(\log^2 k)$ factor in the update time. The estimation step consists of hashing to $O(k/\epsilon)$ buckets and repeating $O(\log(1/\epsilon))$ times. Since the identification step returns $O(k/\epsilon)$ coordinates, the $O(\log(1/\epsilon))$ repetitions of the estimation step ensure that at most $k/3$ coordinates out of the $O(k/\epsilon)$ will not be estimated correctly. This is a desired property, since it allows the algorithm to keep the $2k$ coordinates with the largest estimates, subtract them from x and iterate.

In the next section, we will lay out our approach which improves the decoding time and the column sparsity of [GLPS10]. The iterative procedure of [GLPS10] lies in the heart of most compressed sensing schemes, so we believe that this new approach could be applied elsewhere in the sparse recovery literature.

Our approach

As we mentioned before, our approach is totally different from previous work, avoiding the iterative loop that all algorithms before applied. Our algorithm consists of four steps, each one being a different matrix responsible for a different task. The first matrix, with a constant number of rows allows us to approximate the tail of the vector x, an approximation that will appear useful in the next step. The second matrix along with its decoding procedure, which should be regarded as the identification step, enables us to find a list L of size $O(k/\epsilon)$

[1]The authors do not specifically address the exponent in the $\text{poly}(\log n)$, but we estimate it to be ≥ 4.

that contains k coordinates[2], which are sufficient for the ℓ_2/ℓ_2 guarantee. This matrix has $O(\epsilon^{-1}k \cdot \log(\epsilon n/k)))$ rows. The third matrix with its decoding procedure, which should be regarded as the pruning step, takes the aforementioned list L, and prunes it down to $O(k)$ coordinates, which are sufficient for the ℓ_2/ℓ_2 guarantee. This matrix again has $O(\epsilon^{-1}k \cdot \log(1/\epsilon))$ rows, for a total of $O(\epsilon^{-1}k \log(n/k))$ rows. The last matrix is a standard set-query sketch.

Step 1: Tail Estimation

Lemma 1.1.5 (tail estimation). *Let $c_1 \geq 1$ denote some fixed constant. There is an oblivious construction of matrix $\Phi \in \mathbb{R}^{m\times n}$ with $m = O(\log(1/\delta))$ and column sparsity $O(\log(1/\delta))$ such that, given Φx, there is an algorithm that outputs a value $V \in \mathbb{R}$ in time $O(m)$ such that*

$$\frac{1}{10k}\|x_{-c_1\cdot k}\|_2^2 \leq V \leq \frac{1}{k}\|x_{-k}\|_2^2$$

holds with probability $1-\delta$.

Our first step towards the way for stronger sparse recovery is the design a routine that estimates the ℓ_2 norm of the tail of a vector $x \in \mathbb{R}^n$, which we believe might be interesting in its own right. More generally, our algorithm obtains a value V such that $\frac{1}{10k}\|x_{-c_1\cdot k}\|_p^p \leq V \leq \frac{1}{k}\|x_{-k}\|_p^p$, using $O(1)$ measurements. Here c_1 is some absolute constant. To obtain this result we subsample the vector at rate $\Theta(1/k)$ and then use a p-stable distribution to approximate the subsampled vector. While the upper bound is immediate, the Paley-Zygmund inequality does not give a sufficient result for the lower bound, so more careful arguments are needed to prove the desired result. We obtain our result by employing a random walk argument.

One additional possible application in sparse recovery applications where a two-stage scheme is allowed, e.g. [DLTY06, DDT+08], would be to first use the above routine to roughly estimate how many heavy coordinates exist, before setting up the measurements.

[2]We note that this term is exactly k, not $O(k)$. Although not important for our main result, it will be crucial for some of our applications of our techniques.

For example, we could first run the above routine for $k = 1, 2, 2^2, \ldots, 2^{\log n}$, obtaining values $V_1, V_2, V_4, \ldots, V_{\log n}$, and then use these values to estimate the size of the tail of the vector is, or equivalently approximate the size of the set of the heavy coordinates by a number k'. We can then run a sparse recovery algorithm with sparsity k'. Details can be found in Section 1.4.

Step 2: The Identification Step The goal of this step is to output a list L of size $O(k/\epsilon)$ that contains a set of k coordinates that are sufficient to satisfy the ℓ_2/ℓ_2 guarantee. The column sparsity we are shooting for at this point is $O(\log(\epsilon n/k))$, and the decoding time should be $O(m \log(\epsilon n/k))$.

Lemma 1.1.6 (identification sketch)**.** *There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with $m = O(\epsilon^{-1}k \cdot \log(\epsilon n/k))$ and column sparsity $O(\log(\epsilon n/k))$, such that given $y = \Phi x$, one can find a set L of size $O(k/\epsilon)$ in $O(m \log(\epsilon n/k))$ time, such that*

$$\exists T \subset S, |T| \leq k : \|x - x_T\|_2 \leq (1+\epsilon)\|x_{-k}\|_2,$$

holds with probability at least 9/10.

For this routine, we will set up a hierarchical separation of $[n]$ to trees. We will call this separation interval forest. we set $\tau = k/\epsilon$. Then we partition $[n]$ into τ intervals of length $q = n/\tau$, and set up an interval tree for each interval in the following way: every interval tree has branching factor

$$\frac{\log q}{\log \log q}$$

with the same height (this is consistent with the fact that every tree contains q nodes). At the leaves of every interval tree there are the nodes of the corresponding interval.

Our approach consists is now the following. For each level of the interval forest we hash everyone to k/ϵ buckets in the following way: if two coordinates i, i' are in the same interval (node of the interval forest) they are hashed to the same bucket. The above property can be regarded as "hashing interval to buckets". Moreover, every x_i is multiplied by a **Gaussian** random variable. We repeat this process for $\log\log(\epsilon n/k)$ per level.

The decoding algorithm is the following. First, we obtain a value V using the routine in Step 1, Lemma 1.1.5. Then we proceed in a breadth-first (or depth-first, it will make no difference) search manner and find an estimate for every interval, similarly to [LNNT16, CH09], by taking the median of the $\log\log(\epsilon n/k)$ buckets it participates to. There are two technical things one needs to show: First, we should bound the decoding time, as well as the size of the output list L. Second, we need to show that there exists a set T' of size at most k that satisfies the guarantee of the Lemma 1.1.6. For the first part, we show that the branching process defined by the execution of the algorithm is bounded due to the $\log\log(\epsilon n/k)$ repetitions per level. For the second part, we show that for every coordinate $i \in H(x,k)$ the probability that $i \in L$ is proportional to $k|x_i|^2/(\epsilon\|x_{-k}\|_2^2)$. Then we show that the expected ℓ_2^2 mass of coordinates $i \in L \setminus H(x,k)$ is $\epsilon\|x_{-k}\|_2^2$. This suffices to give the desired guarantee for Lemma 1.1.6.

We provide details in Section 1.2.

Step 3: The Pruning Step

Lemma 1.1.7 (pruning sketch)**.** *Let $c_2, c_3 > 1$ denote two fixed constants. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m\times n}$, with $m = O(\epsilon^{-1}k \cdot \log(1/\epsilon))$, with column sparsity $O(\log(1/\epsilon))$ such that the following holds :*
Suppose that one is given a (fixed) set $L \subseteq [n]$ such that

$$|L| = O(k/\epsilon), \qquad \exists T \subset L, |T| \le k : \|x - x_T\|_2 \le (1+\epsilon)\|x_{-k}\|_2.$$

Then one can find a set S of size $c_2 \cdot k$ in time $O(m)$, such that

$$\|x - x_S\|_2 \le (1 + c_3 \cdot \epsilon)\|x_{-k}\|_2$$

holds with probability $9/10$.

We will now prune the list L obtained from the previous step, to $O(k)$ coordinates. We are going to use $O(\epsilon^{-1}k \cdot \log(1/\epsilon))$ measurements. We hash every coordinate to k/ϵ buckets, combining with **Gaussians**, and repeating $O(\log(1/\epsilon))$ times. A similar matrix was also used in [GLPS10], but there the functionality and the analysis were very different; moreover, the authors used random signs instead of Gaussians. We will heavily exploit the fact that a standard normal g satisfies $\Pr[|g| < x] = O(x)$. The reasons why we need Gaussians is the following: if we have a lot of large coordinates which are equal and much larger than $\sqrt{\epsilon}\|x_{-k}\|_2$, we want to find all of them, but due to the use of random signs, they might cancel each other in a measurement. Switching to Gaussians is the easiest way of avoiding this undesirable case.

Our algorithm computes, for every $i \in L$, an estimate $\widehat{x}_i$ by taking the median of the $O(\log(1/\epsilon))$ buckets it participates to, and then keeps the largest $O(k)$ coordinates in magnitude to form a set S. We then show that these coordinates satisfy the ℓ_2/ℓ_2 guarantee. We will say that a coordinate is well-estimated if $|\widehat{x}_i| = \Theta(|x_i|) \pm \sqrt{\epsilon k^{-1}}\|x_{-k}\|_2$. For the analysis, we define a threshold $\tau = \|x_{-k}\|_2/\sqrt{k}$, and classify coordinates based on whether $|x_i| \geq \tau$ or not.

- In the case $|x_i| \geq \tau$ the expected mass of these coordinates $i \notin S$ is small;
- In the other case the number of coordinates i with $|x_i| < \tau$ are $O(k)$. This allows us to employ an exchange argument, similar to previous work, e.g. [PW11], but more tricky due to the use of Gaussians instead of random signs.

We note that the way we compute the estimates $\widehat{x}_i$ is different from previous work: one would expect to divide the content of a bucket that i hashed to by the coefficient assigned to x_i, in order to get an unbiased estimator, but this will not work. The details can be found in Section 1.3.

In the end, this step gives us a set S suitable for our goal, but **does not** give good estimations of the coordinates inside that set. For that we need another, standard step.

Step 4: Set Query We estimate every coordinate in S using a set query algorithm of Price [Pri11], obtaining the desired guarantee. This matrix needs only $O(k/\epsilon)$ measurements, and runs in $O(k)$ time, while having constant column sparsity.

Lemma 1.1.8 (set query, [Pri11]). *For any $\epsilon \in (0, 1/10]$. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with $m = O(k/\epsilon)$ and column sparsity $O(1)$, such that given $y = \Phi x$ and a set $S \subseteq [n]$ of size at most k, one can find a k-sparse vector $\widehat{x}_S$, supported on S in $O(m)$ time, such that*

$$\|\widehat{x}_S - x_S\|_2^2 \le \epsilon \|x_{[n] \setminus S}\|_2^2.$$

holds with probability at least $1 - 1/\operatorname{poly}(k)$.

Our Theorem 1.1.2 follows from the above four steps by feeding the output of each step to the next one. In the end, we rescale ϵ. More specifically, by the identification step we obtain a set L of size $O(k/\epsilon)$ which contains a subset of size k that satisfies the ℓ_2/ℓ_2 guarantee. Then, the conditions for applying the pruning step are satisfied, and hence we can prune the set L down to $O(k)$ coordinates, which satisfy the ℓ_2/ℓ_2 guarantee. Then we apply the set-query sketch to obtain estimates of these coordinates.

In what follows we ignore constant terms. The number of measurements in total is

$$\underbrace{1}_{\text{tail estimation}} + \underbrace{(k/\epsilon)\log(\epsilon n/k)}_{\text{identification step}} + \underbrace{(k/\epsilon)\log(1/\epsilon)}_{\text{pruning step}} + \underbrace{(k/\epsilon)}_{\text{set query}}.$$

The decoding time equals

$$\underbrace{1}_{\text{tail estimation}} + \underbrace{(k/\epsilon)\log(\epsilon n/k)\log(\epsilon n/k)}_{\text{identification step}} + \underbrace{(k/\epsilon)\log(1/\epsilon)}_{\text{pruning step}} + \underbrace{k}_{\text{set query}}.$$

The column sparsity equals

$$\underbrace{1}_{\text{tail estimation}} + \underbrace{\log(\epsilon n/k)}_{\text{identification step}} + \underbrace{\log(1/\epsilon)}_{\text{pruning step}} + \underbrace{1}_{\text{set query}}.$$

1.1.5 Possible applications of our approach to other problems

Exactly k-sparse signals. When the vector we have to output has to be k-sparse, and not $O(k)$-sparse, the dependence on ϵ has to be quadratic [PW11]. Our algorithm yields a state of the art result for this case, too. One can observe that the analysis of the algorithm in

Section 1.2 outputs a $O(k/\epsilon)$-sized set which contains a set T of size k that allows ℓ_2/ℓ_2 sparse recovery. Performing a CountSketch with $O(k/\epsilon^2)$ columns and $O(\log(k/\epsilon)$ rows, and following a standard analysis, one can obtain a sublinear algorithm with measurement complexity $O(\epsilon^{-1}k\log(\epsilon n/k) + \epsilon^{-2}k\log(k/\epsilon))$. This is an improvemnet on both the runtime and measurement complexity over previous work [CCF02].

Block-Sparse Signals. Our algorithm easily extends to block-sparse signals. For a signal of block size b we obtain sample complexity $O(k/(\epsilon b)\log(bn/k) + (k/\epsilon))$, with a running time nearly linear in k. This matches the sample complexity of previous super-linear algorithms [BCDH10, CIHB09], which also could not facilitate $\epsilon < 1$.

Phaseless Compressed Sensing. In Phaseless Compressed Sensing, one wants to design a matrix Φ with m rows, such that given $y = \Phi x$, one can find a vector $\widehat{x}$ such that $\min_{\theta\in[0,2\pi]}\|x - e^{i\theta}\widehat{x}\|_2 \leq (1+\epsilon)\|x_{-k}\|_2$. This problem has received a fair amount of attention [OYDS11, LV13, CBJC14, YLPR15, PYLR17, Nak17a, LN18], and the state of the art algorithm has $O(k\log n)$ measurement complexity [Nak17a, LN18]. One of the problems is that the iterative loop approach cannot be used here, since it is heavily based on the linearity of the sketch. However, our identification and pruning step do not use the linearity of the sketch, and work also with phaseless measurements. Previous algorithms such as [Nak17a, LN18] suffered a $k\log n$ factor in the number of measurements already from the first step, but this is avoidablue using our new approach. We hope to see an extension down this avenue that gives $O(k\log(n/k))$ measurements.

One-Bit Compressed Sensing. Another important subfield of Sparse Recovery is one-bit compressed sensing, where one has access only to one-bit measurements, i.e. $y = \text{sign}(Ax)$, where the sign function on vectors should be understood as pointwise application of the sign function on each entry. Sublinear algorithms appear in [Nak17b, Nak19], but they both do not obtain the optimal number of measurements in terms of k and n, which is $k\log(n/k)$, but rather the slightly suboptimal $k\log n$. One of the most important reasons is that the iterative loop cannot be implemented in such a scenario. It is a natural question whether our new approach can give the optimal number of measurements. The thresholding step, namely the part where we take use V to filter out non-heavy intervals cannot be implemented here, but perhaps there is still a way to make a similar argument. One first approach should be to show that sublinear decoding with optimal measurements is achieved using non-adaptive threshold measurements, such as in [KSW16] and [BFN$^+$17] (note that the latter one uses adaptive measurements though).

Sparse Fourier Transform. The standard approach to discrete sparse Fourier transform, is to implement linear measurements by using Fourier measurements [GGI$^+$02, GMS05, HIKP12a, HIKP12b, Iwe13, IKP14, IK14, Kap16, CKSZ17, Kap17]. The idea is to hash the spectrum to B buckets by carefully multiplying in the time-domain the vector x with a sparse vector z. In the frequency domain this corresponds to convolving the spectrum of the x with an approximation of the filter of an indicator function of an interval of length roughly B. Due to the Uncertainty Principle, however, one has to exchange measurement complexity and decoding time with the quality of the filter. For example, implementing hashing to buckets using "crude" filters leads to leakage in subsequent buckets, giving additional error terms. When iterating as usual, these errors accumulate and make identification much harder. The sophisticated approach of [Kap17] manages to design an iterative algorithm, in the same vein with previous algorithms, which takes $O(\epsilon^{-1}k \log n)$ measurements. It would be interesting to see if the approach we suggest avoids some of the problems created by this iterative loop, and can give simpler and faster sparse Fourier transform schemes. It would be interesting to obtain such a result even using adaptive measurements. The work [CKSZ17] has some interesting ideas in the context of block-sparse vectors that could be relevant.

1.2 The Identification Linear Sketch

The goal of this section is to prove the following result,

Theorem 1.2.1 (Restatement of Lemma 1.1.6)**.** *Let $C_L > 1$ be a fixed constant. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with*

$$m = O(\epsilon^{-1} k \log(\epsilon n / k)),$$

with column sparsity $O(\log(\epsilon n/k))$ such that given $y = \Phi x$, one can find in time $O(m \log(n/k))$ a set L of size $C_L \cdot k/\epsilon$, such that

$$\exists T \subset L, |T| \leq k : \|x - x_T\|_2 \leq (1+\epsilon)\|x_{-k}\|_2,$$

with probability $9/10$.

In Section 1.2.1, we provide the definition of sketching matrix Φ and present the decoding algorithm. We proved some concentration result in Section 1.2.2. We analyzed the running time of algorithm in Section 1.2.3. We proved the guarantees of the algorithm in Section 1.2.4. Finally, we bound the number of measurements in Section 1.2.5.

1.2.1 Design of the sketch and decoding algorithm

Notation	Choice	Statement	Parameter
C_H	4	Definition 1.2.3	H
C_R	100	Definition 1.2.4	R
C_B	10^5	Definition 1.2.5	B
C_0	10^3	Lemma 1.4.4	Blow up on tail size
C_L	10^4	Lemma 1.2.9	L
η	$1/9$	Lemma 1.2.6,1.2.7	Shrinking factor on V
ζ	$1/4000$	Lemma 1.2.6,1.2.7	$\zeta \leq \eta/400$

Table 1.2: *Summary of constants in Section 1.2, the column "Parameter" indicates which parameter is depending on that constant. Note that constants C_H, C_R, C_B, C_0, η are used in both algorithm and analysis, but constants C_L and ζ are only being used in analysis. C_L is the related to the guarantee of the output of the algorithm.*

We are going to use a hierarchical separation of $[n]$ into intervals. We will call this separation an interval forest.

Before discussing the matrix, we need the following definitions. We define

Definition 1.2.2 (size of the each tree in the forest)**.** *Let*

$$\tau = k/\epsilon,$$

assuming that $k/\epsilon \leq n/16$. The size of each tree in the forest is

$$q = n/\tau = n\epsilon/k$$

Definition 1.2.3 (degree and height of the interval forest)**.** *Let $C_H > 1$ be a sufficiently large constant such that*

$$(\log q/\log\log q)^{C_H \log q/\log\log q} \geq q.$$

Let D denote the degree of the tree, and let H denote the height of the tree. We set D and H, $D = \lceil \log q/\log\log q \rceil$, and $H = \lceil C_H \log q/\log\log q \rceil$.

Definition 1.2.4 (number of repetitions per level)**.** *Let R denote the number of repetitions in each level. Let $C_R > 1$ denote some sufficiently large constant. We set $R = C_R \log\log q$.*

For $\ell \in \{0, 1, \dots, H\}$ we define $\mathcal{I}_\ell$, which is a family of sets. Every set $\mathcal{I}_\ell$ is a decomposition of $[n]$ to τD^ℓ intervals of (roughly) the same length. The set $\mathcal{I}_0$ is a decomposition of $[n]$ to τ intervals of (roughly) the same length length q. If needed, we can round q to a power of 2. This means that

$$\mathcal{I}_0 = \{I_{0,1}, I_{0,2}, \dots\}$$

where

$$I_{0,1} = [1, \tau], I_{0,2} = [\tau + 1, 2\tau], \dots$$

We can conceptualize these sets as a forest consisting of τ trees, each of branching factor D and height H, where the ℓ-th level partitions $[n]$ into disjoint τD^ℓ intervals of length $n/(\tau \cdot D^\ell) = q \cdot D^{-\ell}$. For $\ell \in \{0, \dots, H\}$, interval $I_{\ell,j}$ is decomposed to D disjoint and continuous intervals

$$I_{\ell+1,j\cdot D+1}, \dots, I_{\ell+1,(j+1)\cdot D}$$

of the same length, except possibly the last interval.

We say that an interval $I_{\ell+1,j}$ is a child of interval $I_{\ell,j'}$ if $j' = \lceil j/D \rceil$.

Definition 1.2.5 (sketching matrix Φ)**.** *Let $C_B > 1$ be a sufficiently large constant. Let $B = C_B k/\epsilon$. Let matrices $\Phi^{(1)}, \dots, \Phi^{(H)}$, where every matrix $\Phi^{(\ell)}$ consists of R submatrices $\{\Phi_r^{(\ell)}\}_{r\in[R]}$. For every $\ell \in [H]$ and $r \in [R]$, we pick 2-wise independent hash functions $h_{\ell,r} : [\tau D^\ell] \to [B]$.*

We define measurement $y_{\ell,r,b} = (\Phi_r^{(\ell)} x)_b$ as:

$$y_{\ell,r,b} = \sum_{j\in h_{\ell,r}^{-1}(b)} \sum_{i\in I_{\ell,j}} g_{i,\ell,r} x_j,$$

where $g_{i,\ell,r} \sim \mathcal{N}(0,1)$, i.e. independent standard Gaussians.

We slightly abuse notation and treat y as matrix the mapping to vector should be clear.

Note that C_B should be chosen such that $C_B \gg C_0$, where C_0 appears in tail estimation in Lemma 1.4.4.

1.2.2 Concentration of estimation

In the following lemmata, $\zeta, \eta \in (0,1)$ are absolute constants with $1 > \eta > 1/10 > \zeta$ (See a choice for our application in Table 1.2). The exact values of the constants will be chosen below.

The following lemma handles the probability of detecting a heavy interval at level ℓ.

Lemma 1.2.6 (handling the probability of catching a heavy hitter)**.** *Let $\overline{V}$ be the value in Line 8 of Algorithm 1. Let $V = \epsilon\overline{V}$ be the value in Line 9 of Algorithm 1. Let $j' \in T_{\ell-1}$, and let j be one of its children. Let z_j be defined as follows,*

$$z_j = \operatorname*{median}_{r\in[R]} \left| y_{\ell,r,h_{\ell,r}(j)} \right|^2.$$

Algorithm 1 interval-forest sparse recovery

```
1: procedure IntervalForestSparseRecovery($x, n, k, \epsilon$)    ▷ Theorem 1.2.1
2:     Choose constants $C_H$, $C_R$, $\eta$, $C_0$    ▷ According to Table 1.2
3:     $\tau \leftarrow (k/\epsilon)$    ▷ Definition 1.2.2
4:     $q \leftarrow n/\tau$    ▷ Definition 1.2.2
5:     $H \leftarrow \lceil C_H \log q / \log\log q \rceil$    ▷ Definition 1.2.3
6:     $D \leftarrow \lceil \log q / \log\log q \rceil$    ▷ Definition 1.2.3
7:     $R \leftarrow \lceil C_R \log\log q \rceil$    ▷ Definition 1.2.4
8:     $\overline{V} \leftarrow$ LpLpTailEstimation($x, k, 2, C_0, 1/100$)    ▷ Algorithm 3
9:     $V \leftarrow \epsilon \overline{V}$    ▷ Lemma 1.2.6
10:    $T_0 \leftarrow \{I_{0,1}, \dots, I_{0,\tau}\}$
11:    for $\ell = 1 \to H$ do
12:        $T_\ell \leftarrow$ RecursiveBTree($\ell, R, D, \eta, T_{\ell-1}, V$)    ▷ Lemma 1.2.9
13:    end for
14:    $L \leftarrow T_H$
15:    return $L$    ▷ Lemma 1.2.10
16: end procedure
17: procedure RecursiveBTree($\ell, R, D, \eta, T, V$)
18:    $T' \leftarrow \emptyset$
19:    for $t \in T$ do
20:        Let $I_{\ell,j_1}, I_{\ell,j_2}, \dots, I_{\ell,j_D}$ denote the child intervals of $I_{\ell-1,t}$
21:        for $p \in [D]$ do
22:            $z_{j_p} \leftarrow \mathrm{median}_{r\in[R]} |y_{\ell,r,h_{\ell,r}(j_p)}|^2$    ▷ Definition 1.2.5
23:        end for
24:        if $z_{j_p} \geq \eta V$ then
25:            $T' \leftarrow T' \cup \{j_p\}$
26:        end if
27:    end for
28:    return $T'$
29: end procedure
```

If $\|x_{I_{\ell,j}}\|_2^2 \geq C_j \frac{\epsilon}{k} \|x_{-k}\|_2^2$, where $C_j \geq 2$, then with probability $1 - C_j^{-R/6}$ (over the randomness of $h_{\ell,r}$ and $g_{i,\ell,r}$ for $r \in [R], i \in [n]$ in Definition 1.2.5), we have that

$$z_j \geq \eta V.$$

Proof. Fix $r \in [R]$. Let $b = h_{\ell,r}(j)$, and define $J = \cup_{t \in h_{\ell,r}^{-1}(b)} I_{\ell,t}$. We observe that

$$|y_{\ell,r,b}|^2 = \|x_J\|_2^2 g^2, \text{ where } g \sim \mathcal{N}(0,1).$$

By property of Gaussian distribution, we have

$$\Pr\left[|y_{\ell,r,b}|^2 \le (\eta/C_j)\|x_J\|_2^2\right] \le \frac{2}{\sqrt{2\pi}}\sqrt{\frac{\eta}{C_j}} \le \sqrt{\frac{2\eta}{\pi C_j}},$$

It implies, since $\|x_J\|_2^2 \ge \|x_{I_{\ell,j}}\|_2^2 \ge C_j \frac{\epsilon}{k}\|x_{-k}\|_2^2$, that

$$\Pr\left[|y_{\ell,r,b}|^2 \le \eta\frac{\epsilon}{k}\|x_{-k}\|_2^2\right] \le \sqrt{\frac{2\eta}{\pi C_j}}.$$

Since Lemma 1.4.4, we have $\overline{V} \le \frac{1}{k}\|x_{-k}\|_2^2$. Because $V = \epsilon\overline{V}$,

$$\Pr\left[|y_{\ell,r,b}|^2 \le \eta V\right] \le \sqrt{\frac{2\eta}{\pi C_j}}.$$

The R repetitions ensure that the failure probability can be driven down to $C_j^{-R/6}$, because

$$\begin{aligned}\Pr[z_j \le \eta V] &\le \binom{R}{R/2}\cdot\left(\sqrt{\frac{2\eta}{\pi C_j}}\right)^{R/2} \\ &\le 2^R \cdot (2\eta/\pi)^{R/4} \cdot C_j^{-R/4} \\ &\le (2^R \cdot (2\eta/\pi)^{R/4} \cdot 2^{-R/12}) \cdot C_j^{-R/6} \\ &\le \left(2^{11}\cdot(2/9\pi)^3\right)^{R/12} \cdot C_j^{-R/6} \\ &\le C_j^{-R/6},\end{aligned}$$

where the first step follows from a union bound, the third step follows from $C_j \ge 2$, and the forth step follows from $\eta \le 1/9$.

□

The following lemma handles the probability of a non-heavy interval being considered "heavy" by the algorithm at level ℓ.

Lemma 1.2.7 (handling the probability of false positives). *Let V be the value in Line 9 of Algorithm 1. Let j' be an index in $T_{\ell-1}$, and let j be one of its children. If $\|x_{I_{\ell,j}}\|_2^2 \le \zeta\frac{\epsilon}{k}\|x_{-C_0k}\|_2^2$, then with probability $1 - 2^{-R/3}$ (over the randomness of $h_{\ell,r}$ and $g_{i,\ell,r}$ for $r \in [R], i \in [n]$ in*

Definition 1.2.5) we have that

$$z_j < \eta V.$$

Proof. Fix $\ell \in [H]$ and consider the set H_ℓ that contains the $C_0 k$ coordinates j'' with the largest $\|x_{I_{\ell,j''}}\|_2^2$ values. Define $H_\ell^{(j)} = H_\ell \setminus \{j\}$. Fix $r \in [R]$ and observe that by a union-bound we get that

$$\Pr\left[\exists j'' \in H_\ell^{(j)} \mid h_{\ell,r}(j) = h_{\ell,r}(j'')\right] \leq C_0 k \cdot \frac{1}{C_B k/\epsilon} = \frac{C_0 \epsilon}{C_B} \leq \frac{1}{20},$$

because $C_B \geq 20 C_0$.

We condition on the event $\forall j'' \in H_\ell^{(j)} : h_{\ell,r}(j) \neq h_{\ell,r}(j'')$. A standard calculation now shows that

$$\begin{aligned}
\mathbf{E}\left[|y_{\ell,r,h_{r,\ell}(j)}|^2\right] &\leq \|x_{I_{\ell,j}}\|_2^2 + \frac{1}{C_B}\frac{\epsilon}{k}\|x_{-C_0 k}\|_2^2 \\
&\leq \zeta \frac{\epsilon}{k}\|x_{-C_0 k}\|_2^2 + \frac{1}{C_B}\frac{\epsilon}{k}\|x_{-C_0 k}\|_2^2 \\
&\leq \frac{2\zeta\epsilon}{k}\|x_{-C_0 k}\|_2^2,
\end{aligned}$$

where the last step follows from $\frac{1}{C_B} < \zeta$.

We now apply Markov's inequality to obtain

$$\begin{aligned}
\Pr\left[|y_{\ell,r,h_{\ell,r}(j)}|^2 \geq \eta V\right] &\leq \Pr\left[|y_{\ell,r,h_{\ell,r}(j)}|^2 \geq \frac{\eta\epsilon}{10k}\|x_{-C_0 k}\|_2^2\right] \\
&\leq \frac{\frac{2\zeta\epsilon}{k}\|x_{-C_0 k}\|_2^2}{\frac{\eta\epsilon}{10k}\|x_{-C_0 k}\|_2^2} \\
&= \frac{20\zeta}{\eta} \\
&\leq \frac{1}{20}, && \text{by } \zeta \leq \eta/400.
\end{aligned}$$

By a union bound, the unconditional probability $\Pr\left[|y_{\ell,r,h_{\ell,r}(j)}|^2 \geq \eta V\right] \leq \frac{1}{10}$. Finally, we can upper bound the probability that $z_j = \text{median}_{r\in[R]} |y_{\ell,r,h_{\ell,r}(j)}|^2$ is greater than ηV,

$$\begin{aligned}\Pr[z_j \geq \eta V] &\leq \binom{R}{R/2}(1/10)^{R/2}\\ &< 2^R \cdot 2^{-\frac{3}{2}R}\\ &= 2^{-\frac{R}{2}}\\ &< 2^{-\frac{R}{3}}.\end{aligned}$$

□

1.2.3 Analysis of running time

Lemma 1.2.8 (bounds of D, H with respect to R)**.** *Let D, H as in Definition 1.2.3. It holds that $D \leq 2^{\frac{R}{6}-10}$, and $H \leq 2^{\frac{R}{6}-10}$.*

Proof. Since $H \geq D$, it suffices to prove the claim only for H.

$$H = C_H \frac{\log q}{\log\log q} < C_H \log q = C_H \log(\epsilon n/k) = C_H 2^{\log\log(\epsilon n/k)} \leq 2^{\frac{R}{6}-10},$$

where the third step follows from $q = \epsilon n/k$, and the last step follows from $\log\log(\epsilon n/k) + \log C_H \leq (C_R/6)\log\log(\epsilon n/k) - 10$ and note that $\log\log(\epsilon n/k) \geq 2$ because we assume $k/\epsilon \leq n/16$.

□

Lemma 1.2.9 (running time)**.** *Let R as in Definition 1.2.4 and D, H as in Definition 1.2.3. Let T_H be the set obtained by applying procedure* RecursiveBTree *H times (lines 11-13) of Algorithm 1, and let $C_L > 1$ be some sufficiently large absolute constant. With probability $1 - H \cdot 2^{-R/6+1}$ we have that:*

- $|T_H| \leq C_L \cdot k/\epsilon$,
- *The running time of* BTreeSparseRecovery *is*

$$O\left(\epsilon^{-1}k \cdot \log(\epsilon n/k) \cdot D\right).$$

Proof. Let $C_L = 2(C_0 + 1/\zeta)$.

First, it is easy to see that $|T_0|$ is bounded by

$$|T_0| = \tau = k/\epsilon < C_L k/\epsilon.$$

We claim that if we condition on the event that $|T_{\ell-1}| \le C_L k/\epsilon$, then with probability $1 - 2^{-R/6+1}$, $|T_\ell| \le C_L k/\epsilon$. The proof of both bullets will then follow by a union-bound over all H levels. Indeed, consider the set Q_ℓ containing the $|T_{\ell-1}|D \le C_L \cdot (k/\epsilon) \cdot D$ coordinates j that are children of some $j' \in T_{\ell-1}$. Define

$$B_\ell = \left\{ j \in Q_\ell \;\middle|\; \|x_{I_{\ell,j}}\|_2^2 \le \zeta \frac{\epsilon}{k} \|x_{-C_0 k}\|_2^2 \right\}.$$

By definition of B_ℓ and Q_ℓ, we have

$$|B_\ell| \le |Q_\ell| \le C_L \cdot (k/\epsilon) \cdot D.$$

Moreover, Lemma 1.2.7 gives

$$\forall j \in B_\ell, \Pr\left[z_j \ge \eta V\right] \le 2^{-R/3}$$

Define random variables W_j to be 1 if $z_j \ge \eta V$, and 0 otherwise. Then

$$\mathbf{E}\left[\sum_{j \in B_\ell} W_j\right] \le \frac{C_L k}{\epsilon} D \cdot 2^{-R/3}$$

An application of Markov's inequality gives

$$\Pr\left[\sum_{j \in B_\ell} W_j \ge \frac{C_L k}{2\epsilon} D \cdot 2^{-R/6}\right] \le 2^{-R/6+1}.$$

Conditioning on $\sum_{j \in B_\ell} W_j \le \frac{C_L k}{2\epsilon} D \cdot 2^{-R/6}$ we will upper bound the size of T_ℓ.

First, observe that there exist at most $(C_0 k + k/(\zeta\epsilon))$ $j \in \mathcal{I}_\ell$ for which $\|x_{I_{\ell,j}}\|_2^2 > \zeta \frac{\epsilon}{k} \|x_{-C_0 k}\|_2^2$. This gives

$$|T_\ell| \le \left(C_0 k + \frac{k}{\zeta\epsilon}\right) + \frac{C_L k}{2\epsilon} D 2^{-R/6}. \tag{1.2}$$

If $C_L \geq 2(C_0 + 1/\zeta)$, then we can upper bound the first term in Eq. (1.2),

$$C_0 k + \frac{k}{\zeta\epsilon} \leq \left(C_0 + \frac{1}{\zeta}\right)\frac{k}{\epsilon} \leq \frac{1}{2}\frac{C_L k}{\epsilon}.$$

For the second term in Eq. (1.2), we can show that

$$\frac{C_L k}{2\epsilon} \cdot D 2^{-R/6} \leq \frac{1}{2}\frac{C_L k}{\epsilon},$$

or equivalently $D \leq 2^{R/6}$, which holds by Lemma 1.2.8.

We have D levels, and at each level ℓ we have $|T_\ell| = O(k/\epsilon)$, conditioned on the aforementioned events happening. The children of T_ℓ is then $O(k/\epsilon \cdot D)$. Since we have R repetitions the total running time per level is $O((k/\epsilon) \cdot D \cdot R)$, and the total running time is

$$O((k/\epsilon) \cdot D \cdot R \cdot H) = O\left((k/\epsilon) \cdot D \cdot \log\log q \cdot \frac{\log q}{\log\log q}\right) = O\left((k/\epsilon) \cdot D \cdot \log(\epsilon n/k)\right),$$

where the first step follows from definition of R and H, and the last step follows from definition of q.

Therefore, it gives the desired result. $\square$

1.2.4 Guarantees of the algorithm

Lemma 1.2.10 (guarantees). *Let $L = T_H$ be the set obtained by applying procedure* RecursiveBTree *R times (lines 11-13) of Algorithm 1, we have that, with probability $9/10$, there exist $T' \subseteq L$ of size at most k, such that*

$$\|x - x_{T'}\|_2^2 \leq (1+\epsilon)\|x_{-k}\|_2^2.$$

Proof. Define

$$\mathcal{H} = \left\{ j \in H(x,k) \;\middle|\; \exists C_j \geq 2, |x_j|^2 \geq C_j \frac{\epsilon}{k}\|x_{-k}\|_2^2 \right\}.$$

Moreover, associate every $j \in \mathcal{H}$ with its corresponding $C_j = \frac{|x_j|^2}{\frac{\epsilon}{k}\|x_{-k}\|_2^2}$.

Pick $j \in \mathcal{H}$. Let also $j_1, j_2, \dots, j_H$, be numbers such that

$$j \in I_{1,j_1}, j \in I_{2,j_2}, \dots, j \in I_{R,j_H}.$$

For any $t \in \{1, \dots, H-1\}$, if $I_{t,j_t} \in T_t$, then $I_{t+1,j_{t+1}} \in T_{t+1}$ with probability $1 - C_j^{-R/6}$, by Lemma 1.2.6. Since $C_j \geq 2$, this allows us to take a union bound over all H levels and claim that with probablity $1 - HC_j^{-R/6}$, $j \in T_R$. For $j \in \mathcal{H}$ define random variable to δ_j to be 1 if $j \notin T_H$.

$$\begin{aligned}
\mathbf{E}\left[\delta_j x_j^2\right] &\leq H \cdot C_j^{-R/6} x_j^2 \\
&\leq H \cdot C_j^{-R/6} C_j \frac{\epsilon}{k} \|x_{-k}\|_2^2 \\
&= H \cdot C_j^{-R/6+1} \frac{\epsilon}{k} \|x_{-k}\|_2^2 \\
&\leq \frac{\epsilon}{80k} \|x_{-k}\|_2^2,
\end{aligned}$$

where the first step follows by definition of δ_j, the second step follows by the fact that $j \in \mathcal{H}$, and the last step follows by Lemma 1.2.8. Since $|\mathcal{H}| \leq k$, we have that

$$\mathbf{E}\left[\sum_{j \in \mathcal{H}} \delta_j x_j^2\right] \leq |\mathcal{H}| \cdot \frac{\epsilon}{80k} \|x_{-k}\|_2^2 \leq \frac{\epsilon}{80} \|x_{-k}\|_2^2.$$

Then applying Markov's inequality, we have

$$\Pr\left[\sum_{j \in \mathcal{H}} \delta_j x_j^2 > \frac{\epsilon}{2} \|x_{-k}\|_2^2\right] \leq \frac{1}{40}.$$

We condition on the event $\sum_{j \in \mathcal{H}} \delta_j x_j^2 \leq \frac{\epsilon}{2} \|x_{-k}\|_2^2$. Setting $T' = \mathcal{H} \cap T_H$ we observe that

$$\begin{aligned}
\|x - x_{T'}\|_2^2 &= \sum_{j \in \mathcal{H} \cap H(x,k)} \delta_j x_j^2 + \|x_{H(x,k) \setminus \mathcal{H}}\|_2^2 + \|x_{[n] \setminus H(x,k)}\|_2^2 \\
&\leq \frac{\epsilon}{2} \|x_{-k}\|_2^2 + k \cdot \frac{2\epsilon}{k} \|x_{-k}\|_2^2 + \|x_{-k}\|_2^2 \\
&\leq (1 + 3\epsilon) \|x_{-k}\|_2^2.
\end{aligned}$$

where the second step follows by the bound on $\sum_{j \in T} \delta_j x_j^2$ and the fact that every $j \notin T$ satisfies $|x_j|^2 \leq (2\epsilon/k) \|x_{-k}\|_2^2$.

Rescaling for ϵ we get the desired result. $\square$

1.2.5 Bounding the number of measurements

In this subsection, we prove that

Claim 1.2.11 (#measurements)**.** *The number of measurements is $O(\epsilon^{-1}k \cdot \log(\epsilon n/k))$.*

Proof. Recall the definition of τ and q,

$$\tau = k/\epsilon, \quad q = n/\tau.$$

We thus have the following bound on the number of measurements:

$$B \cdot H \cdot R = C_B(k/\epsilon) \cdot C_H \frac{\log(\epsilon n/k)}{\log\log(\epsilon n/k)} \cdot C_R \log\log(\epsilon n/k) = O\left((k/\epsilon)\log(\epsilon n/k)\right).$$

$\square$

1.3 The Pruning Linear Sketch

The goal of this section is to prove Theorem 1.3.1.

Theorem 1.3.1 (Restatement of Lemma 1.1.7)**.** *Let $C_L, \alpha, \beta > 1$ be three fixed constants. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with $m = O((k/\epsilon) \cdot \log(1/\epsilon))$, with column sparsity $O(\log(1/\epsilon))$ such that the following holds :*
Suppose that one is given a set $L \subseteq [n]$ such that

$$|L| = C_L \cdot k/\epsilon, \qquad \exists T \subset L, |T| \le k : \|x - x_T\|_2 \le (1+\epsilon)\|x_{-k}\|_2.$$

Then procedure Prune *(Algorithm 2) can find a set S of size $\beta \cdot k$ in time $O(m)$, such that*

$$\|x - x_S\|_2 \le (1 + \alpha \cdot \epsilon)\|x_{-k}\|_2$$

holds with probability $9/10$.

In Section 1.3.1, we provide some basic definitions and description of our algorithm. We analyze the coordinates from several perspectives in Section 1.3.2. We prove the correctness of our algorithm in Section 1.3.3 and analyze time, number of measurements column sparsity, success probability of algorithm in Section 1.3.4.

1.3.1 Design of the sketching matrix, and helpful definitions

Notation	Choice	Statement	Parameter
C_R	$10^4 + 500C_L$	Definition 1.3.2	R
C_B	5×10^5	Definition 1.3.2	B
C_g	$4/5$	Fact 1.3.3	Gaussian variable
C_L	10^4	Theorem 1.3.1	L
α	5	Theorem 1.3.1	Blow up on ϵ
β	100	Theorem 1.3.1	Blow up on k

Table 1.3: *Summary of constants in Section 1.3, the column "Parameter" indicates which parameter is depending on that constant. Note that set L is the input of the algorithm in Section 1.3 and the output of the algorithm in Section 1.2.*

Definition 1.3.2 (sketching matrix Φ)**.** *Let $C_R, C_B > 1$ be absolute constants. Let $R = C_R \log(1/\epsilon)$. Let $B = C_B k/\epsilon$. For $r \in [R]$, we pick 2-wise independent hash function $h_r : [n] \to [B]$, as well as normal random variables $\{g_{i,r}\}_{i\in[n],r\in[R]}$ and take measurements*

$$y_{r,b} = \sum_{i \in h_r^{-1}(b)} x_i g_{i,r}.$$

Given the set L, for every $i \in L$ we calculate

$$z_i = \operatorname*{median}_{r\in[R]} |y_{r,h_r(i)}|,$$

and keep the indices i with the βk largest z_i values to form a set S of indices, for some absolute constant β sufficiently large. We describe this pruning step in Algorithm 2. For the analysis, we define the threshold

$$\tau = \|x_{-k}\|_2/\sqrt{k}. \tag{1.3}$$

We will need the following standard fact about the Gaussian distribution. Then we proceed with a series of definitions and lemmata.

Algorithm 2 the prune procedure

```
1: procedure PRUNE($x, n, k, \epsilon, L$)                ▷ Theorem 1.3.1
2:     $R \leftarrow C_R \log(1/\epsilon)$
3:     $B \leftarrow C_B k/\epsilon$
4:     for $r = 1 \to R$ do
5:         Sample $h_r : [n] \to [B] \sim$ 2-wise independent family
6:         for $i = 1 \to n$ do
7:             Sample $g_{i,r} \sim \mathcal{N}(0,1)$
8:         end for
9:     end for
10:    for $r = 1 \to R$ do
11:        for $b = 1 \to B$ do
12:            $y_{r,b} \leftarrow \sum_{i \in h_r^{-1}(b)} x_i g_{i,r}$
13:        end for
14:    end for
15:    for $i \in L$ do
16:        $z_i \leftarrow \text{median}_{r \in [R]} |y_{r,h_r(i)}|$
17:    end for
18:    $S \leftarrow \{i \in L : z_i$ is in the top $\beta k$ largest coordinates in vector $z\}$
19:    return $S$
20: end procedure
```

Fact 1.3.3 (property of Gaussian). *Suppose $x \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable. For any $t \in (0, \sigma]$ we have*

$$\Pr[x \geq t] \in \left[\frac{1}{2}(1 - \frac{4}{5}\frac{t}{\sigma}), \frac{1}{2}(1 - \frac{2}{3}\frac{t}{\sigma})\right].$$

Similarly, if $x \sim \mathcal{N}(\mu, \sigma^2)$, for any $t \in (0, \sigma]$, we have

$$\Pr[|x| \geq t] \in \left[1 - \frac{4}{5}\frac{t}{\sigma}, 1 - \frac{2}{3}\frac{t}{\sigma}\right].$$

The form we will need is the following:

$$\Pr_{g \sim \mathcal{N}(0,1)}[|g| \leq t] \leq \frac{4}{5}t.$$

Thought the analysis, for convenience we will set $C_g = 4/5$. Another form we will need is:

$$\Pr_{g \sim \mathcal{N}(0,1)}\left[|g| \in \left[\frac{1}{3C_g}, 2\right]\right] \geq 0.63$$

Proof. The first form is true by simple calculation. The second form is holding due to numerical values of cdf for normal distribution,

$$\Pr\left[|g_{i,r}| \in \left[\frac{1}{3C_g}, 2\right]\right] = 2(f(2) - f(1/3C_g)) = 2(f(2) - f(5/12)) \geq 2(0.977 - 0.662) = 0.63,$$

where $f(x) = \int_{-\infty}^{x} \frac{e^{-x^2/2}}{\sqrt{2\pi}} dx$ is the cdf of normal distribution. $\square$

Stochastic dominance is a partial order between random variables and it is a classical concept in decision theory and decision analysis [HR69, Baw75]. We give the simplest definition below and it is sufficient for our application.

Definition 1.3.4 (stochastic domination of Gaussian random variables). *Let $\sigma_1 < \sigma_2$ and random variables $X \sim \mathcal{N}(0, \sigma_1^2), Y \sim \mathcal{N}(0, \sigma_2^2)$. Then we say that $|Y|$ stochastically dominates $|X|$, and it holds that*

$$\Pr\left[|Y| \geq \lambda\right] \geq \Pr\left[|X| \geq \lambda\right], \quad \forall \lambda \geq 0.$$

We formally define the set L as follows:

Definition 1.3.5 (set L, input of the algorithm). *Let $C_L > 1$ be a fixed constant, and let set $L \subseteq [n]$ be defined as:*

$$|L| = C_L \cdot k/\epsilon, \qquad \exists T \subset |T| \leq k : \|x - x_T\|_2 \leq (1+\epsilon)\|x_{-k}\|_2.$$

We provide a definition called "badly-estimated coordinate",

Definition 1.3.6 (badly-estimated coordinate). *We will say a coordinate $i \in [n]$ is badly-estimated if*

$$z_i \notin \left[\frac{1}{3C_g}|x_i| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2, 2|x_i| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2\right],$$

Then, we can define "badly-estimated set",

Definition 1.3.7 (badly-estimated set $\mathcal{B}$). *Let set L be defined as Definition 1.3.5. We say $\mathcal{B} \subseteq L$ is a badly-estimated set if for all $i \in \mathcal{B}$, z_i is a badly estimated coordinate (see Definition 1.3.6).*

We define a set of large coordinates in head,

Definition 1.3.8 (large coordinates in head)**.** *Let τ be defined in* (1.3)*. Let L be defined in Definition 1.3.5. Let C_g be the constant from Fact 1.3.3. Define set*

$$\mathcal{M} = \{i \in L \cap H(x,k) : |x_i| \geq 3C_g \tau\},$$

which contains the head coordinates of x that are in L and are larger in magnitude than $3C_g\tau$.

1.3.2 Analyzing head and badly-estimated coordinates

Lemma 1.3.9 (expected error from coordinates above τ)**.** *We have that*

$$\mathbf{E}\left[\sum_{i \in \mathcal{M}} x_i^2 \cdot 1_{z_i < \tau}\right] \leq \frac{\epsilon}{100}\|x_{-k}\|_2^2.$$

Proof. Fix $i \in \mathcal{M}$. Observe that for $r \in [R]$

$$|y'_{r,h_r(i)}| \sim \|x_{h_r^{-1}(i)}\|_2 |\mathcal{N}(0,1)|.$$

Since

$$\|x_{h_r^{-1}(i)}\|_2 \geq |x_i|$$

we have that the random variable $|y'_{r,h_r(i)}|$ stochastically dominates the random variable $|x_i| \cdot |\mathcal{N}(0,1)|$.

By Fact 1.3.3, we have that

$$\Pr\left[|y'_{r,h_r(i)}| \leq \tau\right] \leq C_g \frac{\tau}{|x_i|}.$$

Because of the $R = C_R \log(1/\epsilon)$ repetitions, a standard argument gives that

$$\Pr\left[1_{z_i<\tau} = 1\right] \leq \left(C_g \frac{\tau}{|x_i|}\right)^{C' \log(1/\epsilon)},$$

for some absolute constant $C' > C_R/3$.

We now bound

$$\mathbf{E}\left[\sum_{i\in\mathcal{M}} x_i^2 \cdot 1_{z_i<\tau}\right] \leq \sum_{i\in\mathcal{M}} x_i^2 \left(C_g \frac{\tau}{|x_i|}\right)^{C'\log(1/\epsilon)}$$

$$\begin{aligned} &= \sum_{i\in\mathcal{M}} C_g^2\tau^2 \left(C_g \frac{\tau}{|x_i|}\right)^{C'\log(1/\epsilon)-2} \\ &\leq k \cdot \tau^2 \cdot \frac{\epsilon}{100} \\ &= \frac{\epsilon}{100}\|x_{-k}\|_2^2, \end{aligned}$$

where the first step follows by the bound on $\mathbf{E}\left[1_{z_i<\tau}\right] = \Pr\left[1_{z_i<\tau} = 1\right]$, and the third step by choosing by choosing $C_R > 1$ to be some sufficiently large constant and the facts $C' > C_R/3$ and $(C_g\tau)/|x_i| \leq 1/3$.

$\square$

Lemma 1.3.10 (probability of a fixed coordinate is badly-estimated). *A coordinate i is badly-estimated (as in Definition 1.3.6) probability at most*

$$\frac{\epsilon^3}{100^2 C_L}.$$

Proof. Fix r and set set $b = h_r(i)$. Recall the definition of $y_{r,b} = \sum_{i\in h_r^{-1}(b)} x_i g_{i,r}$ in Definition 1.3.2. We have that

$$\left| |g_{i,r}x_i| - \left| \sum_{j\in h_r^{-1}(b)\setminus\{i\}} g_{j,r}x_j \right| \right| \leq |y_{r,b}| \leq |g_{i,r}x_i| + \left| \sum_{j\in h_r^{-1}(b)\setminus\{i\}} g_{j,r}x_j \right|,$$

Now, $|g_{i,r}x_i|$ will be at in $[(1/3C_g)|x_i|, 2|x_i|]$ with probability at least 0.63 (due to Fact 1.3.3).

Moreover, for any $j \in H(x,k) \setminus \{i\}$, $h_r(j) \neq b$ with probability $1 - 1/B = 1 - \epsilon/(C_B k) \geq 1 - 1/(C_B k)$. By a union bound, we get with probability at least $1 - 1/C_B$, for all $j \in H(x,k) \setminus \{i\}$, $h_r(j) \neq b$. Conditioning on this event, we have,

$$\mathbf{E}\left[\left(\sum_{j\in h_r^{-1}(b)\backslash\{i\}} g_{j,r}x_j\right)^2\right] = \frac{\epsilon}{C_B k}\|x_{-k}\|_2^2.$$

We then apply Markov's inequality to get that with probability at least $1-10^4/C_B$,

$$\left(\sum_{j\in h_r^{-1}(b)\backslash\{i\}} g_{j,r}x_j\right)^2 \leq \frac{\epsilon}{10^4 k}\|x_{-k}\|_2^2.$$

Therefore, by a union bound,

$$\begin{aligned}\Pr\left[|y_{r,b}| \in \left[\frac{1}{3C_g}|x_i| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}, 2|x_i| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\right]\right] &\geq 0.63 - \frac{1}{C_B} - \frac{10^4}{C_B} \\ &\geq 0.6,\end{aligned}$$

where the last step follows by $C_B \geq 5\times 10^5$.

Note that z_i is obtained by taking median of R copies of i.i.d. $|y_{r,b}|$. For each $r\in[R]$, we define $Z_r = 1$ if $|y_{r,b}|$ falls into that region, and 0 otherwise. We have $\mathbf{E}[\sum_{r=1}^R Z_r] \geq 0.6R$. Using Chernoff bound, we have

$$\begin{aligned}\Pr\left[\sum_{r=1}^R Z_r < 0.9\cdot 0.6R\right] &\leq \Pr\left[\sum_{r=1}^R Z_r < 0.9\,\mathbf{E}[\sum_{r=1}^R Z_r]\right] \\ &\leq e^{-\frac{1}{3}0.1^2\,\mathbf{E}[\sum_{r=1}^R Z_r]} \\ &\leq e^{-\frac{1}{3}0.1^2\cdot 0.6R}\end{aligned}$$

Thus,

$$\begin{aligned}\Pr\left[z_i \notin \left[\frac{1}{3C_g}|x_i| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}, 2|x_i| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\right]\right] &\leq e^{-\frac{1}{3}0.1^2\cdot 0.6R} \\ &\leq 2^{-0.002R} \\ &= 2^{-0.002C_R\log(1/\epsilon)} \\ &\leq 2^{-0.002(10000+500C_L)\log(1/\epsilon)} \\ &\leq \frac{\epsilon^3}{100^2 C_L},\end{aligned}$$

where the third step follows from choice of C_R.

□

1.3.3 Guarantee of algorithm

We now proceed with the proof of Theorem 1.3.1.

Proof. By Lemma 1.3.9 and an application of Markov's inequality we have that

$$\sum_{i \in \mathcal{M}} x_i^2 \cdot 1_{z_i < \tau} \leq \epsilon \|x_{-k}\|_2^2,$$

with probability 99/100. Let this event be $\mathcal{E}_1$.

Moreover, by Lemma 1.3.10,

$$\mathbf{E}[|\mathcal{B}|] \leq \frac{\epsilon^3 |L|}{100^2 C_L},$$

so, by Markov's inequality, we have

$$\Pr\left[|\mathcal{B}| \leq \frac{\epsilon^2 |L|}{100 C_L}\right] \geq 1 - \epsilon/100 \geq 99/100$$

Let this event be $\mathcal{E}_2$.

By taking a union bound, $\mathcal{E}_1$ and $\mathcal{E}_2$ both hold with probability 98/100. Plugging size of $|L|$ ($\leq C_L \cdot k/\epsilon$) into equation of event $\mathcal{E}_2$, we get

$$|\mathcal{B}| \leq \frac{\epsilon^2 |L|}{100 C_L} \leq \frac{\epsilon k}{100}. \tag{1.4}$$

It means there are at most $\epsilon k/100$ coordinates that badly-estimated.

We remind that our goal is to bound

$$\begin{aligned} \|x - x_S\|_2^2 &= \|x_{\overline{S}}\|_2^2 \\ &= \|x_{\overline{S} \cap \mathcal{M}}\|_2^2 + \|x_{\overline{S} \setminus \mathcal{M}}\|_2^2 \\ &= \|x_{\overline{S} \cap \mathcal{M}}\|_2^2 + \|x_{(\overline{S} \setminus \mathcal{M}) \cap \mathcal{B}}\|_2^2 + \|x_{(\overline{S} \setminus \mathcal{M}) \setminus \mathcal{B}}\|_2^2 \end{aligned}$$

1. Bounding $\|x_{\mathcal{M}\cap\overline{S}}\|_2^2$. Consider the set

$$I = \{i \in L \setminus \mathcal{M} : |x_i| \geq \tau/3\},$$

which contains the coordinates in L with magnitude in the range $[\frac{1}{3}\tau, 3C_g\tau)$. By the definition of τ, clearly, $|I| \leq 3k + k = 4k$, because we can have at most k such elements in $H(x,k)$, and at most $3k$ such elements in the tail $[n] \setminus H(x,k)$. Since the number of badly estimated coordinates is at most $\epsilon k/100$ and the size of S is βk for sufficiently large β, we can have at most $4k + \epsilon k/100 < \beta k$ coordinates $i \in L$ which are not in $\mathcal{M}$ and are larger than τ. This means that all coordinates in $\mathcal{M}$ with estimate $z_i \geq \tau$ will belong to S. This implies that

$$\mathcal{M} \cap \overline{S} = \{i \in \mathcal{M} : z_i < \tau\},$$

and hence

$$\|x_{\mathcal{M}\cap\overline{S}}\|_2^2 = \sum_{i\in\mathcal{M}} x_i^2 \cdot 1_{z_i<\tau} \leq \epsilon\|x_{-k}\|_2^2,$$

since we conditioned on event $\mathcal{E}_1$.

2. Bounding $\|x_{(\overline{S}\setminus\mathcal{M})\cap\mathcal{B}}\|_2^2$. For every $i \in (\overline{S} \setminus \mathcal{M}) \cap \mathcal{B})$ we have the trivial bound $|x_i| \leq \tau$. Since $(\overline{S} \setminus \mathcal{M}) \cap \mathcal{B}) \subseteq \mathcal{B}$, because the event $\mathcal{E}_2$ we get that

$$\|x_{(\overline{S}\setminus\mathcal{M})\cap\mathcal{B}}\|_2^2 \leq |\mathcal{B}| \cdot \tau^2 \leq \frac{\epsilon k}{100} \cdot \frac{\|x_{-k}\|_2^2}{k} = \frac{\epsilon}{100}\|x_{-k}\|_2^2,$$

where the second step follows from (1.4) and (1.3).

3. Bounding $\|x_{(\overline{S}\setminus\mathcal{M})\setminus\mathcal{B}}\|_2^2$. Observe that set $(\overline{S} \setminus \mathcal{M}) \setminus \mathcal{B}$ consists of well-estimated coordinates that are less than τ in magnitude, and their estimates do not belong to the largest βk estimates. For convenience, set $Q = (\overline{S} \setminus \mathcal{M}) \setminus \mathcal{B}$, then it is obvious that $Q = \overline{S}\setminus(\mathcal{M} \cup \mathcal{B})$.

We define three sets H_1, H_2, H_3 as follows,

$$H_1 = Q \cap T, \quad H_2 = Q \cap \overline{T}, \quad \text{and} \quad H_3 = (\overline{T}\setminus(\mathcal{M} \cup \mathcal{B}))\setminus\overline{S}.$$

Using the definition of Q, we can rewrite H_1, H_2, and H_3 as follows

$$\begin{aligned}
H_1 &= (\overline{S}\backslash(\mathcal{M}\cup\mathcal{B}))\cap T = \overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap T,\\
H_2 &= (\overline{S}\backslash(\mathcal{M}\cup\mathcal{B}))\cap \overline{T} = \overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap \overline{T},\\
H_3 &= (\overline{T}\backslash(\mathcal{M}\cup\mathcal{B}))\backslash \overline{S} = S\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap \overline{T}.
\end{aligned}$$

We can show that

$$\begin{aligned}
H_2\cap H_3 &= (\overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\cup(S\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\\
&= \emptyset, \qquad (1.5)
\end{aligned}$$

and

$$\begin{aligned}
H_2\cup H_3 &= (\overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\cap(S\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\\
&= \overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T}\\
&= \overline{T}\backslash(\mathcal{M}\cup\mathcal{B}). \qquad (1.6)
\end{aligned}$$

Then,

$$\begin{aligned}
\|x_Q\|_2^2 &= \|x_{H_1}\|_2^2 + \|x_{H_2}\|_2^2\\
&= \|x_{H_1}\|_2^2 + (\|x_{\overline{T}\backslash(\mathcal{M}\cup\mathcal{B})}\|_2^2 - \|x_{H_3}\|_2^2)\\
&\le \|x_{H_1}\|_2^2 + \|x_{\overline{T}}\|_2^2 - \|x_{H_3}\|_2^2\\
&\le \|x_{H_1}\|_2^2 + (1+\epsilon)\|x_{-k}\|_2^2 - \|x_{H_3}\|_2^2,
\end{aligned}$$

where first step follows from $H_1\cap H_2 = \emptyset$ and $H_1\cup H_2 = Q$, the second step follows from Eq. (1.5) and (1.6), the third step follows from $\|x_{\overline{T}\backslash(\mathcal{M}\cup\mathcal{B})}\|_2^2 \le \|x_{\overline{T}}\|_2^2$ and the last step follows from $\|x_{\overline{T}}\|_2^2 \le (1+\epsilon)\|x_{-k}\|_2^2$.

We define d, E, a, b as follows

$$d = |H_1|, \qquad E = \frac{1}{4}\sqrt{\epsilon/k}\|x_{-k}\|_2, \qquad a = \max_{i\in H_1}|x_i|, \qquad b = \min_{i\in H_3}|x_i|. \qquad (1.7)$$

Let i^* and j^* be defined as follows:

$$i^* = \arg\max_{i \in H_1} |x_i|, \quad \text{and} \quad j^* = \arg\min_{j \in H_3} |x_j|. \tag{1.8}$$

Recall the definitions of H_1 and H_3, we know H_3 is a subset of S and H_1 is a subset of $\overline{S}$. Since the set S contains the largest βk coordinates, thus we have

$$z_j \geq z_i, \forall i \in H_1, j \in H_3.$$

It further implies $z_{j^*} \geq z_{i^*}$.

By Definition 1.3.6, we have

$$z_{i^*} \geq \frac{1}{3C_g}|x_{i^*}| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2, \tag{1.9}$$

and

$$z_{j^*} \leq 2|x_{j^*}| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2. \tag{1.10}$$

Then, we can show that $a \leq 6C_g b + E$ in the following sense:

$$\begin{aligned}
a &= |x_{i^*}| && \text{by def. of } i^*, a, (1.8), (1.7) \\
&\leq 3C_g z_{i^*} + \frac{3C_g}{100}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by } (1.9) \\
&\leq 3C_g z_{j^*} + \frac{3C_g}{200}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by } z_{i^*} \leq z_{j^*} \\
&\leq 6C_g |x_{j^*}| + \frac{6C_g}{200}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by } (1.10) \\
&= 6C_g b + \frac{6C_g}{200}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by def. of } j^*, b, (1.8), (1.7) \\
&\leq 6C_g b + E && \text{by def. of } E, (1.7)
\end{aligned}$$

Note that $H_3 = (\overline{T} \setminus (\mathcal{M} \cup \mathcal{B})) \setminus \overline{S} = S \setminus (T \cup \mathcal{M} \cup \mathcal{B})$. Therefore,

$$\begin{aligned}
|H_3| &\geq |S| - |T| - |\mathcal{M}| - |\mathcal{B}| \\
&\geq \beta k - k - k - k \\
&= (\beta - 3)k.
\end{aligned}$$

Finally, we can have

$$\begin{aligned}
\|x_{H_1}\|_2^2 - \|x_{H_3}\|_2^2 &\leq da^2 - (\beta - 3)kb^2 \\
&\leq d(6C_g b + E)^2 - (\beta - 3)kb^2 && \text{by } a \leq 6C_g b + E \\
&= (36C_g^2 d - (\beta - 3)k)b^2 + 12C_g bdE + dE^2 \\
&\leq (36C_g^2 k - (\beta - 3)k)b^2 + 12C_g bkE + kE^2 && \text{by } d \leq k \\
&\leq (36C_g^2 k - (\beta - 5C_g^2)k)b^2 + 12C_g bkE + kE^2 && \text{by } C_g \geq 4/5 \\
&\leq -36kC_g^2 b^2 + 12C_g bkE + kE^2 && \text{by } \beta \geq 77C_g^2 \\
&= -k(6C_g b - E)^2 + 2kE^2 \\
&\leq 2kE^2 \\
&\leq \epsilon \|x_{-k}\|_2^2.
\end{aligned}$$

where the last step follows from definition of E.

Thus, we have

$$\|x_Q\|_2^2 \leq (1 + 2\epsilon)\|x_{-k}\|_2^2.$$

Putting it all together. We have

$$\begin{aligned}
\|x - x_S\|_2^2 &= \|x_{\overline{S} \cap \mathcal{M}}\|_2^2 + \|x_{(\overline{S} \setminus \mathcal{M}) \cap \mathcal{B}}\|_2^2 + \|x_{(\overline{S} \setminus \mathcal{M}) \setminus \mathcal{B}}\|_2^2 \\
&\leq \epsilon \|x_{-k}\|_2^2 + \frac{\epsilon}{100}\|x_{-k}\|_2^2 + (1 + 2\epsilon)\|x_{-k}\|_2^2 \\
&\leq (1 + 4\epsilon)\|x_{-k}\|_2^2
\end{aligned}$$

Finally, we can conclude $\alpha = 5$ and $\beta = 100$.

□

1.3.4 Time, measurements, column sparsity, and probability

In this section, we will bound the decoding time, the number of measurements, column sparsity and success probability of algorithm.

Decoding time. For each $i \in L$, we compute z_i to be the median of R values. For this part, we spend $O(|L| \cdot R) = O((k/\epsilon) \cdot \log(1/\epsilon))$ time. Moreover, calculating the top βk estimates in L only takes $O(|L|)$ time. Therefore, the decoding time is $O((k/\epsilon) \cdot \log(1/\epsilon))$.

The number of measurements. The number of measurements is the bucket size B times the number of repetitions R, which is $O(BR) = O((k/\epsilon) \cdot \log(1/\epsilon))$.

Column sparsity. Each $i \in [n]$ goes to one bucket for each hash function, and we repeat R times, so the column sparsity is $O(R) = O(\log(1/\epsilon))$.

Success probability. By analysis in Section 1.3.3, the success probability is at least 0.98.

1.4 Tail Estimation

In Section 1.4.1, we present a standard result on random walks. In Section 1.4.2, we present some results on p-stable distribution. In what follows we asssume that $0 < p \leq 2$. We show an algorithm for ℓ_p tail estimation in Section 1.4.3.

1.4.1 Random walks

Theorem 1.4.1. *We consider the following random walk. We go right if $B_i = 1$ and we go left if $B_i = 0$. The probability of $B_i = 1$ is at least $9/10$ and the probability of $B_i = 0$ is at most $1/10$. With at least some constant probability bounded away from $\frac{1}{2}$, for all the possible length of the random walk, it will never return to the origin.*

This is a standard claim, that can be proved in numerous ways, such as martingales etc. For the completeness, we still provide a folklore proof here.

Proof. Let $p > 1/2$ be the probability of stepping to the right, and let $q = 1 - p$. For integer $m \geq 1$, let P_m be the probability of first hitting 0 in exactly m steps. It is obvious that $P_m = 0$ if n is even, and $P_1 = q$. In order to hit 0 for the first time on the third step you must Right-Left-Left, so $P_3 = pq^2$. To hit 0 for the first time in exactly $2k + 1$ steps, you must go right k times and left $k+1$ times, your last step must be to the left, and through the first $2k$ steps you must always have made at least many right steps as left steps. It is well known that the number o such path is C_k, which is the k-th Catalan number. Thus,

$$P_{2k+1} = C_k q^k q^{k+1} = C_k \cdot q(pq)^k = \frac{q(pq)^k}{k+1}\binom{2k}{k},$$

since

$$C_k = \frac{1}{k+1}\binom{2k}{k}$$

By [Wil05], the generating function for the Catalan numbers is

$$c(x) = \sum_{k\geq 0} C_k x^k = \frac{1-\sqrt{1-4x}}{2x},$$

so the probability that the random walk will hit 0 is

$$\begin{aligned}
\sum_{k\geq 0} P_{2k+1} &= q\sum_{k\geq 0} C_k (pq)^k \\
&= q\cdot c(pq) \\
&= q\cdot \frac{1-\sqrt{1-4pq}}{2pq} && \text{by definition of } c(x) \\
&= \frac{1-\sqrt{1-4q(1-q)}}{2p} \\
&= \frac{1-\sqrt{1-4q+4q^2}}{2p} \\
&= \frac{1-(1-2q)}{2p} \\
&= q/p \\
&\leq 1/9.
\end{aligned}$$

Thus, we complete the proof. □

1.4.2 p-stable distributions

We first provide the definition of p-stable distribution. For the more details, we refer the readers to [Ind06].

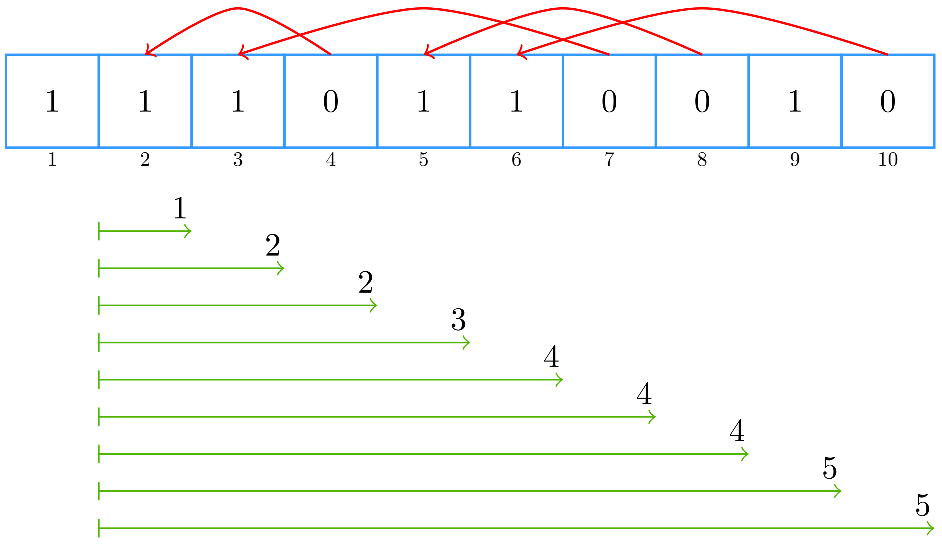


Figure 1.1: *This is a visualization of part of the proof in Claim 1.4.8. We consider an example where there are $l = 10$ blocks, $B_1 = 1$, $B_2 = 1$, $B_1 = 1$, $B_3 = 0$, $B_4 = 0$, $B_5 = 1$, $B_6 = 1$, $B_7 = 0$, $B_8 = 0$, $B_9 = 1$ and $B_{10} = 0$. Recall the two important conditions in the proof of Claim 1.4.8, the first one is $B_1 = 1$ and the second one is, for all $j \in [l], \sum_{j'=2}^{j} B_{j'} > (j-1)/2$. The number on the green arrow is $\sum_{j'=2}^{j} B_{j'}$. It is to see that the example we provided here is satisfying those two conditions. Recall the definition of set S_1 and S_0. Here $S_1 = \{2,3,5,6,9\}$ and $S_0 = \{4,7,8,10\}$. Then $S_1' = \{2,3,5,6\}$. The mapping π satisfies that $\pi(4) = 2$, $\pi(7) = 3$, $\pi(8) = 5$ and $\pi(10) = 6$.*

Definition 1.4.2 (p-stable distribution)**.** *A distribution $\mathcal{D}$ over $\mathbb{R}$ is called p-stable, if there exists $p \geq 0$ such that for any n real numbers $a_1, a_2, \cdots, a_n$ and i.i.d. variables $x_1, x_2, \cdots, x_n$ from distribution $\mathcal{D}$, the random variable $\sum_{i=1}^{n} a_i x_i$ has the same distribution as the variable $\|a\|_p y$, where y is a random variable from distribution $\mathcal{D}$.*

Theorem 1.4.3 ([Zol86])**.** *For any $p \in (0,2]$, there exists a p-stable distribution.*

Gaussian distribution defined by the density function $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$, is 2-stable. Cauchy distribution defined by density function $f(x) = \frac{1}{\pi}\frac{1}{1+x^2}$ is 1-stable. Let $\mathcal{D}_p$ denote the p-stable distribution. For $p = 2$, $\mathcal{D}_p$ is $\mathcal{N}(0,1)$ and for $p = 1$, $\mathcal{D}_p$ is $\mathcal{C}(0,1)$.

1.4.3 ℓ_p-tail estimation algorithm

The goal of this Section is prove Lemma 1.4.4.

One can try to prove such a claim for $p = 2$ with random signs, instead of Gaussians, by applying the Paley-Zygmund inequality to obtain the lower bound. A straightforward

Algorithm 3 ℓ_p tail estimation algorithm

```
1: procedure LpLpTailEstimation($x, k, p, C_0, \delta$)        ▷ Lemma 1.4.4
2:                                                ▷ Requires $C_0 \geq 1000$
3:     $m \leftarrow O(\log(1/\delta))$
4:     Choose $g_{i,t}$ to be random variable that sampled i.i.d. from distribution $\mathcal{D}_p$, $\forall i, t \in [n] \times [m]$
5:     Choose $\delta_{i,t}$ to be Bernoulli random variable with $\mathbf{E}[\delta_{i,t}] = 1/(100k)$, $\forall i, t \in [n] \times [m]$
6:                  ▷ Matrix $A$ is implicitly constructed based on $g_{i,t}$ and $\delta_{i,t}$
7:     for $t \in [m]$ do
8:         $y_t \leftarrow \sum_{i=1}^n \delta_{i,t} \cdot g_{i,t} \cdot x_i$
9:     end for
10:    $V \leftarrow \mathrm{median}_{t \in [m]} |y_t|^2$
11:    return $V$                ▷ $\frac{1}{10k}\|x_{-C_0 k}\|_p^p \leq V \leq \frac{1}{k}\|x_{-k}\|_p^p$
12: end procedure
```

calculation indicates that this approach does not give the desired result, hence we need a new argument to deal with the lower bound.

Lemma 1.4.4 (Restatement of Lemma 1.1.5)**.** *Let $C_0 \geq 1000$ denote some fixed constant. There is an oblivious construction of matrix $A \in \mathbb{R}^{m \times n}$ with $m = O(\log(1/\delta))$ along with a decoding procedure* LpLpTailEstimation(x, k, p, C_0, δ) *(Algorithm 3) such that, given Ax, it is possible to output a value V in time $O(m)$ such that*

$$\frac{1}{10k}\|x_{-C_0 k}\|_p^p \leq V \leq \frac{1}{k}\|x_{-k}\|_p^p,$$

holds with probability $1 - \delta$.

Proof. Let $m = O(\log(1/\delta))$. For each $i \in [n], t \in [m]$, we use $g_{i,t}$ to denote a random variable that sample from distribution $\mathcal{D}_p$.

For each $i \in [n], t \in [m]$, we use $\delta_{i,t}$ to denote a Bernoulli random variable such that

$$\delta_{i,t} = \begin{cases} 1, & \text{with prob. } \frac{1}{100k}; \\ 0, & \text{otherwise.} \end{cases}$$

Then we have

$$\mathbf{E}[\delta_{i,t}] = \frac{1}{100k}.$$

For each $t \in [m]$, we define y_t as follows

$$y_t = \sum_{i=1}^{n} \delta_{i,t} g_{i,t} x_i. \tag{1.11}$$

For each $t \in [m]$, we define Δ_t as follows

$$\Delta_t = \left(\sum_{i=1}^{n} \delta_{i,t}^p x_i^p \right)^{1/p}. \tag{1.12}$$

Using Claim 1.4.5 and Claim 1.4.8

$$\Pr_{g,\delta}\left[|y_t| < \alpha \frac{1}{(2C_0 k)^{1/p}} \|x_{-C_0 k}\|_p \right] \leq 1/5.$$

Using Claim 1.4.6 and Claim 1.4.7

$$\Pr_{g,\delta}\left[|y_t| > \beta \frac{1}{k^{1/p}} \|x_{-k}\|_p \right] \leq 1/5.$$

Finally, we just take the median over m different independent repeats. Since $m = O(\log(1/\delta))$, thus, we can boost the failure probability to δ.

□

It is a standard fact, due to p-stability, that y_t follows the p-stable distribution : $\Delta_t \cdot \mathcal{D}_p$. Since p-stable distributions are continuous functions, we have the following two Claims:

Claim 1.4.5 (upper bound on $|y_t|$)**.** *Let y_t be defined in Eq. (1.11), let Δ_t be defined in Eq. (1.12). There is some sufficiently small constant $\alpha \in (0,1)$ such that*

$$\Pr_{g}[|y_t| < \alpha \cdot \Delta_t] \leq 1/10.$$

Claim 1.4.6 (lower bound on $|y_t|$)**.** *Let y_t be defined in Eq. (1.11), let Δ_t be defined in Eq. (1.12). There is some sufficiently large constant $\beta > 1$ such that*

$$\Pr_{g}[|y_t| > \beta \cdot \Delta_t] \leq 1/10.$$

It remains to prove Claim 1.4.7 and Claim 1.4.8.

Claim 1.4.7 (lower bound on Δ_t)**.** *Let Δ_t be defined in Eq.* (1.12)*. Then we have*

$$\Pr_{\delta}\left[\Delta_t > \frac{1}{k^{1/p}}\|x_{-k}\|_p\right] \leq 1/10.$$

Proof. The proof mainly includes three steps,

First, for a fixed coordinate $i \in [n]$, with probability at most $1/(100k)$, it got sampled. Taking a union bound over all k largest coordinates. We can show that with probability at least $1 - 1/100$, none of k largest coordinates is sampled. Let ξ be that event.

Second, conditioning on event ξ, we can show that

$$\mathbf{E}[\Delta_t] \leq \frac{1}{(100k)^{1/p}}\|x_{-k}\|_p.$$

Third, applying Markov's inequality, we have

$$\Pr[\Delta_t \geq a] \leq \mathbf{E}[\Delta_t]/a.$$

Choosing $a = \frac{1}{k^{1/p}}\|x_{-k}\|_p$, we have

$$\Pr\left[\Delta_t \geq \frac{1}{k^{1/p}}\|x_{-k}\|_p\right] \leq 1/10.$$

□

Claim 1.4.8 (upper bound on Δ_t)**.** *Let Δ_t be defined in Eq.* (1.12)*. For any $C_0 \geq 1000$, we have*

$$\Pr_{\delta}\left[\Delta_t < \frac{1}{(2C_0k)^{1/p}}\|x_{-C_0k}\|_p\right] \leq 1/10.$$

Proof. Without loss of generality, we can assume that all coordinates of x_i are sorted, i.e. $x_1 \geq x_2 \geq \cdots \geq x_n$. Then we split length n vector into l blocks where each block has length $s = C_0k$. Note that it is obvious $l \cdot s = n$.

For each $j \in [l]$, we use boolean variable B_j to denote that if at least one coordinate in j-th block has been sampled. For a fixed block $j \in [l]$, the probability of sampling at least one coordinate from that block is at least

$$1 - \left(1 - \frac{1}{100k}\right)^s = 1 - \left(1 - \frac{1}{100k}\right)^{C_0k} \geq 9/10.$$

Thus, we know $1 \geq \mathbf{E}[B_j] \geq 9/10$.

Warm-up. Note the probability is not allowed to take a union over all the blocks. However, if we conditioned on that each block has been sampled at least one coordinate, then we have

$$\begin{aligned}
\Delta_t^p &= \sum_{i=1}^{n} \delta_{i,t} x_i^p \\
&\geq \sum_{j=1}^{l-1} x_{js}^p \\
&\geq \sum_{j=1}^{l-1} \frac{1}{s} \left(x_{js+1}^p + x_{js+2}^p + \cdots + x_{js+s}^p \right) \\
&= \frac{1}{s} \|x_s\|_p^p.
\end{aligned}$$

Fixed. For simplicity, for each $j \in [l]$, we use set T_j to denote $\{(j-1)s+1, (j-1)s+2, \cdots, (j-1)s+s\}$.

Using random walk Lemma 1.4.1, with probability at least $99/100$, we have : for all $j \in \{2, \cdots, l\}$,

$$\sum_{j'=2}^{j} B_{j'} > (j-1)/2.$$

We know that with probability at least $99/100$, $B_1 = 1$. Then with probability at least $99/100$, we have

$$B_1 = 1, \text{ and } \sum_{j'=2}^{j} B_{j'} > (j-1)/2, \forall j \in [l].$$

We conditioned on the above event holds. Let set $S_1 \subset [n]$ denote the set of indices j such that $B_j = 1$, i.e.,

$$S_1 = \left\{ j \mid B_j = 1, j \in [n] \backslash \{1\} \right\}.$$

Let set $S_0 \subset [n]$ denote the set of indices j such that $B_j = 0$, i.e.,

$$S_0 = \left\{ j \mid B_j = 0, j \in [n] \backslash \{1\} \right\}.$$

Due to $\sum_{j'=2}^{j} B_{j'} > (j-1)/2, \forall j \in [l]$, then it is easy to see $S_1 > S_0$ and there exists a one-to-one mapping $\pi : S_0 \to S_1'$ where $S_1' \subseteq S_1$ such that for each coordinate $j \in S_0$, $\pi(j) < j$. Since we are the coordinates are being sorted already, thus

$$\begin{aligned}\sum_{j\in S_1} \|x_{T_j}\|_p^p &= \sum_{j\in S_1'} \|x_{T_j}\|_p^p \\ &= \sum_{j\in S_1'} \|x_{T_{\pi^{-1}(j)}}\|_p^p \\ &\geq \sum_{j\in S_0} \|x_{T_j}\|_p^p\end{aligned}$$

which implies that

$$\Delta_t^p = \sum_{i=1}^{n} \delta_i^p x_i^p = \sum_{j\in S_1} \|x_{T_j}\|_p^p \geq \frac{1}{2s}\|x_{-s}\|_p^p.$$

Thus, with probability at least 9/10, we have

$$\Delta_t \geq \frac{1}{(2s)^{1/p}}\|x_{-s}\|_p.$$

□

Chapter 2

Non-Linear Compressed Sensing

2.1 Compressed Sensing from Intensity-Only Measurements

2.1.1 Results

In this section we give an overview of the sublinear-time results which we have obtained for the sparse recovery problem with phaseless measurements.

First, we consider the case of noiseless signals. Similar to the classical sparse recovery where $\mathcal{O}(k)$ measurements suffice for noiseless signals by Prony's method [Pro95], it is known that $\mathcal{O}(k)$ phaseless measurements also suffice for exact recovery (up to rotation) and the decoding algorithm runs in time $\mathcal{O}(k \log k)$ [CBJC14]. Their algorithm is based on a multi-phase traversal of a bipartite random graph in a way such that all magnitudes and all phases are recovered by resolving multi-tons. We prove a result with the same guarantee, but our algorithm takes a different route using more basic tools and being less technically demanding. Apart from being significantly simpler, it also can be modified so that it trades the decoding time with the failure probability; see Remark 2.1.19.

Theorem 2.1.1 (noiseless signals)**.** *There exists a randomized construction of $\Phi \in \mathbb{C}^{m\times n}$ and a deterministic decoding procedure $\mathcal{R}$ such that $\widehat{x} = \mathcal{R}(\Phi, |\Phi x|)$ satisfies that $\widehat{x} = e^{i\cdot\theta}x$ for some $\theta \in [0, 2\pi)$ with probability at least $1 - 1/\operatorname{poly}(k)$, where Φ has $m = \mathcal{O}(k)$ measurements and $\mathcal{R}$ runs in time $\mathcal{O}(k \log k)$.*

The next results refer to approximately sparse signals and improve upon the previous ones with various degrees. For the ℓ_∞/ℓ_2 problem our result, stated below, improves upon [Nak17a] in terms of the error guarantee and the decoding time. It requires a modest assumption on the pattern of the valid phases of the heavy hitters as defined below, which is often satisfied in applications where the valid phases lie in a set of equidistant points on $\mathbb{S}^1$. Throughout this subchapter we identify $\mathbb{S}^1$ with $[0, 2\pi)$ and assume both the unoriented distance $d(\cdot,\cdot)$ and the oriented distance $\vec{d}(\cdot,\cdot)$ on $\mathbb{S}^1$ are circular. We shall also use $[m]$ to denote the set $\{1,\ldots,m\}$ for any positive integer m, a conventional notation in computer science literature.

Definition 2.1.2 (η-distinctness). *Let $P = \{p_1,\ldots,p_m\}$ be a finite set on $\mathbb{S}^1$. We say P is η-distinct if the following conditions hold: First, $d(p_i, p_j) \geq \eta$ for all distinct $i, j \in [m]$; and it holds for every pair of distinct $i, j \in [m]$ that*

$$\max_{\ell\in[m]} d(x_\ell + x_j - x_i, P) \in \{0\} \cup [\eta, \pi].$$

Intuitively, (i) means that the phases are at least η apart from each other, and (ii) means that if we rotate the set P of the valid phases to another set P' such that some valid phase coincides with another one (in the expression above x_i is rotated to the position of x_j), then either $P = P'$ or there exists an additive gap of at least η around some phase. This precludes the case where P is approximately, but not exactly, equidistant.

Definition 2.1.3 (head). *Let $x \in \mathbb{C}^n$. Define $H_k(x)$ to be (a fixed choice of) the index set of the k largest coordinates of x in magnitude, breaking ties arbitrarily.*

Definition 2.1.4 (ϵ-heavy hitters). *Let $x \in \mathbb{C}^n$. We say x_i is an ϵ-heavy hitter if $|x_i|^2 \geq \epsilon\|x_{-1/\epsilon}\|_2^2$.*

Definition 2.1.5 (phase-compliant signals). *Let $x \in \mathbb{C}^n$. Let $P \subseteq \mathbb{S}^1$ be a set of possible phases and T be the set of all $(1/k)$-heavy hitters in T. We say that x is (k, P)-compliant if $\{i \in T : \arg x_i\} \subseteq P$.*

Theorem 2.1.6 (ℓ_∞/ℓ_2 with optimal measurements). *There exists a randomized construction of $\Phi \in \mathbb{C}^{m\times n}$ and a deterministic decoding procedure $\mathcal{R}$ such that for $x \in \mathbb{C}^n$ which is $(\mathcal{O}(k), P)$-*

compliant for some η-distinct $P \subset \mathbb{S}^1$, the recovered signal $\widehat{x} = \mathcal{R}(\Phi, |\Phi x|, P)$ satisfies the ℓ_∞/ℓ_2 error guarantee with probability at least 0.6, and Φ has $m = \mathcal{O}((k/\eta)\log n)$ rows and $\mathcal{R}$ runs in time $\mathcal{O}(k/\eta + k\operatorname{poly}(\log n))$.

It is clear that the lower bound for the traditional compressive sensing problem is also a lower bound for the compressive phase retrieval problem, and it is known that the ℓ_∞/ℓ_2 compressive sensing problem requires $\Omega(k \log n)$ measurements [DBIPW10]. Therefore the theorem above achieves the optimal measurements constant η up to a constant factor.

An immediate corollary of the ℓ_∞/ℓ_2 sparse recovery algorithm is an ℓ_2/ℓ_2 sparse recovery algorithm, stated below, which improves upon [Nak17a] in approximation ratio (from a constant factor to $1+\epsilon$) and decoding time but allows a constant failure probability instead of an $o_n(1)$ failure probability as in [Nak17a].

Corollary 2.1.7 (ℓ_2/ℓ_2 with near-optimal measurements)**.** *There exists a randomized construction of $\Phi \in \mathbb{R}^{m\times n}$ and a deterministic decoding procedure $\mathcal{R}$ such that for $x \in \mathbb{C}^n$ which is $(\mathcal{O}(k), P)$-compliant for some η-distinct $P \subset \mathbb{S}^1$, the recovered signal $\widehat{x} = \mathcal{R}(\Phi, |\Phi x|, P)$ satisfies the ℓ_2/ℓ_2 error guarantee with probability at least 0.6, and Φ has $m = \mathcal{O}((k/\min\{\eta,\epsilon\})\log n)$ rows and $\mathcal{R}$ runs in time $\mathcal{O}((k/\min\{\eta,\epsilon\})\operatorname{poly}(\log n))$.*

It is also known that the classical compressive sensing problem with for-each ℓ_2/ℓ_2 error guarantee and constant failure probability requires $\Omega((k/\epsilon)\log(n/k))$ measurements [PW11]. Our result above achieves the optimal number of measurements up to a logarithmic factor.

For the ℓ_2/ℓ_2 error guarantee with $1/\operatorname{poly}(n)$ failure probability, we shall increase the number of measurements to $\mathcal{O}(k/\epsilon^2 \cdot \log n)$, as in the following theorem. This improves on [Nak17a] in terms of the approximation ratio, the failure probability and most importantly the decoding time. We note that the best decoding time of the existing algorithms is $\mathcal{O}(k^{1+o(1)}\operatorname{poly}(\log n))$. However, we restrict the set P of valid phases to an equidistant set with gap at least η, that is, up to a rotation, $P = \{e^{2\pi i \frac{j}{m}}\}_{j=0,\ldots,m-1}$ for some $m \le 2\pi/\eta$.

Theorem 2.1.8 (ℓ_2/ℓ_2 with low failure probability)**.** *There exists a randomized construction of $\Phi \in \mathbb{R}^{m\times n}$ and a deterministic decoding procedure $\mathcal{R}$ such that for each $x \in \mathbb{C}^n$ which is*

$(\mathcal{O}(k/\epsilon), P)$-compliant for some $P \subset \mathbb{S}^1$ that is equidistant with gap at least η, the recovered signal $\widehat{x} = \mathcal{R}(\Phi, |\Phi x|, P)$ satisfies the ℓ_2/ℓ_2 error guarantee with probability at least $1-\delta$, and Φ has $m = \mathcal{O}((\epsilon\eta)^{-2} k \log(n/\delta))$ rows and $\mathcal{R}$ runs in time $\mathcal{O}((\epsilon\eta)^{-2} k \operatorname{poly}(\log(n/\delta)))$.

We note that the number of measurements becomes $\mathcal{O}(\epsilon^{-2} k \log n)$ when η is a constant and the failure probability $\delta = 1/\operatorname{poly}(n)$, which is usually the case.

2.1.2 Toolkit

Theorem 2.1.9 (Bernstein's inequality, [DP09, p9])**.** *Let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables with $X_i - \mathbf{E} X_i \leq K$ and $\sigma^2 = \sum_{i=1}^n \mathbf{E} X_i^2 - (\mathbf{E} X_i)^2$. Then*

$$\Pr\left\{ \sum_{i=1}^n X_i - \mathbf{E} \sum_{i=1}^n X_i \geq \lambda \right\} \leq \exp\left(-\frac{\frac{1}{2}\lambda^2}{\sigma^2 + \frac{1}{3}K\lambda} \right).$$

The two results concern heavy hitters, one for estimating the value of a heavy hitter and the other for finding the positions of the heavy hitters.

Theorem 2.1.10 (Count-Sketch, [CCF02])**.** *There exist a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$ with $m = \mathcal{O}(K \log n)$ and a deterministic algorithm $\mathcal{R}$ such that given $y = |\Phi x|$ for $x \in \mathbb{C}^n$, with probability at least $1 - 1/\operatorname{poly}(n)$, for every $i \in [n]$, the algorithm $\mathcal{R}$ returns in time $\mathcal{O}(\log n)$ an estimate $|\widehat{x}_i|$ such that*

$$||x_i| - |\widehat{x}_i||^2 \leq \frac{1}{K} \|x_{-K}\|_2^2.$$

Theorem 2.1.11 (Heavy hitters, [LNNT16])**.** *There exist a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$ with $m = \mathcal{O}(K \log n)$ and a deterministic algorithm $\mathcal{R}$ such that given $y = |\Phi x|$ for $x \in \mathbb{C}^n$, with probability at least $1 - 1/\operatorname{poly}(n)$ the algorithm $\mathcal{R}$ returns in time $\mathcal{O}(K \cdot \operatorname{poly}(\log n))$ a set S of size $\mathcal{O}(K)$ containing all $(1/K)$-heavy hitters of x.*

We remark that the paper [LNNT16] does not consider complex signals but the extension to complex signals is straightforward. The algorithm is not designed for the phaseless sparse recovery either, the identification algorithm nevertheless works when the measurements are phaseless because it only relies on the magnitudes of the bucket measurements; see

Theorem 2 and Section B in [LNNT16]. Estimating the values of the candidate coordinates requires knowing the phases of the measurements but our theorem above does not concern this part.

Theorem 2.1.12 ([ER59])**.** *Let V be a set of n vertices. There exists an absolute constant κ such that $\kappa n \log n$ uniform samples of pairs of distinct vertices in V induce a connected graph with probability at least $1 - 1/\operatorname{poly}(n)$.*

Theorem 2.1.13 (Phase Prediction, [TMB$^+$17])**.** *Let V be a set of size n and $\pi : V \to \mathbb{S}^1$ be the phase function of the elements in V. A query returns a random pair $\{u, v\} \in V \times V$ uniformly at random, along with an estimate of the oriented distance $\vec{d}(\pi(u), \pi(v))$, which could be incorrect with probability $1/3$. There exists an absolute constant c_{SP} such that $c_{\text{SP}} n \log n$ queries suffice to find the relative phase differences for all $u \in V$ with probability $1 - 1/\operatorname{poly}(n)$ in time $\mathcal{O}(n^2 \log n)$.*

We remark that the paper [TMB$^+$17] concerns only the sign prediction for real signals, i.e., $\pi : S \to \{-1, 1\}$, and can be straightforwardly generalized, with minimum changes, to the setting in the theorem statement above. The runtime in [TMB$^+$17] is $\mathcal{O}(n^3 \log n)$ since for each pair (u, v) it runs a sign prediction algorithm in $\mathcal{O}(n \log n)$ time to determine the sign difference between u and v (correct with high probability) and enumerate all $\Theta(n^2)$ pairs. This is unnecessary, as we can fix u and enumerate v so we run the sign prediction algorithm just $\mathcal{O}(n)$ times.

The following lemmata will be crucial in the analysis of our algorithms.

Lemma 2.1.14. *Suppose that $x, y, n_1, n_2, n_3 \in \mathbb{C}$ such that $|n_1|, |n_2|, |n_3| \leq \epsilon \min\{|x|, |y|\}$ for some $\epsilon \leq 1/9$. Denote by θ be the phase difference between x and y. Then given the norms*

$$|x + n_1|, |y + n_2|, |x + y + n_1 + n_2 + n_3|,$$

we can recover θ up to an additive error of $c_0\sqrt{\epsilon}$. Furthermore, if $\theta \in (c\epsilon, \pi - c\epsilon)$ we can recover θ up to an additive error of $c\epsilon$.

Proof. If we know $|x|$, $|y|$ and $|x + y|$, it follows from the Law of Cosines that

$$\cos(\pi - \theta) = \frac{|x|^2 + |y|^2 - |x + y|^2}{2|x| \cdot |y|} = \frac{-\Re x\overline{y}}{|x| \cdot |y|}.$$

Let $x' = x + n_1$ and $y' = y + n_2$ then $x + y + n_1 + n_2 + n_3 = x' + y' + n_3$. Suppose the phase difference between x' and y' is θ', then we would pretend $x' + y' + n_3$ to be $x' + y'$ and obtain an approximation θ'' to θ' as

$$\cos(\pi - \theta'') = \frac{|x'|^2 + |y'|^2 - |x' + y' + n_3|^2}{2|x'| \cdot |y'|} = \frac{-\Re x'\overline{y} - \Re x'\overline{n_3} - \Re y\overline{n_3} - |n_3|^2}{|x'| \cdot |y'|}.$$

Hence

$$|\cos(\pi - \theta'') - \cos(\pi - \theta')| \le \frac{|x'||n_3| + |y'||n_3| + |n_3|^2}{|x'| \cdot |y'|} \le \epsilon + \epsilon + 9\epsilon^2 \le 3\epsilon.$$

Similarly we have

$$\cos(\pi - \theta') = \cos(\pi - \theta) \cdot \frac{|x|}{|x + n_1|} \cdot \frac{|y|}{|y + n_2|} + \nu, \quad |\nu| \le c_1\epsilon,$$

and thus

$$\cos(\pi - \theta') - \cos(\pi - \theta) = \cos\theta \left(\frac{|x|}{|x + n_1|} \cdot \frac{|y|}{|y + n_2|} - 1 \right) + \nu.$$

Note that $\frac{|x|}{|x+n_1|}, \frac{|y|}{|y+n_2|} \in [\frac{1}{1+\epsilon}, \frac{1}{1-\epsilon}]$, it follows that

$$|\cos(\pi - \theta') - \cos(\pi - \theta)| \le c_2\epsilon$$

and thus

$$|\cos(\pi - \theta'') - \cos(\pi - \theta)| \le c_3\epsilon.$$

Therefore there exists c_0 such that $|\theta - \theta'| \le c_0\sqrt{\epsilon}$; and furthermore, there exists c such that when $\theta \in (c\epsilon, \pi - c\epsilon)$, it holds that

$$|\theta'' - \theta| \le c\epsilon. \qquad \square$$

Lemma 2.1.15. *Let x, y, n_1, n_3, ϵ be as in Lemma 2.1.14. Suppose that* $\arg y = \arg x + \theta$ *for some $\theta \in (0, 2\pi)$, where addition is modulo 2π. Given the norms*

$$|x + n_1|, |y|, |x + y + n_1 + n_3|, |x + \beta y + n_1 + n_3|, \quad \beta = e^{2c\epsilon i},$$

we can recover θ up to an additive error of $c\epsilon$, provided that $\theta \in (2c\epsilon, \pi - 2c\epsilon) \cup (\pi + 2c\epsilon, 2\pi - 2c\epsilon)$.

Proof. By Lemma 2.1.14, we can recover $|\theta|$ up to an additive error of $c\epsilon$ when $|\theta| \in (c\epsilon, \pi - c\epsilon)$. To determine the sign, we rotate y by angle $2c\epsilon$ and test the angle between x and this rotated y again by Lemma 2.1.14. Suppose that the angle between x and βy is ϕ and we have an estimate of $|\phi|$ up to an additive error of $c\epsilon$, provided that $|\phi| \in (c\epsilon, \pi - c\epsilon)$, which means that $|\theta| \in (2c\epsilon, \pi - 2c\epsilon)$. It holds that

$$|\phi| - |\theta| = \begin{cases} -2c\epsilon, & \theta > 0; \\ 2c\epsilon, & \theta < 0. \end{cases}$$

when $|\theta| \in (2c\epsilon, \pi - 2c\epsilon)$. The left-hand side is approximated up to an additive error of $2c\epsilon$ and thus we can distinguish the two cases. $\square$

Lemma 2.1.16 (relative phase estimate)**.** *Let x, y, n_1, n_3, ϵ be as in Lemma 2.1.14 and further assume that $c\epsilon \leq \pi/9$. Suppose that $\arg y = \arg x + \theta$ for some $\theta \in (0, 2\pi)$, where addition is modulo 2π. Given the norms*

$$|x + n_1|, |y|, |x + e^{i(2c\epsilon j + \frac{\pi}{2}\ell)} y + n_1 + n_3|, \quad j, \ell = 0, 1$$

we can recover θ up to an additive error of $c\epsilon$.

Proof. From Lemma 2.1.15, we know that we can recover θ up to an additive error of $c\epsilon$ when $\theta \in I$, where $I = (2c\epsilon, \pi - 2c\epsilon) \cup (\pi + 2c\epsilon, 2pi - 2c\epsilon)$. We accept the estimate if the estimate is in the range of $I' := (3c\epsilon, \pi - 3c\epsilon) \cup (\pi + 3c\epsilon, 2pi - 3c\epsilon)$.

Consider the phase difference between x and $e^{i\pi/2} y$ and suppose that $\arg(e^{i\pi/2} y) = \arg x + \phi$, then we can recover ϕ up to an additive error of $c\epsilon$ for $\phi \in I$, that is, for $\theta \in J := (\pi/2 + 2c\epsilon, 3\pi/2 - 2c\epsilon) \cup (-\pi/2 + 2c\epsilon, \pi/2 - 2c\epsilon)$, which is I rotated by $\pi/2$. We accept the estimate when it is in the range of $J' := (\pi/2 + 3c\epsilon, 3\pi/2 - 3c\epsilon) \cup (-\pi/2 + 3c\epsilon, \pi/2 - 3c\epsilon)$.

Note that $I' \cup J'$ covers the whole $\mathbb{S}^1$ when $c\epsilon < \pi/8$. $\square$

2.1.3 Noiseless Signals

We shall need the following theorem from [Nak17a], which shows that one can recover an exactly K-sparse signal up to a global phase using $\mathcal{O}(K)$ measurements and in time $\mathcal{O}(K^2)$.

The runtime was not claimed in [Nak17a] but is easy to analyse.

Theorem 2.1.17 ([Nak17a])**.** *Let L be a $2K \times 2K$ lower triangular matrix with each non-zero entry being 1, and A be the vertical concatenation of L and $I_{2k\times 2k}$. Let F_{2K} be the first $2K$ rows of a Discrete Fourier Transform matrix. For $x \in \mathbb{C}^n$ such that $\|x\|_0 \le K$, given $y = |AF_{2K}x|$, we can recover x up to a rotation in time $\mathcal{O}(K^2)$.*

We are now ready to prove Theorem 2.1.1, which we restate below.

Theorem 2.1.18 (noiseless signals)**.** *here exists a randomized construction of $\Phi \in \mathbb{C}^{m\times n}$ and a deterministic decoding procedure $\mathcal{R}$ such that $\widehat{x} = \mathcal{R}(\Phi, |\Phi x|)$ satisfies that $\widehat{x} = e^{i\cdot\theta}x$ for some $\theta \in [0, 2\pi)$ with probability at least $1 - 1/\operatorname{poly}(k)$, where Φ has $m = \mathcal{O}(k)$ measurements and $\mathcal{R}$ runs in time $\mathcal{O}(k \log k)$.*

Proof. Let $B = k/(c \log k)$ and $h : [n] \to [B]$ be an $\mathcal{O}(k)$-wise independent hash function, where c is a constant. We hash all n coordinates into B buckets using h. It is a typical application of Chernoff bound that the buckets have small size (see Lemma 2.1.28), more specifically,

$$\Pr\left\{\exists j \in [B] : |h^{-1}(j) \cap \operatorname{supp}(x)| > 5 \log k\right\} \le \frac{1}{\operatorname{poly}(k)}.$$

In each bucket we run the algorithm of Theorem 2.1.17 with $K = 5 \log k$. The number of measurements used for each bucket is $\Theta(\log k)$. For each $j \in [B]$, we can find $x_{h^{-1}(j)}$ up to a global phase, so it remains to find the relative phases across different $x_{h^{-1}(j)}$.

Let $F_1, \dots, F_{\log k}$ be independent random $0/1$ matrices of n columns, where F_ℓ has $\alpha c_R 2^\ell$ rows for $\ell > \lceil \frac{1}{2}\log k\rceil + 1$ and $\alpha c_R 2^\ell \log k$ rows otherwise, and α is a sufficiently large constant. Each entry in F_ℓ equals to 1 with probability $2^{-\ell}$, that is, $\mathbf{E}[(F_\ell)_{i,j}] = 2^{-\ell}$. Our measurement matrix is the vertical concatenation of $F_1, \dots, F_{\log k}$. The total number of measurements is

$$\frac{k}{c\log k} \cdot \Theta(\log k) + \sum_{\ell > \lceil \frac{1}{2}\log k\rceil} \alpha c_R 2^\ell + \sum_{\ell \le \lceil \frac{1}{2}\log k\rceil} \alpha c_R \log k \cdot 2^\ell = \mathcal{O}(k)$$

as desired. Next we show the correctness.

We set $\operatorname{supp}(x) = \bigcup_{j=1}^{B} \operatorname{supp}(x_{h^{-1}(j)})$ and compute ℓ such that $2^{\ell-1} \leq |\operatorname{supp}(x)| \leq 2^{\ell}$. Now, consider the rows of F_ℓ. Denote the j-th row of F_ℓ by $(F_\ell)_j$. Define a row index set J to be

$$J = \left\{ j : \left|\operatorname{supp}((F_\ell)_j) \cap \operatorname{supp}(x)\right| = 2 \right\}.$$

Observe that each j is contained in J with constant probability and we focus on the measurements corresponds to the rows in J. From such a measurement we can obtain a random pair $\{u, v\} \subseteq \operatorname{supp}(x)$ and, moreover, $(h(u), h(v))$ is uniformly random on $[B] \times [B]$. We also obtain $|x_u + x_v|$ and, because we also know $|x_u|, |x_v|$, we can infer the relative phase between x_u, x_v. The relative phases we obtain are always correct since the signal is noiseless. Let $\mathcal{M}$ be the ordered set of such pairs (u, v) along with the label that we obtain about the relative phase between u and v. We split $\mathcal{M}$ into equal-sized sets of edges $\mathcal{M}_1, \mathcal{M}_2, \ldots,$ each of size $c_R \ell 2^{\ell}$. In each $\mathcal{M}_j$ we run a depth-first search to infer the relative phases. If the graph is connected, which happens with probability $1 - 2^{-\ell}$, we will find all the relative phases correctly. We take the pattern of relative phases that appears most often. It follows from standard Chernoff bounds and our choice of parameters for F_ℓ that the overall failure probability is at most $1/\operatorname{poly}(k)$. $\square$

Remark 2.1.19. *Note that if we hash to $k^{1-\alpha}$ buckets, solve in each bucket and then combine the buckets, we can obtain a failure probability at most $\exp(-k^{\alpha})$ and a running time of $\mathcal{O}(k^{1+\alpha})$. This is a trade-off between decoding time and failure probability that the previous algorithms did not achieve.*

Remark 2.1.20. *We show how to implement efficiently the routine which finds the set of rows of F whose support intersect $\operatorname{supp}(x)$ at 2 coordinates. For $\ell > \lceil \frac{1}{2}\log k \rceil$ the expected number of rows of F_ℓ containing an index $i \in \operatorname{supp}(x)$ is $2^{-\ell}\alpha c_R 2^{\ell} = \alpha c_R$. So the probability that there are more than $2\alpha c_R 2^{\ell}$ pairs (i, q) such that $i \in \operatorname{supp}(x) \cap \operatorname{supp}(F_\ell)_q$ is $\exp(-\Omega(2^{\ell})) < 1/\operatorname{poly}(k)$. A similar result can be obtained for $\ell < \frac{1}{2}\log k$. Suppose that F_ℓ is stored using n lists of nonzero coordinates in each column, we can afford to iterate over all such pairs (i, q), keep an array $C[q]$ that holds the cardinality of $\operatorname{supp}(x) \cap \operatorname{supp}((F_\ell)_q)$. At the end, we find the values of q with $C[q] = 2$. This* *implementation makes the algorithm run in $\mathcal{O}(k \log k)$ time.*

2.1.4 ℓ_∞/ℓ_2 Algorithm

In this section, we set c to be the constant in Lemma 2.1.16 and $\epsilon = \min\{\eta/(5c), \pi/(9c)\}$. Let $P \subseteq \mathbb{S}^1$ be η-distinct and suppose that it contains the phases of all $1/(\widetilde{C}k)$-heavy hitters for some (large) constant $\widetilde{C}$.

We first describe our construction of the measurement matrix Φ and then present the analysis and the recovery algorithm. Let $R = c_R \log n$ for some constant c_R to be determined. The overall sensing matrix Φ is a layered one as

$$\Phi = \begin{bmatrix} \Phi_{\mathrm{HH}} \\ \Phi_{\mathrm{CS}} \\ \rho \\ \Phi_1 \\ \vdots \\ \Phi_R \end{bmatrix}.$$

Here

- Φ_{HH} is the sensing matrix in Theorem 2.1.11 with $K = k$.
- Φ_{CS} is the sensing matrix of Count-Sketch with $K = Ck/\epsilon$.
- ρ is a row vector
$$\rho = \begin{pmatrix} \eta_1 g_1 & \eta_2 g_2 & \cdots & \eta_n g_n \end{pmatrix},$$
where η_i are i.i.d. Bernoulli variables with $\mathbf{E}\,\eta_i = 1/(C_0 k)$ and g_i are i.i.d. $\mathcal{N}(0,1)$ variables.
- Each Φ_r ($r \in [R]$) is a matrix of $4B$ rows defined as follows, where $B = c_B k/\epsilon$. Let $h_r : [n] \to [B]$ be a pairwise independent hash function and $\{\sigma_{r,i}\}_{i=1}^n$ be pairwise

Algorithm 4 Algorithm for the ℓ_∞/ℓ_2 phaseless sparse recovery. Assume that the elements in P are sorted.

```
1: $S \leftarrow$ the set returned by the algorithm in Theorem 2.1.11 with $K = k$
2: Run a Count-Sketch algorithm with $K = Ck/\epsilon$ to obtain an approximation $|\widehat{x}_i|$ to $|x_i|$
   for all $i \in S$
3: $L \leftarrow |\sum_{i=1}^{n} \eta_i g_i x_i|$
4: $S' \leftarrow \{i \in S : |\widehat{x}_i| \geq L\}$
5: if $L = 0$ then
6:     Run the algorithm for the noiseless case with sparsity $C_2 k$
7: else
8:     for each $r \in [R]$ do
9:         $b_i \leftarrow h_r(i)$ for all $i \in S'$
10:        for each $i \in S'$ with distinct $b_i$ do
11:            $\widetilde{\theta}_{r,i} \leftarrow$ estimate of phase difference between $x_i$ and $\langle \rho, x \rangle$ using Lemma 2.1.16
12:    for each $i \in S'$ do
13:        $\widetilde{\theta}_i \leftarrow \text{median}_{r \in [R]} \widetilde{\theta}_{r,i}$
14:    Choose an arbitrary $i_0 \in S'$
15:    for each $p \in P$ do
16:        $\theta'_{i_0} \leftarrow p$
17:        $\theta'_i \leftarrow \theta'_{i_0} + \widetilde{\theta}_i - \widetilde{\theta}_{i_0}$ for all $i \in S' \setminus \{i_0\}$
18:        if $d(\theta'_i, P) \leq \eta/2$ for all $i \in S'$ then
19:            return $\widehat{x}$ supported on $S'$ with $\arg \widehat{x}_i = \theta_i$
20:        end if
21: end if
```

independent random signs. Define a $B \times n$ hashing matrix H_r as

$$(H_r)_{j,i} = \begin{cases} (1-\eta_i)\sigma_{r,i}, & i \in h_r^{-1}(j); \\ 0, & \text{otherwise.} \end{cases}$$

The $4B$ rows of Φ_r are defined to be

$$e^{i(2c\epsilon\ell_1 + \frac{\pi}{2}\ell_2)}\rho + (H_r)_{b,\cdot}, \quad \ell_1, \ell_2 = 0, 1, \; b = 1, \ldots, B.$$

We present the recovery algorithm in Algorithm 4, where we assume that the set P of valid phases has been sorted. In the following we analyse the algorithm in four steps.

Step 1 By Theorem 2.1.11, the set S has size $\mathcal{O}(k)$ and, with probability $1 - 1/\operatorname{poly}(n)$, contains all $(1/k)$-heavy hitters. The Count-Sketch (Theorem 2.1.10) guarantees that

$$||x_i| - |\widehat{x}_i||^2 \leq \frac{\epsilon}{Ck}\|x_{-k}\|_2^2 \tag{2.1}$$

for all $i \in S$ with probability at least $1 - 1/\operatorname{poly}(n)$.

Step 2 We shall see that L, calculated in Line 3, 'approximates' the desirable tail $\frac{1}{k}\|x_{-k}\|_2^2$.

First we upper bound L. Decompose x into real and imaginary parts as $x = y + iz$ with $y, z \in \mathbb{R}^n$ and consider $L_1 = \sum_i \eta_i g_i y_i$ and $L_2 = \sum_i \eta_i g_i z_i$. Note that $L^2 = L_1^2 + L_2^2$.

Choosing $C_0 \geq 200$, we have

$$\Pr\left\{\eta_i = 0 \text{ for all } i \in H_k(y) \cup H_k(z)\right\} \geq 0.99 \tag{2.2}$$

Condition on this event below. Note that $L_1 \sim \mathcal{N}(0, |\sum_i \eta_i y_i|_2^2)$, and

$$\Pr\left\{L_1^2 \geq 2.282^2 \left|\sum_i \eta_i y_i\right|_2^2\right\} \leq 2\Phi(-2.282) \leq 0.0225.$$

On the other hand, $\mathbf{E}\,|\sum_i \eta_i y_i|^2 = \sum_i (\mathbf{E}\,\eta_i) y_i^2 \leq \|y_{-k}\|_2^2/(C_0 k)$ thus

$$\Pr\left\{\left|\sum_i \eta_i y_i\right|^2 \geq \frac{20}{C_0 k}\|x_{-k}\|_2^2\right\} \leq 0.05,$$

and hence

$$\Pr\left\{L_1^2 \geq \frac{105}{C_0 k}\|y_{-k}\|_2^2\right\} \leq 0.0725.$$

Similarly we have

$$\Pr\left\{L_2^2 \geq \frac{105}{C_0 k}\|z_{-k}\|_2^2\right\} \leq 0.0725.$$

Therefore, taking a union bound of both events above and noting that $\|y_{-k}\|_2^2 + \|z_{-k}\|_2^2 \leq \|x_{-k}\|_2^2$, we have that

$$\Pr\left\{L^2 \geq \frac{105}{C_0 k}\|x_{-k}\|_2^2\right\} \leq 0.145. \tag{2.3}$$

We therefore obtained an upper bound of L. The next lemma lower bounds L.

Lemma 2.1.21. *With probability at least* 0.8*, it holds that* $L^2 \geq \frac{1}{C_1 k}\|x_{-C_2 k}\|_2^2$*, where* C_1, C_2 *are absolute constants.*

Proof. Decompose x into real and imaginary parts as $x = y + iz$ with $y, z \in \mathbb{R}^n$. Consider $L_1 = \sum_i \eta_i g_i y_i$ and $L_2 = \sum_i \eta_i g_i z_i$, which are both real. Note that $L^2 = L_1^2 + L_2^2$.

First consider L_1. We sort coordinates $[n] \setminus H_k(y)$ by decreasing order of magnitude. Then, we split the sorted coordinates into continuous blocks of size $C_2' k$ and let S_j denote the j-th block. Let δ_j be the indicator variable of the event that there exists $i \in S_j$ such that $\eta_i = 1$, then δ_j's are i.i.d. Bernoulli variables with $\mathbf{E}\,\delta_j = 1 - (1 - 1/(C_0 k))^{C_2' k} \geq 1 - \exp(-C_2'/C_0)$, which can be made arbitrary small by choosing C_2' big enough. It is a standard result in random walks (see, e.g., [KT75, p67]) that when $\mathbf{E}\,\delta_j$ is small enough, with probability at least 0.95, every partial prefix of the 0/1 sequence $(\delta_1, \delta_2, \delta_3, \ldots)$ will have more 1s than 0s. Call this event $\mathcal{E}$. In fact, one can directly calculate that $\Pr(\mathcal{E}) = 1 - (1-p)^2/p$ when $p := \mathbf{E}\,\delta_j \geq 1/2$, and thus one can take $C_2' = \lceil 1.61 C_0 \rceil$ such that $\Pr(\mathcal{E}) \geq 0.95$.

Condition on $\mathcal{E}$. We can then define an injective function π from $\{j : \delta_j = 0\}$ to $\{j : \delta_j = 1\}$. Specifically, we define $\pi(j) = \ell$, where δ_j is the k-th 0 in the sequence and ℓ is the k-th 1 in the sequence. Clearly that π is injective, $\pi(j) < j$ and $\delta_{\pi(j)} = 1$. It follows that

$$\begin{aligned}
\sum_{i \in [n]} \eta_i |y_i|^2 \geq \sum_j \delta_j \|S_{j+1}\|_\infty^2 &\geq \sum_{j:\delta_j = 1} \frac{1}{C_2' k}\|S_{j+1}\|_2^2 \\
&\geq \frac{1}{2}\sum_{j:\delta_j=1} \frac{1}{C_2' k}\|S_{j+1}\|_2^2 + \frac{1}{2}\sum_{\substack{j:\delta_j = 1 \\ \pi^{-1}(j) \text{ exists}}} \frac{1}{C_2' k}\|S_{\pi^{-1}(j)}\|_2^2 \\
&\geq \frac{1}{2}\sum_{j:\delta_j=1} \frac{1}{C_2' k}\|S_{j+1}\|_2^2 + \frac{1}{2}\sum_{j:\delta_j=0} \frac{1}{C_2' k}\|S_j\|_2^2 \\
&\geq \frac{1}{2C_2' k}\|y_{-C_2' k}\|_2^2.
\end{aligned}$$

This implies that $L_1 = \sum_i \eta_i g_i y_i$ with probability at least 0.95 will stochastically dominate a gaussian variable $\mathcal{N}(0, \frac{1}{2C_2 k}\|y_{-C_2 k}\|_2^2)$. Combining with the fact that $\Pr_{g \sim \mathcal{N}(0,1)}\{|g| \leq \frac{1}{16}\} \leq 0.05$, we see that

$$\Pr\left\{L_1^2 \geq \frac{1}{16^2 \cdot 2C_2' k}\|y_{-C_2' k}\|_2^2\right\} \geq 0.9.$$

Similarly for the imaginary part z and L_2,

$$\Pr\left\{L_2^2 \geq \frac{1}{16^2 \cdot 2C_2'k}\|z_{-C_2'k}\|_2^2\right\} \geq 0.9.$$

Condition on that both events above happen. For notational convenience, let $T_1 = H_{C_2'k}(y)$ and $T_2 = H_{C_2'k}(z)$, then

$$\begin{aligned}
L^2 = L_1^2 + L_2^2 &\geq \frac{1}{16^2 \cdot 2C_2'k}(\|y_{T_1^c}\|_2^2 + \|z_{T_2^c}\|_2^2) \\
&\geq \frac{1}{16^2 \cdot 2C_2'k}\|y_{(T_1\cup T_2)^c}\|_2^2 + \|z_{(T_1\cup T_2)^c}\|_2^2) \\
&= \frac{1}{16^2 \cdot 2C_2'k}\|x_{(T_1\cup T_2)^c}\|_2^2 \\
&\geq \frac{1}{16^2 \cdot 2C_2'k}\|x_{-2C_2'k}\|_2^2.
\end{aligned}$$

Therefore, we can take $C_2 = 2C_2'$ above and $C_1 = 16^2 C_2$. □

Combining (2.2), (2.3) and Lemma 2.1.21 and taking $C_0 = 210$, we conclude that with probability at least $1 - 0.365$,

$$\frac{1}{C_1 k}\|x_{-C_2k}\|_2^2 \leq L^2 \leq \frac{1}{2k}\|x_{-k}\|_2^2. \tag{2.4}$$

Step 3 We now show that the trimmed set S' is good in the sense that its elements are not too small and it contains all $(1/k)$-heavy hitters. This is formalized in the following lemma.

Lemma 2.1.22. *With probability at least* 0.63, *it holds that*

(i) $|x_i|^2 \geq \frac{1}{2C_1k}\|x_{-C_2k}\|_2^2$ *for all* $i \in S'$*; and*

(ii) S' *contains all coordinates* i *such that* $|x_i|^2 \geq \frac{1}{k}\|x_{-k}\|_2^2$.

Proof. The events (2.1) and (2.4) happen simultaneously with probability at least $1 - 0.365 - 1/\operatorname{poly}(n) \geq 1 - 0.37$. Condition on both events. Let $C = \frac{\sqrt{2}}{\sqrt{2}-1}C_1$, then

(i) for $i \in S'$, it holds that $|x_i| \geq L - \frac{1}{\sqrt{Ck}}\|x_{-C_2k}\|_2 \geq \frac{1}{\sqrt{2C_1k}}\|x_{-C_2k}\|_2$;

(ii) if $|x_i|^2 \geq \frac{1}{k}\|x_{-k}\|_2^2$, then $|\widehat{x}_i| \geq \frac{1}{\sqrt{k}}\|x_{-k}\|_2 - \frac{1}{\sqrt{Ck}}\|x_{-k}\|_2 \geq L$. □

Step 4 The rest of the algorithm is devoted to finding the relative phases among $i \in S'$. We have from our construction the measurements

$$\left| e^{i(2c\epsilon\ell_1+\frac{\pi}{2}\ell_2)} \sum_{i=1}^{n} \eta_i g_i x_i + \sum_{i\in h_r^{-1}(j)} (1-\eta_i)\sigma_{i,r} x_i \right|, \quad \ell_1,\ell_2 = 0,1,\ j\in[B],\ r\in[R].$$

We note that Line 11 in the algorithm is valid because we have access to

$$|\widehat{x}_i|,\ \left|\sum_{i=1}^{n}\eta_i g_i x_i\right|,\ \left|\widehat{x}_i + e^{i(2c\epsilon\ell_1+\frac{\pi}{2}\ell_2)}\sum_{i=1}^{n}\eta_i g_i x_i + \sum_{i'\in h_r^{-1}(h_r(i))\setminus S'} (1-\eta_{i'})\sigma_{i',r}x_{i'}\right|.$$

The analysis of this step directly leads to a proof of Theorem 2.1.6, which we show below.

Proof of Theorem 2.1.6. First we condition on the success of Lemma 2.1.22, which hold with probability at least $1-0.37$.

Fix an $i\in S'$. For $r\in[R]$, the probability that it is isolated from every other $i'\in S'$ is $\frac{1}{C_B}$. Define the random variable

$$Z = \sum_{i'\in h_r^{-1}(h_r(i))\setminus S'} (1-\eta_{i'})\sigma_{i',r}x_{i'}.$$

Observe that

$$\Pr\left\{\left|h_r^{-1}(h_r(i))\cap H_{C_2k}(x)\right| = 1\right\} \ge 1-\frac{C_2}{C_B}$$

and that

$$\mathbf{E}\,|Z|^2 = \frac{1}{C_B k}\|x_{-C_2k}\|_2^2.$$

By Markov's inequality, we have that $|Z|^2 \le \frac{10\epsilon}{C_B k}\|x_{-k}\|_2^2$ with probability at least 0.1. Choose C_B such that $\frac{10}{C_B}\le\frac{1}{2C_1}$ and $\frac{C_2}{C_B}<\frac{1}{10}$, then the assumptions on noise magnitude in Lemma 2.1.16 will be satisfied for x_i with probability at least 0.8.

Let θ_i be the (oriented) phase difference x_i with $\sum_{j=1}^n \eta_j g_j x_j$. We can invoke Lemma 2.1.16 and obtain an estimate $\widetilde{\theta}_{r,i}$ which satisfies $|\widetilde{\theta}_{r,i}-\theta_i|\le c\epsilon$. This happens with probability at least 0.8 as demonstrated above. Taking the median over $R = c_R\log n$ repetitions with an appropriate constant c_R, we see that $\widetilde{\theta}_i$ satisfies

$$|\widetilde{\theta}_i - \theta_i| \leq c\epsilon \tag{2.5}$$

with probability at least $1 - 1/n^2$. This allows for taking a union bound over all $i \in S'$. Therefore (2.5) holds for all $i \in S'$ simultaneously with probability $\geq 1 - 1/n$.

Next, assume that it happens that (2.5) holds for all $i \in S'$. Consider the for-loop of Lines 15 to 20. It is clear that when θ'_{i_0} is exactly the phase of x_{i_0}, it will hold that θ'_i is an accurate estimate to the phase of x_i up to an additive error of $2c\epsilon < \eta/2$. The if-clause in Line 18 will be true and the algorithm will terminate with an $\widehat{x}$. Since the phases of the entries are at least η apart, there will be no ambiguity in rounding and the phases in $\widehat{x}$ are all correct, hence the error $\|x - \widehat{x}\|_2$ only depends on the magnitude errors, which is exactly (2.1), obtained from applying Count-Sketch. When θ'_{i_0} is not x_{i_0}, by the rotational (k, η)-distinctness, $\{\theta'_i\}$ will coincide with P exactly or the if-clause will not be true. This shows the correctness.

Remove the conditioning at the beginning of the proof increases the overall failure probability by an additive $1/n$. The overall failure probability is therefore at most $0.37 + 1/n < 0.4$.

Number of Measurements The submatrix Φ_{HH} has $\mathcal{O}(k \log n)$ rows, the submatrix Φ_{CS} has $\mathcal{O}((k/\epsilon)\log n)$ rows, each Φ_r for $r \in [R]$ has $\mathcal{O}(k/\epsilon)$ rows. Hence the total number of rows is dominated by that of Φ_{CS} and the R independent copies of Φ_r's, that is, $\mathcal{O}((k/\epsilon)\log n + R(k/\epsilon)) = \mathcal{O}((k/\epsilon)\log n) = \mathcal{O}((k/\eta)\log n)$.

Runtime Line 1 runs in time $\mathcal{O}(k\,\mathrm{poly}(\log n))$, Line 2 in time $\mathcal{O}(|S|\log n) = \mathcal{O}(k \log n)$, Line 4 in time $\mathcal{O}(k)$. The runtime before the if-branch of Line 5 is thus $\mathcal{O}(k\,\mathrm{poly}(\log n))$.

For the if-branch of Line 5, the noiseless case runs in time $\mathcal{O}(k \log k)$, the for-loop from Line 8 to 11 in time $\mathcal{O}(Rk \log k) = \mathcal{O}(k\,\mathrm{poly}(\log n))$, the for-loop from Line 12 to 13 in time $\mathcal{O}(R|S'|) = \mathcal{O}(k \log n)$, the for-loop from Line 15 to 20 in time $\mathcal{O}(k/\eta)$ since $|P| = \mathcal{O}(1/\eta)$, the if-clause in Line 18 can be verified in time $\mathcal{O}(k)$ if the elements in P are sorted in advance.

Hence the total runtime of the if-branch of Line 5 is $\mathcal{O}(k/\eta + k\operatorname{poly}(\log n))$.

Therefore, the overall runtime is $\mathcal{O}(k/\eta + k\operatorname{poly}(\log n))$. $\square$

2.1.5 ℓ_2/ℓ_2 with low failure probability for real signals

In this section, we assume that the valid phases are equidistant on $\mathbb{S}^1$ with a gap at least η for all $1/(\widetilde{C}k)$-heavy hitters of x, where $\widetilde{C}$ is a (large) absolute constant.

Overview

Our algorithm resembles the real-signal algorithm in [Nak17a], but with a careful modification so that it achieves a better decoding time. Similarly to the ℓ_∞/ℓ_2 case, we first find a set S of size $\mathcal{O}(\frac{k}{\epsilon})$ containing all $\frac{\epsilon}{k}$-heavy hitters, point-query every $i \in S$ and then keep the largest $\mathcal{O}(k)$ coordinates. As before, our goal is to find the relative phases among the coordinates in S. For the real-signal algorithm in [Nak17a], phases degenerate to signs, and the goal becomes what is called a Sign Prediction Problem, which is solved via a careful reduction to the stochastic block model on a graph of t nodes with failure probability $o(1)$. The failure probability has been improved to $1/\operatorname{poly}(t)$ in [TMB+17], and this polynomially small failure probability, as we shall see later, will be critical in attaining an $\mathcal{O}(k \log n)$ measurement complexity while achieving an $\mathcal{O}(k\operatorname{poly}(\log n))$ decoding time.

As said above, the previous algorithm in [Nak17a] essentially reduces the problem of inferring the relative signs on set S to a sign prediction problem, which we now extend to complex signals as the Phase Prediction Problem. In order for this type of reduction to work, the algorithm employed a pruning procedure on S to obtain a subset $T \subseteq S$ such that the following three conditions hold: (a) finding the relative phases in T still gives the ℓ_2/ℓ_2 error guarantee; (b) for every $i \in T$, $|x_i|$ is “large” enough; (c) sampling a pair from T is “fast” enough. We adopt the same pruning but do not immediately reduce to the Phase Prediction Problem. Instead, we hash all n coordinates to $B = \mathcal{O}(\frac{|T|}{\log|T|})$ buckets and solve the Phase Prediction problem in each bucket separately using $\mathcal{O}(\log^2|T|)$ measurements. Invoking the Chernoff bound and the Bernstein’s inequality, we see that the conditions (a),

(b) and (c) will hold simultaneously in each bucket with high probability, which allows for a union bound over all buckets. The low failure probability of the Phase Prediction Problem also guarantees that the algorithm will succeed in all buckets. The remaining piece is to combine the relative phases across buckets. However, we cannot run again the algorithm for the Phase Prediction Problem directly on B buckets because it would not give a runtime linear in k. Observe that we can afford an additional $\mathcal{O}(\log |T|)$ factor in the number of measurements, it is possible to obtain a time linear in k as follows. We create a graph on B vertices (corresponding to the B buckets) with $\mathcal{O}(|T|)$ random edges, and thus the graph is connected except with probability at least $\frac{1}{\operatorname{poly}(|T|)}$. Each edge consists of $\mathcal{O}(\log |T|)$ estimates of the relative phase between two vertices (buckets), which drives the failure probability down to $\frac{1}{\operatorname{poly}(|T|)}$. The algorithm for finding the relative phases among buckets is now a simple Depth First Search, running in linear time of the graph size. At the end we output $\widehat{x}$ supported on T with the relative phases found.

To recover the relative phases within and across buckets, we downsample the coordinates to obtain a subsignal consisting of exactly two coordinates in T so that we can infer their relative phases. We repeat this process for sufficiently many times so that we can recover the relative phases among different pairs of coordinates in T and obtain a global picture of relative phases for all coordinates in T. Therefore the downsampling rates have to be carefully chosen in order not to blow up the number of measurements while achieving an overall failure probability of $\frac{1}{\operatorname{poly}(n)}$. We also note that our algorithm is non-adaptive and T is unknown before the execution, thus we concatenate the sensing matrices of carefully chosen sizes for each possible value of $\lceil \log |T| \rceil = 1, 2, \ldots, \Theta(\log k)$.

Algorithm and Sensing Matrix

We present the algorithm in Algorithm 5, and describe the sensing matrices of the subroutines ComputeApprox, SignedEdgesPrediction and CombineBuckets below. We note that the absolute constants involved in the algorithms can be determined constructively as we unfold our analysis, though we do not attempt to optimize the constants in the analysis.

Algorithm 5 An algorithm for the ℓ_2/ℓ_2 sparse recovery. The absolute constants C, C', C_0, C_2, etc., are constructive from the analysis.

```
1: $\widetilde{S} \leftarrow$ the set returned by the algorithm in Theorem 2.1.11 with $K = Ck/\epsilon$
2: Estimate $|\widehat{x}_i|$ for all $i \in \widetilde{S}$ using Count-Sketch with $K = C'k/\epsilon$
3: $S \leftarrow$ index set of the largest $C_2 k$ coordinates among $\{|\widehat{x}_i|\}_{i\in\widetilde{S}}$
4: for $t \in [C_2 k]$ do
5:     $L_t \leftarrow$ ComputeApprox$(x, t)$
6: end for
7: $T \leftarrow$ Prune$(x, S, \{L_t\})$
8: $B \leftarrow \lceil \frac{|T|}{c \log |T|} \rceil$
9: $l \leftarrow \lceil \log |T| \rceil$
10: $\Delta \leftarrow \lceil \frac{\log k}{4 \log |T|} \rceil \log_k(\frac{2}{\delta})$
11: for $r \in [\Delta]$ do
12:     Pick $O(k)$-wise independent hash function $h_r : [n] \to [B]$
13:     for $j \in [B]$ do
14:         Signs$_j \leftarrow$ RelPhasesInBucket$_{r,l}(x_{h_r^{-1}(j)}, h_r^{-1}(j) \cap T)$
15:     end for
16:     Signs$_B \leftarrow$ CombineBuckets$_{r,l}(|\widehat{x}|, T, h_r, \{L_t\})$
17:     Signs$_r \leftarrow$ relative phases on $T$ inferred from $\{\text{Signs}_j\}_{j\in[B]}$ and Signs$_B$
18: end for
19: Keep the most frequent pattern among $\{\text{Signs}_r\}_{r\in[\Delta]}$
20: Output $\widehat{x}_T$ with the relative phases inferred
```

- ComputeApprox(x, t, S): The sensing matrix has $C_2 k$ layers. The t-th layer has $\Theta(\log n)$ independent rows of the form

$$\begin{pmatrix} \delta_1 g_1 & \delta_2 g_2 & \cdots & \delta_n g_n \end{pmatrix},$$

  where g_i are i.i.d. $\mathcal{N}(0,1)$ variables and δ_i are i.i.d. Bernoulli random variables such that $\mathbb{E}\delta_i = 1/(C_L t)$, where C_L is an absolute constant.

- RelPhasesInBuckets$_{r,l}$: The sensing matrix has $\rho_{r,l}$ independent rows, where the q-th row is given by

$$\begin{pmatrix} \delta_{q,1} g_{q,1} & \delta_{q,2} \sigma_{q,2} & \cdots & \delta_{q,n} \sigma_{q,n} \end{pmatrix},$$

  and

$$\rho_{r,l} = \Theta\left(\frac{1}{\epsilon^2 \eta^2} l^2 (\log(C_2 k) - l + 2)^4\right).$$

  In the above, $\{\delta_{q,i}\}$ are i.i.d. Bernoulli variables with $\mathbf{E}\,\delta_{q,i} = \frac{\epsilon\eta^2}{C_B l(\log(C_2 k) - l + 2)^2}$ for some

Algorithm 6 Prune Algorithm, which, given a vector $\widehat{x}$, a set S and a sequence of thresholds $\{L_t\}$, outputs a pruned set T

```
1: function PRUNE($\widehat{x}, S, \{L_t\}$)
2:     $\{z_i\}_{i\in[|S|]} \leftarrow \{|\widehat{x}_i|\}_{i\in S}$
3:     Sort all $z_i$ in decreasing order
4:     Find maximum $m \in [|S|]$ such that $|z_m|_2^2 > \frac{\epsilon}{C_0(\log(C_2k)-l_0+2)^2}L_m$ (where $2^{l_0-1} < m \leq 2^{l_0}$)
5:     $T \leftarrow \{i \in S : |\widehat{x}_i| \geq z_m\}$
6:     return $T$
7: end function
```

constant C_B large enough and $\{\sigma_{q,i}\}$ are i.i.d. random signs, and the constant inside the Θ-notation for $\rho_{r,l}$ depends on C_B and the absolute constant c_{SP} in Theorem 2.1.13.

- CombineBuckets$_{r,l}$: The sensing matrix has $(C_{\text{noise}} + C_{\text{phase}})lQ_l$ rows, divided into Q_l layers of $(C_{\text{noise}} + C_{\text{phase}})l$ rows each, where C_{noise} and C_{phase} are absolute constants, and

$$Q_l = \frac{C_Q}{\epsilon^2\eta^2}2^l(\log(C_2k) - l + 2)^4$$

for some absolute constant C_Q.

For each $q = 1, \ldots, Q_l$ we pick a random vector $(\delta_{q,1}, \delta_{q,2}, \ldots, \delta_{q,n})$ of i.i.d. Bernoulli coordinates such that $\mathbf{E}\,\delta_{q,i} = \Theta(\epsilon\eta^2 2^{-l}(\log(C_2k) - l + 2)^{-2})$. Each layer consists of two sublayers, a noise estimation layer of $C_{\text{noise}}l$ rows and a relative phase estimation layer of $C_{\text{phase}}l$ rows. The j-th row ($j = 1, \ldots, C_{\text{noise}}l$) in the noise estimation layer is

$$\begin{pmatrix} \delta_{q,1}\xi_{q,j,1}g_{q,j,1} & \delta_{q,2}\xi_{q,j,2}g_{q,j,2} & \cdots & \delta_{q,n}\xi_{q,j,n}g_{q,j,n} \end{pmatrix},$$

where $\{\xi_{q,j,i}\}$ are i.i.d. Bernoulli variables such that $\mathbf{E}\,\xi_{q,j,i} = 1/2$ and $\{g_{q,j,i}\}$ are i.i.d. standard normal variables. The j-th row ($j = 1, \ldots, C_{\text{phase}}l$) in the phase estimation layer is

$$\begin{pmatrix} \delta_{q,1}\sigma_{q,j,1} & \delta_{q,2}\sigma_{q,j,2} & \cdots & \delta_{q,n}\sigma_{q,j,n} \end{pmatrix},$$

where $\{\sigma_{q,j,i}\}$ are i.i.d. Bernoulli variables such that $\mathbf{E}\,\sigma_{q,j,i} = 1/C''$ for some absolute constant C''.

Algorithm 7 CombineBuckets Algorithm. The absolute constants $C_0, C_1, C_2, C_1', C_2'', C''$, etc., are constructive from the analysis.

```
1: function CombineBuckets$_{r,l}(|\widehat{x}|, T, h_r, \{L_t\})$
2:     $Q_{\text{good}} \leftarrow \{q \in [Q_l] : |\{i \in T : \delta_{q,i} = 1\}| = |h_r(\{i \in T : \delta_{q,i} = 1\})| = 2\}$
3:     for each $q$ in $Q_{\text{good}}$ do
4:         $\{u_q, v_q\} \leftarrow \{i \in T : \delta_{q,i} = 1\}$
5:     end for
6:     Trim $Q_{\text{good}}$ by removing the $q$'s with duplicate $\{u_q, v_q\}$ pairs
7:     $G_B \leftarrow$ empty graph on $h_r(T)$
8:     for each $q$ in $Q_{\text{good}}$ do
9:         $|w_1|, \ldots, |w_{C_{\text{noise}}l}| \leftarrow$ measurements from the noise estimation layer
10:        $J \leftarrow \{j \in [C_{\text{noise}}l] : \xi_{q,j,u_q} = \xi_{q,j,v_q} = 0\}$
11:        $L_q' \leftarrow \text{median}_{j \in J} |w_j|^2$
12:        $L_{\text{thres}} \leftarrow \frac{C''\epsilon\eta^2}{4C_1'C_0(\log(C_2k)-l+2)^2} L_{|T|}$
13:        if $L_q' \geq L_{\text{thres}}$ then
14:            $|z_1|, \ldots, |z_{C_{\text{phase}}l}| \leftarrow$ measurements from the phase estimation layer
15:            $J_{\text{good}} \leftarrow \{j \in [C_{\text{phase}}l] : \sigma_{q,j,u_q} = \sigma_{q,j,v_q} = 1\}$
16:            for each $j \in J_{\text{good}}$ do
17:                $\theta_j \leftarrow$ relative phase between $x_{u_q}$ and $x_{v_q}$ by applying Relative Phase Test
                       (Lemma 2.1.14) to $|\widehat{x}_{u_q}|$, $|\widehat{x}_{v_q}|$ and $|z_j|$
18:                Round $\theta_j$ to the nearest phase in $P$
19:            end for
20:            Add an edge $(h_r(u), h_r(v))$ to $G_B$ with label being the most frequent
   $\{\theta_j\}_{j \in [J_{\text{good}}]}$
21:        end if
22:    end for
23:    $\text{Sign}_B \leftarrow$ the relative phases among all $j \in h_r(T)$ collected by a depth first search on
   $G_B$
24:    return $\text{Sign}_B$
25: end function
```

Next we describe how the algorithms ComputeApprox, SignedEdgesPrediction, CombineBuckets operate.

- ComputeApprox(x, t): Suppose that the measurements in the t-th layer are $y_1, \ldots, y_{\Theta(\log n)}$. Return $L = \text{median}_q\, y_q^2$.

- RelPhasesInBucket$_{r,l}(z, T)$: For notational convenience, let us drop the subscripts r in this paragraph and call a $q \in [\rho_l]$ good if $|\{i \in T : \delta_{q,i} = 1\}| = 2$. For each good q, let $\{u, v\} = \{i \in T : \delta_{q,i} = 0\}$ and run the Relative Phase Test (Lemma 2.1.14) to find an estimate of the relative phase between z_u and z_v. Recall that P is equidistant with gap at least η, we can correct the estimate up to an additive error of $\eta/2$, and therefore we obtain the relative phase between z_u and z_v with probability at least $2/3$. We split all good q's into groups of size $c_{\text{SP}} l \lceil \log l \rceil$, where c_{SP} is the constant from Theorem 2.1.13. For each such group, we build a graph, called a working graph, on vertex set T with the edge set and labels defined by the pairs recovered from the corresponding q's in the group, and solve the Phase Prediction Problem (Theorem 2.1.13) to find a pattern of relative phases on T. We then return the most frequent pattern across all groups.

- CombineBuckets$_{r,l}(|\widehat{x}|, T, h_r, \{L_t\})$: The pseudocode is presented in Algorithm 7. The following is an intuitive description.

  We call a $q \in [Q_l]$ good if $|\{i \in T : \delta_{q,i} = 1\}| = |h_r(\{i \in T : \delta_{q,i} = 1\})| = 2$, that is, there are exactly two indices $\{u_q, v_q\}$ in T which are subsampled and hashed to different buckets. Retain the good q's with distinct $\{u_q, v_q\}$ pairs only. For each of the retained good q's, we first check whether the noise in the subsampled signal is too large (Lines 9 to 13). Each $|w_j|^2$ for $j \in J$ is an estimate of the noise energy, and their median L'_q is supposed to be a good estimate. If L'_q is bigger than some threshold L_{thres}, we reject that q; otherwise, we accept the q and proceed to estimate the relative phase between x_{u_q} and x_{v_q}. For each measurement from the phase estimation layer, we run the Relative Phase Test (Lemma 2.1.14) to find $C_{\text{phase}} l$ estimates of the relative phase between x_{u_q} and x_{v_q}, and keep the most frequent estimate of the relative phase as the relative phase estimate between the bucket pair $\{h_r(u_q), h_r(v_q)\}$. We build a graph G_B on the vertex set $h_r(T)$ with the edge set and labels defined by the accepted $\{h_r(u_q), h_r(v_q)\}$ pairs. By traversing G_B with a depth first search, we can collect the estimates of the relative phases among all $j \in h_r(T)$, whenever G_B is connected.

2.1.6 Analysis

We start with the total number of measurements and runtime.

Lemma 2.1.23. *The total number of measurements is $\mathcal{O}(\epsilon^{-2}\eta^{-2}k\log n)$.*

Proof. It is straightforward that the number of measurements for both the heavy hitter algorithm in Theorem 2.1.11 and CountSketch is $\mathcal{O}(\epsilon^{-1}k\log(n/\delta))$. The number of measurements for ComputeApprox is $\mathcal{O}(k\log n)$.

Since we need to stack the sensing matrices of RelSignsBucket and CombineBuckets for $l = 1, 2, \ldots, \log k$, the total number of measurements for RelSignsBucket is upper bounded by (up to a constant factor)

$$\begin{aligned}
&\sum_{l=1}^{\log(C_2k)} \sum_{r=1}^{\lceil \frac{\log k}{4l} \rceil \log_k n} \sum_{j=1}^{\frac{2^l}{cl}} \frac{1}{\epsilon^2\eta^2} l^2(\log(C_2k) - l + 2)^4 \\
&= \frac{1}{\epsilon^2\eta^2}\log_k n \sum_{l=1}^{\log(C_2k)} \frac{2^l}{cl}\left\lceil \frac{\log k}{4l} \right\rceil l^2(\log(C_2k) - l + 2)^4 \\
&\lesssim \frac{1}{\epsilon^2\eta^2}\log_k n \left(\sum_{l > \frac{1}{4}\log k} 2^l l(\log(C_2k) - l + 2)^4 + \sum_{l \le \frac{1}{4}\log k} 2^l \log k(\log(C_2k) - l + 2)^4 \right).
\end{aligned}$$

The first term in the bracket can be bounded as

$$\begin{aligned}
\sum_{l > \frac{1}{4}\log k} 2^l l(\log(C_2k) - l + 2)^4 &\le \sum_{u=\log 5+2}^{\log(C_2k^{3/4})+2} 2^{\log(C_2k)-u+2}(\log(C_2k) - u + 2)u^4 \\
&\le 20k\log(C_2k)\sum_{u=0}^{\infty}\frac{u^4}{2^u} \\
&\lesssim k\log k,
\end{aligned}$$

and the second term as

$$\sum_{l \le \frac{1}{4}\log k} 2^l \log k(\log(C_2k) - l + 2)^4 \lesssim k^{\frac{1}{4}}\log^6 k.$$

It follows that the number of measurements needed for RelSignsBucket is $\mathcal{O}(\epsilon^{-2}k\log n)$.

The total number of measurements for CombineBuckets are (constants are suppressed):

$$\sum_{l=1}^{\log k}\left(\left\lceil\frac{\log k}{4l}\right\rceil \log_k n\right)\left(\frac{1}{\epsilon^2\eta^2}2^l(\log(C_2k)-l+2)^4\cdot l\right)$$
$$\leq \frac{1}{\epsilon^2\eta^2}\log_k n\left(\sum_{l\geq\frac{1}{4}\log k} l2^l(\log(C_2k)-l+2)^4+\sum_{l<\frac{1}{4}\log k}2^l l^2(\log(C_2k)-l+2)^4\right)$$

The two terms in the bracket can be bounded similarly as before, giving $\mathcal{O}(\epsilon^{-2}\eta^{-2}k\log n)$ measurements for CombineBuckets in total.

Therefore, the total number of measurements used by the algorithm overall is $\mathcal{O}(\epsilon^{-2}\eta^{-2}k\log n)$, as desired. □

Lemma 2.1.24. *The decoding time of the algorithm is $\mathcal{O}(\epsilon^{-2}k\operatorname{poly}(\log(n/\delta)))$.*

Proof. In Algorithm 5, Line 1 takes $\mathcal{O}((k/\epsilon)\operatorname{poly}(\log(n/\delta))$ times, Line 2 $\mathcal{O}((k/\epsilon)\log n)$ time and Line 3 $\mathcal{O}(k/\epsilon)$ time. Each call to routine ComputeApprox routine takes $\mathcal{O}(\log n)$ time and thus Lines 4–6 take $\mathcal{O}(k\log n)$ time. The routine Prune takes $\mathcal{O}(|S|\log|S|) = \mathcal{O}(k\log k)$ time owing to sorting. Hence Lines 1–10 takes $\mathcal{O}((k/\epsilon)\operatorname{poly}(\log(n/\delta))$ time in total.

Next we examine RelSignsInBuckets. We shall see later in the analysis (Lemma 2.1.28) that we can discard the repetition r in which hashing results in some bucket having more than $K_2\log|T|$ elements from T, where K_2 is an absolute constant. We can compute $h_r(i)$ for all $i\in T$ in time $\mathcal{O}(|T|\operatorname{poly}(\log k))$ using multi-point polynomial evaluation, and thus in time $\mathcal{O}(|T|+|B|)=\mathcal{O}(T)$ count the number of elements of T in each bucket. Hence we may assume that each bucket contains at most $K_2\log|T|$ elements from T in each call to RelSignsInBuckets, which will run in $\mathcal{O}(\rho_{r,l}\operatorname{poly}(\log|T|)) = \mathcal{O}(\frac{1}{\epsilon^2\eta^2}\operatorname{poly}(\log k))$ time. The total contribution of RelSignsInBuckets to the recovery algorithm is thus

$$\mathcal{O}\left(\Delta\cdot B\cdot\frac{1}{\epsilon^2\eta^2}\operatorname{poly}(\log k)\right)=\mathcal{O}\left(\frac{1}{\epsilon^2\eta^2}|T|\operatorname{poly}\left(\log\frac{k}{\delta}\right)\right).$$

In CombineBuckets (Algorithm 7), Lines 2–5 take time $\mathcal{O}(\frac{|T|}{\epsilon}\operatorname{poly}(\log k))$, provided that the sensing matrix is stored as n lists of nonzero entries in each column (cf. Remark 2.1.20). We can use the phase estimation sublayer to determine whether $\delta_{q,i}=0$; note that we may lose a good row q if $\sigma_{q,j,u_q} = 0$ for all $j \in [C_{\text{phase}} l]$, but this happens with $1/\operatorname{poly}(|T|)$ probability and counts in the failure probability of the algorithm. The trimming step in Line 6 can be implemented by ignoring q if (u_q, v_q) has already been added to the graph G_B, which will be stored using the adjacency list representation. The body of the loop from Line 9 to Line 13 takes $\mathcal{O}(l) = \mathcal{O}(\log k)$ time, and repeating $|Q_{\text{good}}| = \mathcal{O}(\frac{|T|}{\epsilon^2\eta^2} \operatorname{poly}(\log k))$ times takes $\mathcal{O}(\frac{1}{\epsilon^2\eta^2}|T| \operatorname{poly}(\log k))$ time in total. Since we can stop adding new edges to G_B after adding $\kappa|T|$ edges, Line 23 runs in time $\mathcal{O}(|T|)$. The overall runtime of a call to CombineBuckets is thus $\mathcal{O}(\frac{1}{\epsilon^2\eta^2}|T| \operatorname{poly}(\log k))$, and the total contribution of CombineBuckets to the decoding time is

$$\mathcal{O}\left(\Delta \cdot \frac{|T|}{\epsilon^2\eta^2} \operatorname{poly}(\log k)\right) = \mathcal{O}\left(\frac{1}{\epsilon^2\eta^2}|T| \operatorname{poly}\left(\log \frac{k}{\delta}\right)\right).$$

Line 19 of Algorithm 5 runs in time $\mathcal{O}(\Delta \cdot |T| \cdot (1/\eta)) = \mathcal{O}((|T|/\eta)\log(1/\delta))$. The total decoding time follows immediately. □

Now we start proving the correctness of Algorithm 5. First we have the following lemma for Steps 1–3.

Lemma 2.1.25. *With probability $1 - \delta/2$, it holds that*

(i) $||\widehat{x}_i| - |x_i||^2 \le \frac{\epsilon}{2C_2 k}\|x_{-C_2 k}\|_2^2$ *for all $i \in [n]$; and*

(ii) $\|x_S - x\|_2^2 \le (1 + 0.9\epsilon)\|x_{-k}\|_2^2$.

Proof. Part (i) is the classical Count-Sketch guarantee (Theorem 2.1.10). Part (ii) is similar to the proof of [PW11, Theorem 3.1] but we present the proof below for completeness.

Let $H_1 = (\widetilde{S} \setminus S) \cap H_k(x)$, $H_2 = (\widetilde{S}^c) \cap H_k(x)$ and $I = S \setminus H_k(x)$. It is clear that

$$\|x_S - x\|_2^2 = \|x_{-k}\|_2^2 + \|x_{H_1}\|_2^2 + \|x_{H_2}\|_2^2 - \|x_I\|_2^2. \tag{2.6}$$

Since each $i \in H_1$ is displaced by $i' \in I$, we have that

$$|x_i| - \delta \le |\widehat{x}_i| \le |\widehat{x}_{i'}| \le |x_{i'}| + \delta, \quad \forall i \in H_1,\ i' \in I,$$

where $\delta = \sqrt{\frac{\epsilon}{2C_2 k}}\|x_{-C_2 k}\|_2$ is the estimation error from Count-Sketch. Let $a = \max_{i \in H_1} |x_i|$,

$b = \min_{i \in I} |x_i|$, then $a \le b + 2\delta$ and

$$\begin{aligned}
\|x_{H_1}\|_2^2 - \|x_I\|_2^2 &\le ka^2 - (C_2 - 1)kb^2 \\
&\le k(b+2\delta)^2 - (C_2-1)kb^2 \\
&= -(C_2-2)kb^2 + (4k\delta)b + 4k\delta^2 \\
&\le \frac{C_2+2}{C_2-2}k\delta^2 \\
&\le \epsilon\frac{C_2+2}{2(C_2-2)}\|x_{-C_2k}\|_2^2
\end{aligned} \tag{2.7}$$

On the other hand, by the guarantee of Theorem 2.1.11, the set $\widetilde{S}$ contains all $\frac{1}{Ck}$-heavy hitters and thus

$$|x_i|^2 \le \frac{\epsilon}{Ck}\|x_{-Ck/\epsilon}\|_2^2, \quad \forall i \in H_2.$$

Hence

$$\|x_{H_2}\|_2^2 \le |H_2| \cdot \frac{\epsilon}{Ck}\|x_{-Ck/\epsilon}\|_2^2 \le \frac{\epsilon}{C}\|x_{-k}\|_2^2 \tag{2.8}$$

By choosing C and C_2 large enough, part (ii) follows immediately from (2.6), (2.7) and (2.8). □

In the rest of the analysis we condition on the events in the preceding lemma. Recall that we have access only to $|\widehat{x}_i|$ in our scenario.

Lemma 2.1.26 (ComputeApprox)**.** *With probability at least $1 - 1/\operatorname{poly}(n)$, the subroutine* ComputeApprox(x,t) *returns a number L which satisfies $\frac{1}{C_1 t}\|x_{-C_2t}\|_2^2 \le L \le \frac{1}{t}\|x_{-t}\|_2^2$, where C_1 and C_2 are absolute constants.*

Proof. The argument is similar to the proof of (2.4) in Section 2.1.4. For instance, one can take $C_L = 110$ and show that

$$\Pr\left\{\frac{1}{19747t}\|x_{-353t}\|_2^2 \le L \le \frac{1}{t}\|x_{-t}\|_2^2\right\} \ge 0.55.$$

Repeating $\Theta(\log n)$ times with a big enough constant in the Θ-notation, we can boost the success probability of the event above to $1 - 1/\operatorname{poly}(n)$. □

Lemma 2.1.27. *The subroutine* PRUNE$(x, S, \{L_t\})$ *returns a set* $T \subseteq S$ *such that the following conditions hold:*

- $\forall i \in T, |x_i| \geq \frac{\epsilon}{C_0(\log(C_2 k) - l_0 + 2)^2} L_{|T|}$, *where* l_0 *is such that* $2^{l_0 - 1} < |T| \leq 2^{l_0}$.
- $\|x_T - x\|_2^2 \leq (1+\epsilon)\|x_{-k}\|_2^2$.

Proof. The first bullet is immediate by the design of the algorithm. For the second bullet first observe that

$$\|x_T - x\|_2^2 = \|x_{S \setminus T}\|_2^2 + \|x_{S^c}\|_2^2.$$

The second term is bounded in part (ii) of Lemma 2.1.25 that $\|x_{S^c}\|_2^2 \leq (1+\epsilon)\|x_{-k}\|_2^2$.

For the first term, let l_m be such that $2^{l_m - 1} < m \leq 2^{l_m}$ and $C_m = (\log(C_2 k) - l_m + 2)^2$. Suppose that the coordinates of x are sorted in decreasing order in magnitude. We have for $|T| + 1 \leq m \leq \log(C_2 k)$ that

$$|\widehat{x}_m|^2 \leq \frac{\epsilon}{C_0 C_m} L_m \leq \frac{\epsilon}{C_0 m C_m} \|x_{-m}\|_2^2 \leq \frac{\epsilon}{C_0 m C_m} (|x_{m+1}|^2 + \cdots + |x_{C_2 k}|^2 + \|x_{-C_2 k}\|^2).$$

thus by the guarantee of COUNT-SKETCH,

$$|x_m|^2 \leq \frac{\epsilon}{C_0 m C_m} (|x_{m+1}|^2 + \cdots + |x_{C_2 k}|^2 + \|x_{-C_2 k}\|_2^2) + \frac{\epsilon}{2 C_2 k} \|x_{-C_2 k}\|_2^2.$$

One can inductively obtain that

$$\begin{aligned} \|x_{S \setminus T}\|_2^2 &= |x_{|T|+1}|^2 + \cdots + |x_{C_2 k}|_2^2 \\ &\leq \left[\frac{\epsilon}{C_0(|T|+1) C_{|T|+1}} \sum_{m=|T|+2}^{\log(C_2 k)} \left(1 + \frac{\epsilon}{C_0 m C_m}\right) + \frac{\epsilon}{2 C_2 k} \sum_{m=|T|+1}^{\log(C_2 k)} \left(1 + \frac{\epsilon}{C_0 m C_m}\right) \right] \|x_{-C_2 k}\|_2^2. \end{aligned}$$

Observe that

$$\sum_{m=|T|+1}^{\log(C_2 k)} \left(1 + \frac{\epsilon}{C_0 m C_m}\right) \leq \exp\left(\frac{\epsilon}{C_0} \sum_{m=|T|+1}^{\log(C_2 k)} \frac{1}{m C_m} \right)$$

and

$$\begin{aligned}
\sum_{m=|T|+1}^{\log(C_2k)} \frac{1}{mC_m} &\le (2^{l_0}-m)\frac{1}{2^{l_0-1}(\log(C_2k)-l_0+2)^2} + \sum_{l=l_0+1}^{\log(C_2k)} 2^l \frac{1}{2^{l-1}(\log(C_2k)-l+2)^2} \\
&\le 2\sum_{l=l_0}^{\log(C_2k)} \frac{1}{(\log(C_2k)-l+2)^2} \\
&\le 2\sum_{i\ge 1}\frac{1}{i^2} \le 4,
\end{aligned}$$

which implies that

$$\|x_{S\setminus T}\|_2^2 \le \epsilon \exp\left(\frac{e}{8C_0}+\frac{e}{2C_2}\right)\|x_{-C_2k}\|_2^2.$$

Taking C_0 and C_2 big enough completes the proof. □

The second part of the preceding lemma shows that if we can recover the phases of the coordinates in T exactly, the ℓ_2/ℓ_2 error guarantee will be satisfied. Next we shall argue that we can recover the phases of the coordinates in T exactly with probability at least $1-1/\operatorname{poly}(|T|)$ in each loop from Lines 12 to 17 in Algorithm 5. Assume that $|T|\ge 2$, since otherwise the algorithm is trivially correct with any guess of phase of the only coordinate in T.

Lemma 2.1.28. *It holds that*

$$\Pr\left\{K_1\log|T| \le |T\cap h_r^{-1}(j)| \le K_2\log|T| \text{ for all } j\in[B]\right\} \ge 1-\frac{1}{20|T|^4}.$$

Let $T' = T\cup H_{(C_2+1)|T|}(x)$. It similarly holds that

$$\Pr\left\{K_1'\log|T| \le |T'\cap h_r^{-1}(j)| \le K_2'\log|T| \text{ for all } j\in[B]\right\} \ge 1-\frac{1}{20|T|^4}.$$

Furthermore,

$$\Pr\left\{\left\|x_{h_r^{-1}(j)\setminus T'}\right\|_2^2 \le \frac{K_3\log|T|}{|T|}\|x_{-C_2|T|}\|_2^2 \text{ for all } j\in[B]\right\} \ge 1-\frac{1}{10|T|^4}.$$

In the above, K_1, K_2, K_3 are constants depending only on c and K_1', K_2' are constants depending only on c and C_2.

Proof. The first is a standard application of the Chernoff bound. We nonetheless present the proof for completeness. For $i \in [n]$ and $j \in [B]$, let $X_{i,j}$ be the indicator variable of the event $h(i) = j$. Then $\mathbb{E}X_{i,j} = \frac{c \log|T|}{|T|}$ and $\mathbf{E}\sum_{i\in T} X_{i,j} = c\log|T|$. Note that $X_{i,j}$ are negatively associated, thus Chernoff bound can be applied, which, with appropriate constants, yields that

$$\begin{aligned}\Pr\left\{K_1\log|T| \le |T\cap h^{-1}(j)| \le K_2\log|T|\right\} &= \Pr\left\{K_1\log|T| \le \sum_{i\in T} X_{i,j} \le K_2\log|T|\right\}\\ &\ge \frac{1}{20|T|^5}.\end{aligned}$$

This allows us to take a union bound over all $j \in [B]$.

The bound on $|T' \cap h_r^{-1}(j)|$ is similar, noting that now $\mathbf{E}\sum_{i\in T'} X_{i,j} = c\frac{|T'|}{|T|}\log|T| \in [c(C_2+1)\log|T|, c(C_2+2)\log|T|]$ and one can choose K_1' and K_2' to be linear in C_2.

Next we prove the last part of the lemma. We shall use Bernstein's inequality (Theorem 2.1.9). Fix $j \in B$ and consider the random variables $Z_{i,j}$, indexed by $([n]\setminus S)\times[B]$, defined as $Z_{i,j} = \mathbf{1}_{\{h(i)=j\}}|x_i|^2$. It is easy to see that

$$|x_i|^2 \le \frac{1}{|T|}\left\|x_{-C_2|T|}\right\|_2^2, \quad \forall i \notin T'. \tag{2.9}$$

Indeed, let $T'' = H_{(C_2+1)|T|}(x)\setminus H_{C_2|T|}(x)$, then

$$|T|\cdot|x_{(C_2+1)|T|}|^2 \le \|x_{T''}\|_2^2 \le \|x_{-C_2|T|}\|_2^2,$$

whence (2.9) follows immediately.

In order to apply the Bernstein's inequality, we need to bound the variance of $\sum_{i\notin T'} Z_{i,j}$, which we can do as

$$\begin{aligned}\mathbf{E}\left(\sum_{i\notin T'} Z_{i,j}\right)^2 - \left(\mathbf{E}\sum_{i\notin T'} Z_{i,j}\right)^2 = \left(\frac{1}{|B|} - \frac{1}{|B|^2}\right)\sum_{i\notin T'}|x_i|^4 &\le \frac{1}{|B|}\cdot\max_{i\notin T'}|x_i|^2\cdot\left\|x_{(T')^c}\right\|_2^2\\ &\le \frac{1}{|B|\cdot|T|}\left\|x_{-C_2|T|}\right\|_2^4,\end{aligned}$$

where the last inequality follows from (ii) in Lemma 2.1.27.

It then follows from the Bernstein's inequality (Theorem 2.1.9) and appropriate choices of constants that

$$\Pr\left\{\sum_{i\in T^c} Z_{i,j} \geq \frac{K_3 \log|T|}{|T|}\left\|x_{-C_2|T|}\right\|_2^2\right\} \leq \frac{1}{10|T|^5},$$

which again allows for taking a union bound over all $j \in [B]$. □

We continue with the proof of the guarantees of ComputeApprox, RelPhasesInBucket and CombineBuckets.

Lemma 2.1.29 (RelPhasesInBuckets)**.** *In Line 14 the invocation of* RelPhasesInBuckets$_{r,l}$ *finds the relative phases of coordinates $i \in h_r^{-1}(j) \cap T$ with probability* $1 - \frac{1}{10|T|^4}$.

Proof. Assume that the events in the preceding lemma all happen. Let $T' = T \cup H_{(C_2+1)|T|}(x)$ and we call a $q \in [\rho_{r,l}]$ super-good if $|T' \cap h_r^{-1}(j)| = 2$.

A $q \in [\rho_{r,l}]$ is good with probability at least

$$\binom{|T\cap h_r^{-1}(j)|}{2}\left(\frac{\epsilon}{C_{\text{B}} l(\log(C_2k)-l+2)^2}\right)^2\left(1-\frac{\epsilon}{C_{\text{B}} l(\log(C_2k)-l+2)^2}\right)^{|T\cap h_r^{-1}(j)|-2}$$
$$\geq \exp\left(-\frac{\epsilon K_2}{4C_{\text{B}}}\right)\frac{K_1^2\epsilon^2}{2C_{\text{B}}^2(\log(C_2k)-l+2)^4}.$$

Conditioned on the fact that q is good, it is super-good with probability

$$\left(1-\frac{\epsilon}{C_{\text{B}} l(\log(C_2k)-l+2)^2}\right)^{|(H_{(C_2+1)|T|}\setminus T)\cap h_r^{-1}(j)|} \geq \exp\left(-\frac{\epsilon}{4C_{\text{B}}}(K_2'-K_1)\right),$$

which can be made arbitrarily close to 1 by adjusting the constant C_{B}.

Since there are $\rho_{r,l}$ rows, choosing an appropriate hidden constant in $\rho_{r,l}$, we can guarantee that the expected number of good q is $4.68c_{\text{SP}}l^2$. Hence by a Chernoff bound, with probability at least $1-(0.1183)^{l^2} \geq 1 - \frac{1}{20|T|^4}$, there are $c_{\text{SP}}l^2$ measurements corresponding to good q's, and most of them are supergood. This implies that there are at least $\frac{c_{\text{SP}}l^2}{c_{\text{SP}}l\log l} = \frac{l}{\log l}$ working graphs. Moreover, the expected energy of noise in each good measurement equals

(omitting the subscript q)

$$\begin{aligned}
\mathbf{E}\left|\sum_{i\in h_r^{-1}(j)\setminus T}\delta_i\sigma_i x_i\right|^2 &= \frac{\epsilon\eta^2}{C_{\mathrm{B}} l(\log(C_2k)-l+2)^2}\left\|x_{h_r^{-1}(j)\setminus T}\right\|_2^2 \\
&= \frac{\epsilon\eta^2}{C_{\mathrm{B}} l(\log(C_2k)-l+2)^2}\left\|x_{h_r^{-1}(j)\setminus T'}\right\|_2^2 \\
&\le \frac{\epsilon\eta^2}{C_{\mathrm{B}} l(\log(C_2k)-l+2)^2}\frac{K_3\log|T|}{|T|}\left\|x_{-C_2|T|}\right\|_2^2 \\
&\le \frac{K_3}{C_{\mathrm{B}}}\eta^2\cdot\frac{\epsilon}{(\log(C_2k)-l+2)^2}\cdot\frac{1}{|T|}\|x_{-C_2|T|}\|_2^2 \\
&\le \frac{K_3C_1C_0}{C_{\mathrm{B}}}\eta^2\cdot\frac{\epsilon}{C_0(\log(C_2k)-l+2)^2}L_{|T|}.
\end{aligned}$$

We can choose an appropriate constant C_{B} that is big enough such that in each such measurement the Relative Phase Test succeeds, by Markov's inequality, with probability at least $\frac{2}{3}$. Along with the guarantees of the Phase Prediction Problem, it follows that using every working graph we can find the relative phases of the coordinates in $h_r^{-1}(j)\cap T$ with probability at least $1-\frac{1}{l^{8.33}}$. By a Chernoff bound, with probability at least $1-(\frac{1}{l^{8.33}})^{\frac{l}{\log l}} = 1-\frac{1}{2^{8.33l}} \ge 1-\frac{1}{20|T|^4}$, at least half of the working graphs predict the relative phases correctly.

The overall failure probability is at most $\frac{1}{20|T|^4}+\frac{1}{20|T|^4}=\frac{1}{10|T|^4}$. □

Lemma 2.1.30 (CombineBuckets)**.** *In Line 16 the subroutine* CombineBuckets$_{r,l}$ *finds the relative phases between $h_r(T)$ with probability at least $1-\frac{1}{10|T|^4}$.*

Proof. We call a $q\in[Q_l]$ good, and call a good q accepted if $L'_q \ge L_{\text{thres}}$. We shall also define (i) a notion called excellent q for the good q's; and (ii) an event $\mathcal{E}_0$ regarding a low noise magnitude in the Relative Phase Test for all accepted q's. Then our argument goes as follows. Consider the following events:

- $\mathcal{E}_1$: There are at least $\kappa 2^l$ excellent q's.
- $\mathcal{E}_2$: All excellent q's will be accepted.
- $\mathcal{E}_3(q)$: Given that q is accepted, its associated edge has the correct relative phase between $\{u_q, v_q\}$.

- $\mathcal{E}_4$: G_B is connected.

When $\mathcal{E}_1$ and $\mathcal{E}_2$ happen, there are at least $\kappa 2^l$ edges in the graph G_B. If $\mathcal{E}_3(q)$ happens for all accepted q's, all edges in the graph have correct labels, and the algorithm would return a correct answer whenever $\mathcal{E}_4$ happens. We claim that

$$\Pr_{\{\delta_{q,i}\}}(\mathcal{E}_0) \geq 1 - \frac{1}{20|T|^4}; \tag{2.10}$$

$$\Pr_{\{\delta_{q,i}\}}(\mathcal{E}_1) \geq 1 - \frac{1}{40|T|^4}; \tag{2.11}$$

$$\Pr_{\{\xi_{q,j,i}\},\{g_{q,j,i}\}}(\mathcal{E}_2|\mathcal{E}_0) = 1; \tag{2.12}$$

$$\Pr_{\{\sigma_{q,j,i}\}}(\mathcal{E}_3(q)|\mathcal{E}_0) \geq 1 - \frac{1}{120|T|^6} \text{ for each accepted } q. \tag{2.13}$$

Since the pair $\{u_q, v_q\}$ is uniformly random for a good q, it follows from Theorem 2.1.12 that

$$\Pr_{\{\delta_{q,i}\}}(\mathcal{E}_4) \geq 1 - \frac{1}{60|T|^4}.$$

Hence the overall failure probability, after taking a union bound, is at most $\frac{1}{10|T|^4}$ as desired.

Below we prove our claims. A $q \in [Q_l]$ is good with probability

$$\begin{aligned}
&\Pr\left\{|h(\{i \in T : \delta_{q,i} = 1\})| = 2 \middle| |\{i \in T : \delta_{q,i} = 1\}| = 2\right\} \Pr\left\{|\{i \in T : \delta_{q,i} = 1\}| = 2\right\} \\
&= \Omega(1) \cdot \Omega\left(\binom{|T|}{2}\left(\frac{\epsilon}{2^l(\log(C_2k) - l + 2)^2}\right)^2\right) \\
&= \Omega\left(\frac{\epsilon^2}{(\log(C_2k) - l + 2)^4}\right).
\end{aligned}$$

Similarly to the proof of the preceding lemma, a good q is super-good with probability that can be made arbitrarily close to 1.

For a good q, let $\{u_q, v_q\} = \{i \in T : \delta_{q,i} = 1\}$ and $\nu_q \in \mathbb{R}^n$ be the noise vector defined as

$$(\nu_q)_i = \begin{cases} \delta_{q,i} x_i, & i \notin T; \\ 0, & i \in T. \end{cases}$$

When q is supergood,

$$\mathop{\mathbf{E}}_{\{\delta_{q,i}\}} \|v_q\|_2^2 \simeq \frac{\epsilon\eta^2}{2^l(\log(C_2k)-l+2)^2}\|x_{T'^c}\|_2^2 \le \frac{\epsilon\eta^2}{|T|(\log(C_2k)-l+2)^2}\|x_{-C_2|T|}\|_2^2.$$

Recall that $L_{|T|} \ge \frac{1}{C_1|T|}\|x_{-C_2|T|}\|_2^2$. With an appropriate choice of the hidden constant (depending on C_0 and C_1) in the subsampling rate $\mathbf{E}\,\delta_{q,i}$, it follows from Markov's inequality that

$$\Pr_{\{\delta_{q,i}\}}\left\{\|v_q\|_2^2 \le \frac{C''\epsilon\eta^2}{4C_1'C_3'C_0(\log(C_2k)-l+2)^2}L_{|T|}\right\} \ge 0.95. \tag{2.14}$$

We call a q excellent if it is supergood and satisfies the event in (2.14). Unconditioning on goodness and supergoodness, we know that each q is excellent with probability $\Omega(\epsilon^2/(\log(C_2k)-l+2)^4)$, and the expected number of excellent q's among $[Q_l]$ is $\Omega(2^l)$. By an appropriate choice of the constant C_Q and a Chernoff bound, we see that with probability $1-\frac{1}{40\cdot 2^{4l}}$ there exist $\kappa 2^l$ excellent q's that correspond to different edges in the graph G_B, where κ is the constant in Theorem 2.1.12. This proves (2.11), which regards $\mathcal{E}_1$.

Next we consider $\mathcal{E}_2$. Note that $\Pr(j\in J)=1/4$, by a Chenorff bound and choosing $C_{\text{noise}}=1320$, we have that $|J|\ge 260\log|T|$ with probability at least $1-\frac{1}{40|T|^6}$. Taking a union bound over all good q's,

$$\Pr\{|J|\ge 260\log|T| \text{ for all good } q\} \ge 1-\frac{1}{40|T|^4}. \tag{2.15}$$

Recall that a good bucket is supergood with overwhelming probability. A similar argument to the proof of (2.4) in Section 2.1.4 gives that (for instance, $C_1'=316$, $C_2'=10$, $C_3'=3$)

$$\Pr\left\{\frac{1}{C_1'}\|(v_q)_{-C_2'}\|_2^2 \le |w_j|^2 \le C_3'\|v_q\|_2^2\right\} \ge 0.7.$$

By a Chernoff bound and a union bound over all good q's,

$$\Pr\left\{\frac{1}{C_1'}\|(v_q)_{-C_2'}\|_2^2 \le L_q' \le C_3'\|v_q\|_2^2 \text{ for all good } q \,\middle|\, |J|\ge 260\log|T| \text{ for all good } q\right\} \ge 1-\frac{1}{40|T|^4}. \tag{2.16}$$

Define the event $\mathcal{E}_0$ as

$$\mathcal{E}_0 = \left\{ \frac{1}{C_1'} \|(\nu_q)_{-C_2'}\|_2^2 \leq L_q' \leq C_3' \|\nu_q\|_2^2 \text{ for all good } q \right\},$$

then it follows from (2.15) and (2.16) that

$$\Pr(\mathcal{E}_0) \geq 1 - \frac{1}{40|T|^4} - \frac{1}{40|T|^4} = 1 - \frac{1}{20|T|^4},$$

which proves (2.10).

In the rest of the proof we condition on $\mathcal{E}_0$. When q is excellent, we have that

$$L_q' \leq C_3' \|\nu_q\|_2^2 \leq C_3' \cdot \frac{C'' \epsilon \eta^2}{4 C_1' C_3' C_0 (\log(C_2 k) - l + 2)^2} L_{|T|} = \frac{C'' \epsilon \eta^2}{4 C_1' C_0 (\log(C_2 k) - l + 2)^2} L_{|T|}$$

and q will be accepted. This proves (2.12), which regards $\mathcal{E}_2$.

As the last step, we consider $\mathcal{E}_3$. Suppose that q is accepted, then we have

$$\frac{1}{C_1'} \|(\nu_q)_{-C_2'}\|_2^2 \leq L_q' \leq \frac{C''}{C_1'} \cdot \frac{\epsilon \eta^2}{4 C_0 (\log(C_2 k) - l + 2)^2} L_{|T|},$$

that is,

$$\|(\nu_q)_{-C_2'}\|_2^2 \leq C'' \cdot \frac{\epsilon \eta^2}{4 C_0 (\log(C_2 k) - l + 2)^2} L_{|T|}.$$

Consider now one of those accepted q's and the associated $C_{\text{phase}} l$ estimates from the phase estimation layer. Define the following two conditions:

$$\begin{aligned} &(P_1) \qquad \sigma_{q,j,u_q} = \sigma_{q,j,v_q} = 1 \\ &(P_2) \qquad \sigma_{q,j,i} = 0 \text{ for all } i \in H_{C_2'}(v_q) \end{aligned}$$

Note that $\Pr(P_1) = (\frac{1}{C''})^2 =: 2\gamma$ and $\Pr(P_2) \geq e^{-C_2'/C''}$. Choosing $C'' = 45$ and $C_{\text{phase}} \geq \frac{1159}{\gamma}$ large enough and by two Chernoff bounds, we conclude that, with probability at least $1 - \frac{1}{120|T|^4}$, there are at least $\gamma C_{\text{phase}} l$ measurements satisfying (P_1) and at least 0.7 fraction of them satisfy (P_2). We shall focus on the measurements satisfying (P_1).

In each measurement that further satisfies (P_2), the expected noise energy

$$\mathop{\mathbf{E}}_{\{\sigma_{q,j,i}\}} \left(\sum_{i \notin T} \sigma_{q,j,i} \delta_{q,i} x_i \right)^2 = \frac{1}{C''} \|(\nu_q)_{-C_2'}\|_2^2 \leq \frac{\epsilon \eta^2}{4 C_0 (\log(C_2 k) - l + 2)^2} L_{|T|}.$$

By Markov's inequality, we can guarantee that each Relative Phase Test fails with probability at most $\frac{1}{4}$. Hence, by a standard Chernoff bound, at least $\frac{5}{7}$ of those tests will give the correct answer, and thus the majority of the $\gamma C_{\text{phase}} l$ tests will give the correct answer, with probability at least $1 - \frac{1}{120 \cdot 2^{6l}}$, provided that $C_{\text{phase}} \geq \frac{1292}{\gamma}$. This proves (2.13), which regards $\mathcal{E}_3$. □

Combining Lemmata 2.1.28, 2.1.29 and 2.1.30, we see that each loop from Lines 12 to 17 in Algorithm 5 finds the relative phases among $\{x_i\}_{i \in T}$ correctly with probability at least $1 - \frac{1}{\sqrt{2e}|T|^4}$. Since there are Δ repetitions, we can recover the relative phases with probability at least $1 - \delta/2$. Unconditioning on the events in Lemma 2.1.25, we see that the failure probability is at most δ. The proof of Theorem 2.1.8 is now complete.

2.2 One-Bit Compressed Sensing

2.2.1 Our Contribution

We study the non-uniform case under adversarial noise and give the first result that achieves sublinear decoding time and nearly optimal $\mathcal{O}(\delta^{-2}k + k \log n)$ measurements, where δ is the reconstruction error, k is the sparsity and n is the universe size. For clearness, this scheme allows reconstruction of a fixed $x \in \mathbb{R}^n$ and not of all $x \in \mathbb{R}^n$; we refer to this a non-uniform guarantee.

We compare with two previous schemes, which are the state of the art. The first scheme appears in [PV13b], which achieves $\delta^{-2}k \log(n/k)$ measurements and $\text{poly}(n)$ decoding time, while the other appears in [Nak17b] and achieves $\mathcal{O}(\delta^{-2}k + k \log(n/k)(\log k + \log \log n))$ measurements and $\text{poly}(k, \log n)$ decoding time. We mention that the aforementioned two works are incomparable, since they exchange measurements and decoding time. However, generalizing [Nak17b] and using the linking/clustering idea of [LNNT16] (which is closely related to list-recoverable codes), we are able to almost get the best of both worlds. Our scheme is strictly better the scheme of [PV13b] when $k \leq n^{1-\gamma}$, for any constant γ; we note that the exponent of k in our running time is the same as the exponent of n in the running time of the relevant scheme of [PV13b].

We note that [PV13b] discusses also uniform guarantees for the one-bit compressed sensing problem. Our result is non-uniform and thus incomparable with some of the results in that paper; the relevant parts from [PV13b] are Theorem 1.1 and subsection 3.1. It is important to note that the guarantee of our algorithm **cannot** be achieved in the uniform setting, even when linear measurement are allowed [CDD09] (i.e. we do not have access only to the sign of the measurement), thus a comparison is meaningful (and fair) only with a non-uniform algorithm.

2.2.2 Preliminaries and Notation

For a vector $x \in \mathbb{R}^n$ we define $H(x,k) = \{i \in [n] : |x_i|^2 \geq \frac{1}{k}\|x_{-k}\|_2^2\}$. If $i \in H(x,k)$, we will say that i is a $1/k$-heavy hitter of x. For a set S we define x_S to be the vector that occurs after zeroing out every $i \in [n] \setminus S$. We define $\mathrm{head}(k)$ to be the largest k in magnitude coordinates of x, breaking ties arbitrarily, and we define $x_{-k} = x_{[n]\setminus \mathrm{head}(k)}$, which we will also refer to as the tail of x. Let $\mathcal{S}^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$. For a number θ we set $\mathrm{sign}(\theta) = 1$ if $\theta \geq 0$, and -1 otherwise. For a vector $v = (v_1, v_2, \dots, v_n)$ we set $\mathrm{sign}(v) = (\mathrm{sign}(v_1), \mathrm{sign}(v_2), \dots, \mathrm{sign}(v_n))$. We also denote $\mathcal{P}([n])$ to be the powerset of $[n]$.

Definition 2.2.1 (Vertex Expander). *Let $\Gamma : [N] \times [D] \to [M]$ be a bipartite graph with N left vertices, M right vertices and left degree D. Then, the graph G will be called a (k,ζ) vertex expander if for all sets $S \subseteq [N], |S| \leq k$ it holds that $\Gamma(S) \geq (1-\zeta)|S|D$.*

2.2.3 Main Result

The main result of this subchapter is the following.

Theorem 2.2.2. *There exists a distribution $\mathcal{D} \in \mathbb{R}^{m \times n}$, a procedure* Dec $: \{-1,+1\}^m \to \mathbb{R}^n$ *and absolute constants $C_1, C_2 > 1$ such that*

$$\forall x \in \mathcal{S}^{n-1} : \mathbb{P}_{\Phi \sim \mathcal{D}}[\widehat{x} = \textsc{Dec}(\text{sign}(\Phi x)) : \|x - \widehat{x}\|_2^2 > 2\|x_{-k}\|_2^2 + \delta] \leq e^{-C_1 \delta^{-1} k} + n^{-C_2},$$

and $\|\widehat{x}\|_0 = O(k)$*.*

The number of rows of Φ *is* $m = \mathcal{O}(k \log n + \delta^{-2} k)$*, and the running time of* Dec *is* $\text{poly}(k, \log n)$*.*

It should clear that since $\widehat{x}$ is $O(k)$-sparse, we do not need to output an n-dimensional vector, but only the positions where the vector is non-zero.

2.2.4 Overview of our Approach

The one-bit compressed sensing framework has a neat geometrical representation: one can think of every measurement $\text{sign}(\langle \Phi_j, x \rangle)$ indicating on which side of the hyperplane Φ_j the vector x lies. One of the results of [PV13b] shows that this is possible with $\mathcal{O}(\delta^{-2} k \log(n/k))$ random hyperplanes when random post-measurement noise v is added, i.e. $y = \text{sign}(\Phi x + v)$; the paper gives also other, very intersesting results, but we will not focus on them in this work. To achieve sublinear decoding time we do not pick the hyperplanes (measurements) at random, but we construct a structured matrix that allows us to find all $1/k$-heavy hitters of x. This approach also has been followed in one of the schemes of [Nak17b]. There the author implemented the dyadic trick [CH09] in the one-bit model, showing that it is possible to recover the heavy hitters of x from one-bit measurements, using $O(k \log(n/k)(\log k + \log\log n))$ measurements. Our results is an extension and generalization of that paper, along with the linking and clustering technique of [LNNT16].

In the core of our scheme, lies the design of a randomized scheme which is analogous to the “partition heavy hitters” data structure of [LNNT16]; we call this scheme One-Bit PartitionPointQuery. More concretely, the question is the following: given a partition $\mathcal{P}$ of the universe $[n]$, is it possible to decide if a given set $S \in \mathcal{P}$ is heavy, when we are given access only to one-bit measurements? We answer this question in the affirmative and then combine this routine with the graph clustering technique of [LNNT16]. We thus show that, similarly to that paper, it is possible to reduce the problem of finding the heavy coordinates in the one-bit framework to the same clustering problem.

2.2.5 Toolkit

Lemma 2.2.3 (Chernoff Bound)**.** *Let $X_1, \ldots, X_r$ be Bernoulli random variables with $\mathbb{E}[X_i] = p$. There exists an absolute constant c_{ch} such that*

$$\mathbb{P}\left[|\sum_i X_i - pr| > \epsilon pr \right] \leq e^{-c_{ch}\epsilon^{-2}pr}$$

Lemma 2.2.4 (Bernstein's Inequality)**.** *There exists an absolute constant c_B such that for independent random variables $X_1, \ldots, X_r$, with $|X_i| \leq K$ we have that*

$$\forall \lambda > 0, \mathbb{P}\left[|\sum_i X_i - \mathbb{E}\sum_i X_i| > \lambda \right] \leq e^{-C_B\lambda/\sigma^2} + e^{-C_B\lambda/K},$$

where $\sigma^2 = \sum_i \mathbb{E}(X_i - \mathbb{E}X_i)^2$.

Theorem 2.2.5 (Fixed Signal, Random Noise Before Quantization [PV13b])**.** *Let $x \in \mathbb{R}^N$ and $G \in \mathbb{R}^{m \times N}$, each entry of which is a standard gaussian. If $y = \text{sign}(Gx + v)$, where $v \sim \mathcal{N}(0, \sigma^2 I)$, then the following program*

$$\widehat{x} = \text{argmax} \langle y, Gx \rangle, s.t. \ \|z\|_1 \leq \sqrt{k}$$

returns a vector $\widehat{x}$ such that $\|x - \widehat{x}\|_2^2 \leq \delta$, as long as

$$m = \Omega(\delta^{-2}(\sigma^2 + 1)k \log(N/k)).$$

2.3 Main Algorithm

Our algorithm proceeds by finding a set S of size $\mathcal{O}(k)$ containing all coordinates $i \in H(x,k)$ and then runs the algorithm of [PV13b], by restrictring on columns indexed by S. The scheme that is used to find the desired set S is guaranteed by the following Theorem.

Theorem 2.3.1. *There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m' \times n}$, a decoding procedure* OneBitHeavyHitters : $\{-1, 1\}^{m'} \to \mathcal{P}([n])$ *and an absolute constant c, such that $S =$* OneBitHeavyHitters(sign(Φx)) *satisfies the following, with probability $1 - \frac{1}{n^{C_1}}$. a) $|S| \leq ck$, and b) $\forall i \in H(x, k), i \in S$. Moreover, the number of rows of Φ equals $m' = \mathcal{O}(k \log n)$ and the running time of* OneBitHeavyHitters *is $\mathcal{O}(k \cdot \text{poly}(\log n))$.*

Given the above theorem we show how to prove the Theorem 1.

Proof. We vertically concatenate the matrix Φ from Theorem 2.3.1 and the matrix G guaranteed by Theorem 2.2.5. Then, we run the algorithm OneBitHeavyHitters(sign(Φx)) to obtain a set S. Then we run the following algorithm:

$$\widehat{x} = \text{argmax} \langle y, G_S z \rangle, s.t. \|z\|_1 \leq \sqrt{k}.$$

Last, we output $\widehat{x}$. Since $Gx = G_S x_S + G_{[n]\setminus S} x_{[n]\setminus S}$, and $G_{[n]\setminus S} x_{[n]\setminus S} \sim \mathcal{N}(0, \|x_{[n]\setminus S}\|_2^2 I)$ and $\|x_{[n]\setminus S}\|_2 \leq 1$, by combining the guarantees of theorems 2.2.5 and 2.3.1 we have that

$$\|x - \widehat{x}\|_2^2 = \|x_S - \widehat{x}_S\|_2^2 + \|x_{[n]\setminus S}\|_2^2 \leq \delta + 2\|x_{-k}\|_2^2,$$

because $\|x_{[n]\setminus S}\|_2^2 \leq \|x_{-k}\|_2^2 + \sum_{i \in \text{head}(k)\setminus H(x,k)} x_i^2 \leq \|x_{-k}\|_2^2 + k\frac{1}{k}\|x_{-k}\|_2^2 = 2\|x_{-k}\|_2^2$. □

Remark: From the discussion in this subsection, it should be clear than any algorithm that runs in linear time in n and has the same guarantees as as Theorem 2.2.5 immediatelly implies, by our reduction, an algorithm that achieves $\mathcal{O}(k\text{poly}(\log n))$ time. Thus, any subsequent improvement of that type over [PV13b] gives an improvement of our main result in a black-box way.

2.3.1 Reduction to small Sparsity

The following trick is also used in [LNNT16]. If $k = \Omega(\log n)$, we can hash every coordinate to $\Theta(k/\log n)$ buckets and show that it suffices to find the $\frac{1}{\log n}$-heavy hitters in every bucket separately. Here, we give a proof for completeness. First, we state the following lemma, which is proven in section 2.4.

Theorem 2.3.2. *Let C', C_0 be absolute constants and suppose that $k \leq C' \log n$. Then there exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m'' \times n}$ with $m'' = \mathcal{O}(k \log n)$ rows, such that given $y = \operatorname{sign}(\Phi x)$, we can find, with probability $1 - n^{-C_0}$ and in time $\mathcal{O}(\operatorname{poly}(\log n))$, a set S of size $\mathcal{O}(k)$ containing every $i \in H(x,k)$.*

Given this lemma, we show how to prove Theorem 2. This lemma is also present in [LNNT16], but, for completeness, we prove it again here.

Proof. If $k < C' \log n$, we run the algorithm guaranteed by the previous lemma. Otherwise, we pick a hash function $g : [n] \to [C''k/\log n]$ and for $j \in [C''k/\log n]$ we obtain set S_j using lemma. We then output the union of all these sets. Define $z = C''k/\log n$. We argue correctness.

For $j \in [C''k/\log n]$ we use the Chernoff Bound to obtain that

$$\mathbb{P}\left[|g^{-1}(j) \cap H(x,k)| \geq \log n\right] \leq e^{-C''' \log n}.$$

We will now invoke Bernstein's inequality for the random variables $\left\{X_i = \mathbf{1}_{g(i)=j}\right\}_{i \in [n] \setminus H(x,k)}$; for these variables we have $K < \frac{1}{k}\|x_{-k}\|_2^2$ and

$$\sigma^2 < \sum_{i \in [n] \setminus H(x,k)} x_i^4 (z^{-1} - z^{-2}) \leq \frac{k}{z}\|x_{-k}\|_2^4 \sum_{i \in [n] \setminus H(x,k)} x_i^2 = \frac{k}{z}\|x_{-k}\|_2^4$$

$$\mathbb{P}\left[| \sum_{i \in g^{-1}(j) \setminus H(x,k)} x_i^2 \geq \frac{\log k}{k}\|x_{-k}\|_2^2 \right] \leq e^{-C''' \log n}.$$

By a union-bound over all $2z = 2C''k/\log n$ events, 2 for every buckets $j \in [z]$, we get the proof of the lemma.

□

We now focus on proving Theorem 2.3.2.

2.3.2 One-BitPartitionPointQuery

In this section we prove the following Theorem, which is the main building block of our algorithm.

Theorem 2.3.3. *Let $x \in \mathbb{R}^n$ and a partition $\mathcal{P} = \{P_1, P_2, \dots, P_T\}$ of $[n]$. There exists an oblivious randomized construction of a matrix $Z \in \mathbb{R}^{m \times n}$ along with a procedure* One-BitPartitionPointQuery : $[T] \to \{0,1\}$*, where $m = \mathcal{O}(k \log(1/\delta))$, such that given $y = \operatorname{sign}(Zx)$ the following holds for $j^* \in [T]$.*

1. *If P_{j^*} contains a coordinate $i \in H(x,k)$, then* One-BitPartitionPointQuery$(j^*) = 1$ *with probability $1-\delta$.*

2. *If there exist at least ck indices such that $\|x_{P_j}\|_2 \geq \|x_{P_{j^*}}\|_2$, then* One-BitPartitionPointQuery$(j^*) = 0$ *with probability $1-\delta$.*

Moreover, The running time of is $\mathcal{O}(\log(1/\delta))$.

We describe the construction of the the matrix Z. We are going to describe the matrix as a set of linear measurements on the vector x. For $i \in [n], j \in [T], B \in [C_B k], \ell \in [3], r \in [C_3 \log(1/\delta)]$ we pick the following random variables:

1. fully independent hash functions $h_{r,\ell} : [T] \to [C_B k]$.

2. random signs $\sigma_{j,B,\ell,r}$. Intuitively, one can think of this random variable as the sign assigned to set P_j in bucket B of sub-iteration ℓ of iteration r.

3. normal random variables $g_{i,r}$. One can think of this random variable as the gaussian associated with i in iteration r.

Then, for every $B \in [C_B k], \ell \in [3], r \in [C_B \log(1/\delta)]$ we perform linear measurements

$$z_{B,\ell,r} = \sum_{j \in h_{r,\ell}^{-1}(B)} \sigma_{j,B,\ell,r} \sum_{i \in P_j} g_{i,r} x_i,$$

as well as measurements $-z_{B,\ell,r}$ (the reason why we need this will become clear later).

Of course we have access only to the sign of the measurement: $y_{B,\ell,r} = \text{sign}(z_{B,\ell,r})$. We slightly abuse notation here, as y is described as a 3-dimensional vector; it is straightforward to see how this vector can be mapped to a 1-dimensional vector.

We will make use of the following lemmata. The value C_B is a large enough constant, chosen in order for the analysis to work out. Before proceeding with the lemmas, we pick constants C_u, C_d such that

1. $\mathbb{P}_{Y\sim\mathcal{N}(0,1)}[|Y| < C_u] = \frac{19}{20}$.
2. $\mathbb{P}_{Y\sim\mathcal{N}(0,1)}[|Y| > C_d] = \frac{19}{20}$.

Lemma 2.3.4. *Fix $i^* \in H(x,k)$, j^* such that $i^* \in P_{j^*}$, as well as $r \in [C_3 \log(1/\delta)]$. We also set $B_\ell = h_{r,\ell}(j^*)$. Then, with probability at least $\frac{3}{5}$ we have that for all $\ell \in [3]$ either*

$$y_{B_\ell,\ell,r} = \sigma_{j^*,B_\ell,\ell,r} \text{ or } y_{B_\ell,\ell,r} = -\sigma_{j^*,B_\ell,\ell,r}.$$

Proof. For the need of the proof we define $\mathcal{B}_\ell^{-1} = h_{r,\ell}^{-1}(h_{r,\ell}(j^*))$. First, observe that for all $\ell \in [3]$ that the random variable

$$Y_\ell = \sum_{j\in\mathcal{B}_\ell^{-1}\setminus\{j^*\}} \sigma_{j,B_\ell,\ell,r} \sum_{i\in P_j} g_{i,r} x_i$$

is distributed as

$$\sqrt{\left(\sum_{j\in\mathcal{B}_l^{-1}\setminus\{j^*\}} \sum_{i\in P_j} x_i^2\right)} \cdot \mathcal{N}(0,1).$$

Observe that with probability at least $\frac{19}{20}$, $|Y_\ell|$ will be at most

$$C_u \sqrt{\sum_{j\in\mathcal{B}_\ell^{-1}\setminus\{j^*\}} \sum_{i\in P_j} x_i^2}.$$

Define

$$Z_\ell = \sum_{j\in\mathcal{B}_\ell^{-1}\setminus\{j^*\}} \sum_{i\in P_j} x_i^2.$$

Consider now the set $\mathcal{P}_{\text{bad}}$ of P_j, $j \in [T] \setminus \{j^*\}$ for which there exists $i \in H(x,k)$ such that $i \in P_j$. Since there are at most $2k$ elements in $\mathcal{P}_{\text{bad}}$, with probability at least $1 - \frac{2}{C_B}$

it holds that $\mathcal{B}_\ell^{-1} \cap \mathcal{P}_{\text{bad}} = \emptyset$. Let this event be $\mathcal{W}$. It is a standard calculation that $\mathbb{E}[Z_l|\mathcal{W}] \leq \frac{1}{C_B k}\|x_{-k}\|_2^2$. Invoking Markov's inequality one gets that Z_l is at most $\frac{20}{C_B k}\|x_{-k}\|_2^2$ with probability at least $\frac{19}{20}$. Putting everything together, this gives that

$$|Y_\ell| > C_u\sqrt{\frac{20}{C_B k}}\|x_{-k}\|_2$$

with probability $\frac{1}{20}$. The probability that there exist $l \in [3]$ such that $|Y_\ell| > C_u\sqrt{\frac{20}{C_B k}}\|x_{-k}\|_2$ is at most $\frac{3}{20}$. We now observe that the

$$|\sum_{i \in P_{j^*}} g_{i,r}x_i| \geq C_d\|x_{P_{j^*}}\|_2 \geq C_d\frac{1}{\sqrt{k}}\|x_{-k}\|_2$$

with probability at least $\frac{19}{20}$. The above discussion implies that with probability at least $\frac{15}{20}$ the quantity $|\sum_{i \in P_{j^*}} g_{i,r}x_i|$ is larger than $|Y_\ell|$, for all $l \in [3]$, if $C_d/\sqrt{k} > C_u\sqrt{\frac{20}{C_B k}}$. This means that, with probability at least $\frac{3}{4}$, the sign of $z_{B_\ell,\ell,r}$ will be determined by the sign of $\sigma_{j^*,B_\ell,\ell,r}\sum_{i \in P_{j^*}} g_{i,r}x_i$ for all $\ell \in [3]$. This implies that if $\sum_{i \in P_{j^*}} g_{i,r}x_i > 0$, we will get that $y_{B_\ell,\ell,r} = \sigma_{j^*,B_\ell,\ell,r}$. On the other hand, if $\sum_{i \in P_{j^*}} g_{i,r}x_i < 0$ then $y_{B_\ell,\ell,r} = -\sigma_{j^*,B_\ell,\ell,r}$. This gives the proof of the lemma.

□

Lemma 2.3.5. *Let j^* such that $\|x_{P_{j^*}}\|_2 > 0$. We also define $B_\ell = h_{r,\ell}(j^*)$. Assume that there exist at least ck indices j such that $\|x_{P_j}\|_2 \geq \|x_{P_{j^*}}\|_2$, for some absolute constant c. Then, with probability $\frac{3}{5}$, there exists indices $\ell_1, \ell_2 \in [3]$ such that*

$$y_{B_{\ell_1},\ell_1,r} = \sigma_{j^*,B_{\ell_1},\ell_1,r} \text{ and } y_{B_{\ell_2},\ell_2,r} = -\sigma_{j^*,B_{\ell_2},\ell_2,r}.$$

Proof. For the need of the proof we also define $\mathcal{B}_\ell^{-1} = h_{r,\ell}^{-1}(h_{r,\ell}(j^*))$. Fix $\ell \in [3]$. Let $\mathcal{P}_{\text{good}}$ be the set of indices $j \in [T]$ such that $\|x_{P_j}\|_2 \geq \|x_{P_{j^*}}\|_2$. Let the random variable Z_ℓ be defined as

$$Z_\ell = |\{j \in \mathcal{P}_{\text{good}} \setminus \{j^*\} : j \in \mathcal{B}_\ell^{-1}\}|.$$

Observe now that $\mathbb{E}[Z_\ell] = \frac{ck}{C_B k} = \frac{c}{C_B}$ and moreover Z_l is a sum of independent Bernoulli random variables with mean $\frac{1}{C_B k}$, hence a standard concetration bound gives that, for c large enough, Z_ℓ will be larger than $4C_d^2C_u^2$ with probability $\frac{19}{20}$. This implies that

$$\sum_{j\in\mathcal{B}_\ell^{-1}\setminus\{j^*\}} \|x_{P_j}\|_2^2 \geq 4C_d^2C_u^2\|x_{P_{j^*}}\|_2^2.$$

for all $\ell \in [3]$. This implies that, for any $\lambda \in \mathbb{R}$,

$$\mathbb{P}\left[|\sum_{j\in\mathcal{B}_\ell^{-1}\setminus\{j^*\}} \sigma_{j,B_\ell}\sum_{i\in P_j} g_{i,r}x_i| \geq \lambda\right] \geq$$
$$\mathbb{P}\left[2C_dC_u\|x_{P_{j^*}}\|_2 \cdot |\mathcal{N}(0,1)| \geq \lambda\right].$$

The above implies that

$$\mathbb{P}\left[|\sum_{j\in B_l\setminus\{j^*\}} \sigma_{j,B_\ell,\ell,r}\sum_{i\in P_j} g_{i,r}x_i| \geq 2C_u\|x_{P_{j^*}}\|_2\right] \geq \frac{19}{20}$$

and moreover

$$\mathbb{P}\left[|\sum_{i\in\mathcal{P}_{j^*}} g_{i,r}x_i| \leq C_u\|x_{P_{j^*}}\|_2\right] \geq \frac{19}{20},$$

which implies that with probability $\frac{17}{20}$ we have that

$$|\sum_{j\in\mathcal{B}_\ell^{-1}\setminus\{j^*\}} \sigma_{j,\mathcal{B}_\ell^{-1},l,r}\sum_{i\in P_j} g_{i,r}x_i| \leq 2|\sum_{i\in\mathcal{P}_{j^*}} g_{i,r}x_i|.$$

Observe now that $y_{B_\ell,\ell,r}$ is the same as the sign of

$\sum_{j\in B_\ell\setminus\{j^*\}} \sigma_{j,\mathcal{B}_\ell^{-1},l,r}\sum_{i\in P_j} g_{i,r}x_i$, which, because of the random signs, means that

$$\mathbb{P}\left[y_{B_\ell,\ell,r} = 1\right] = \frac{1}{2}.$$

Moreover, we get that $y_{B_\ell,\ell,r}$ and $\sigma_{j^*,B_\ell,\ell,r}$ are independent. Conditioned on the previous events, the probability that either

$$y_{B_\ell,\ell,r} = \sigma_{j^*,B_\ell,\ell,r}$$

for all $\ell \in [3]$, or

$$y_{B_\ell,\ell,r} = -\sigma_{j^*,B_\ell,\ell,r}$$

for all $\ell \in [3]$, is $\frac{2}{8}$. This gives the proof of the claim since $\frac{3}{20} + \frac{2}{8} \leq \frac{8}{20} = \frac{2}{5}$.

□

We are now ready to proceed with the proof of Theorem 2.3.3.

Proof. We iterate over all $r \in [C_3 \log(1/\delta)]$ and count the number of "good" repetitions: a repetition r is good if for all $\ell \in [3]$, $y_{h_{r,\ell}(j^*),\ell,r} = \sigma_{j,h_{r,\ell}(j^*),\ell,r}$ or $y_{h_{r,\ell}(j^*),\ell,r} = -\sigma_{j,h_{r,\ell}(j^*),\ell,r}$. We also check if there exists $l \in [3]$ such that $y_{hr,\ell(j^*),\ell,r} = 0$ by checking the values of $y_{hr,\ell(j^*),\ell,r} = 0$ and $-y_{hr,\ell(j^*),\ell,r} = 0$. If there exists no such ℓ and the number of good repetitions is at least $\lceil \frac{1}{2} C_3 \log(T/\delta) \rceil + 1$ we output 1, otherwise we output 0.

We proceed with the analysis. First of all, if there exists an $\ell \in [3]$ that satisfies $y_{hr,\ell(j^*),\ell,r} = 0$, this would mean that $\|x_{P_{j^*}}\| = 0$. Let us assume that this is not the case, otherwise we can ignore j^*. If $i^* \in H(x,k)$ belongs to P_{j^*}, for some j^*, using Lemma 2.3.4 the expected number of good iterations equals $(3/5)C_3 \log(1/\delta)$,and by a Chernoff Bound we get that at least $(2.6/3) \cdot (3/5)C_3 \log(1/\delta) = (2.6/5)C_3 \log(1/\delta)$ repetitions will be good with probability

$$1 - e^{-\Omega(\log(|T|/\delta))} \geq 1 - \delta,$$

for large enough C_3. In the same way, using Lemma 2.3.5 we can bound by δ the probability that a set P_{j^*}, for which there exist at least ck set P_j with $\|x_{P_j}\|_2 \geq \|x_{P_{j^*}}\|_2$, has more than $\lceil \frac{1}{2} C_3 \log(T/\delta) \rceil - 1$ good repetitions. This concludes the proof of the lema.

□

The following lemma is immediate by taking $\delta = T^{-C_0-1}$ and taking a union-bound over all $j \in [T]$.

Lemma 2.3.6 (ONE-BITPARTITIONCOUNTSKETCH). *Let $x \in \mathbb{R}^n$ and a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_T\}$ of $[n]$. There exists a randomized construction of a matrix $Z \in \mathbb{R}^{m \times n}$, such that given $y = \operatorname{sign}(Zx)$, we can find in time $\mathcal{O}(k \log T)$ a set S of size $\mathcal{O}(k)$ that satisfies contains every $j \in [T]$ for which there exists $i \in H_k(x) \cap P_j$. Moreover, the failure probability is T^{-C_0}.*

2.3.3 One-Bit b-tree

We now describe the scheme of One-Bit b-tree. The b-tree is a folkore data structure in streaming algorithms, first appearing in [CH09] in the case of vectors with positive coordinates. The version of the b-tree we are using here is more closely related in [LNNT16]. We remind the reader that the aforementioned papers treated the case where we have access to Φx and not only to $\text{sign}(\Phi x)$. Here, we describe it a sensing matrix associated with a decoding procedure, rather than a data structure. Given the b-tree, we can find elements $i \in H(x,k)$ and get an analog of Theorem 1; however, this would only give $1/\text{poly}(\log n)$ failure probability. Getting $1/\text{poly}(n)$ failure probability requires using the ExpanderSketch algorithm of [LNNT16]In fact, we can use the One-Bit b-tree to speed up the One-Bit ExpanderSketch decoding procedure, but since our overall scheme already has a polynomial dependence on k in the running time due to the application of Theorem, this will not give us any crucial improvement. However, we believe that it might of independent interest in the sparse recovery community.

The following lemma holds.

Lemma 2.3.7. *Let $k, b < n$ be integers. There exists a randomized construction of a matrix $A \in \mathbb{R}^{M \times n}$ such that given $y = \text{sign}(Ax)$ we can find a set S of size $\mathcal{O}(k)$ such that $\forall i \in H(x,k), i \in S$. The total number of measurements equals*

$$M = \mathcal{O}(k\frac{\log(n/k)}{\log b}(\log(k/\delta) + \log\log(n/k) - \log\log b))$$

the decoding time is

$$\mathcal{O}(bk\frac{\log(n/k)}{\log b}(\log(k/\delta) + \log\log(n/k) - \log\log b))$$

and the failure probability is δ.

Proof. Let R be the smallest integer such that $kb^R \geq n$; this means that $R = \lceil \log(n/k)/\log b \rceil$. For $r = 0, \dots, R$ we use the One-Bit PartitionCountSketch scheme guaranteed by Lemma, with $\delta = \delta/(bkR)$ and partition $\mathcal{P}_r = \{\{1, \dots, \lceil \frac{n}{kb^r} \rceil\}, \{\lceil \frac{n}{kb^r} \rceil + 1, \dots, 2\lceil \frac{n}{kb^2} \rceil\}, \dots\}$, of size

$T_r = \Theta(kb^r)$.

The total number of measurements equals

$$\mathcal{O}(Rk\log(bkR/\delta)) = \mathcal{O}(k\frac{\log(n/k)}{\log b}(\log(k/\delta) + \log\log(n/k) - \log\log b)).$$

We can think of the partitions $T_1, T_2, \ldots, T_R$ as the levels of a b-ary tree; for every set if $T \in T_r$, there are b sets $T' \in T_{r+1}$ which are neighbours of T. The decoding algorithms starts at quering the One-Bit PartitionCountSketch for $r = 0$ to obtain a set S_0. Then, for every $i \in [1, r]$, it computes all the neighbours of S_{r-1}, where the for a total of $\mathcal{O}(b|S|)$ sets. Then using One-BitPartitionPointQuery we query every new partition, to obtain a set S_r of size $\mathcal{O}(k)$. The output of the algorithm is the set S_R. The running time then is computed as

$$\mathcal{O}(bRk\log(bkR/\delta) = \mathcal{O}(bk\frac{\log(n/k)}{\log b}(\log(k/\delta) + \log\log(n/k) - \log\log b))$$

□

From the above lemma, we get the following result, by carefully instatianting the parameter b.

Lemma 2.3.8. *There exists a b such that the* One-Bit *b-tree uses* $\mathcal{O}(\gamma^{-1}k\log(n/\delta))$ *measurements and runs in time* $\mathcal{O}(\gamma^{-1}(k\log(n/\delta))^{2+\gamma})$*, for any arbitarily constant* γ.

Proof. We set $b = (k\log(n/\delta))^\gamma$ and observe that the number measurements is at most

$$\mathcal{O}\left(k\frac{\log n}{\gamma(\log(k/\delta) + \log\log n)}(\log(k/\delta) + \log\log n)\right) = \mathcal{O}\left(\frac{1}{\gamma}k\log(n/\delta)\right),$$

while the decoding time becomes

$$\mathcal{O}\left((k\log(n/\delta))^\gamma k\frac{\log n}{\gamma(\log(k/\delta) + \log\log n)}(\log(k/\delta) + \log\log n)\right) = \mathcal{O}\left(\frac{1}{\gamma}(k\log(n/\delta))^{2+\gamma}\right)$$

□

2.3.4 One-Bit ExpanderSketch

In this subsection we prove Theorem 2.3.2. Given the results about ONE-BITPARTITIONCOUNTSKETCH we developed in the previous sections, the proof of the theorem is almost identical to [LNNT16] with a very simple modification. For completeness, we go again over their construction. We remind the reader that in our case $k = \mathcal{O}(\log n)$.

Construction of the Sensing Matrix: We first pick a code $\mathrm{enc} : \{0,1\}^{\log n} \to \{0,1\}^{\mathcal{O}(\log n)}$, which corrects a constant fraction of errors with linear-time decoding; such a code is guaranteed by [Spi96]. We then partition $\mathrm{enc}(i)$ into $s = \Theta(\log n / \log\log n)$ continuous substrings of length $t = \Theta(\log\log n)$. We denote by $\mathrm{enc}(i)_j$ the j-th bitstring of length t in $\mathrm{enc}(i)$.

We define s hash functions $h_1, h_2, \ldots, h_s : [n] \to [\mathrm{poly}(\log n)]$. Let also F be an arbitrary d-regular connected expander on the vertex set $[s]$ for some $d = \mathcal{O}(1)$. For $j \in [s]$, we define $\Gamma_j \subset [s]$ as the set of neighbours of j. Then, for every $j \in [n]$ we define the bit-strings

$$m_{i,j} = h_j(i) \circ \mathrm{enc}(i)_j \circ h_{\Gamma_1(j)}(i) \ldots \circ h_{\Gamma_d(j)}(i),$$

and the following partitions $\mathcal{P}^{(j)}$ containg set $P^{(j)}_{m_{i,j}}$, where $m_{i,j}$ is a string of $\Theta(t)$ bits, such that:

$$\forall i \in [n], i \in P^{(j)}_{m_{i,j}}$$

Then for every partition $\mathcal{P}^{(j)}$ we pick a random matrix $\Phi^{(j)}$ using Lemma 2.3.6 with sparsity k, as well as a random matrix $Z^{(j)}$ using Lemma 2.3.3 with sparsity k and failure probability $\frac{1}{\mathrm{poly}(\log n)}$. Each of these matrices has $\mathcal{O}(k \log(2^{\mathcal{O}(t)})) = \mathcal{O}(kt) = \mathcal{O}(k \log\log n)$ rows. The total number of rows is $\mathcal{O}(sk \log\log n) = \mathcal{O}(k \log n)$. Then our sensing matrix is the vertical concatenation of $\Phi^{(1)}, Z^{(1)}, \ldots, \Phi^{(s)}, Z^{(s)}$.

Decoding Algorithm: For every $j \in [s]$ we run the decoding algorithm of Lemma 2.3.6 on matrix $\Phi^{(j)}$ to obtain a list L_j of size $\mathcal{O}(k)$ such that every "heavy" set of $\mathcal{P}^{(j)}$ is included.

The running time in total is $m \cdot k \cdot \mathrm{poly}(\log n) = \mathrm{poly}(\log n)$. For every $j \in [s]$, we now have that:

- With probability $1/\mathrm{poly}(\log n)$, h_j perfectly hashes every $P^{(j)}_{m_{i,j}}$ for every $i \in H(x,k)$.
- With probability $1/\mathrm{poly}(\log n)$, for every $i^* \in H(x,k)$, $\|x_{P^{(j)}_{m_{i,j}}}\|_2 \geq \frac{9}{10}\|x_{-k}\|_2$.
- With probability $1/\mathrm{poly}(\log n)$, the decoding procedure on $\Phi^{(j)}$ succeeds. This follows by taking a union bound over the events of the previous two bullets and the failure probability guarantee of Lemma 2.3.6 in our instance.

We call by "name" of $P^{(j)}_{m_{i,j}}$ the $\mathcal{O}(\log\log n)$-length substring of bits of $m_{i,j}$, which correspond to the bits of $h_j(i)$. We then filter out vertices in layer j, by keeping only those that have unique names. Our next step is to point-query every set $z \in L_j$ using the matrices $Z^{(j)}$ and Theorem 2.3.3 and keep the largest $\mathcal{O}(k)$ coordinates; this is the difference with [LNNT16], since we can implement only one-bit point query. Now we let G be the graph created by including the at most $(d/2)\sum_{j=1}^{s} L_j$ edges suggested by the z's across all L_j, where we only include an edge if both endpoints suggest it. Now the algorithm and analysis proceeds exactly as [LNNT16].

Chapter 3

Sparse Fourier Transform

3.1 (Nearly) Sample Optimal Sparse Fourier Transform in Any Dimension

3.1.1 Preliminaries

For any positive integer n, we use $[n]$ to denote $\{1, 2, \cdots, n\}$.

We assume that the universe size $n = p^d$ for any positive integer p. Our algorithm facilitates $n = \Pi_{j=1}^d p_j$ for any positive integers $p_1, \ldots, p_d$, but we decide to present the case $n = p^d$ for ease of exposition; the proof is exactly the same in the more general case. Let $\omega = e^{2\pi \mathbf{i}/p}$ where $\mathbf{i} = \sqrt{-1}$. We will work with the normalized d-dimensional Fourier transform

$$\widehat{x}_f = \frac{1}{\sqrt{n}} \sum_{t \in [p]^d} x_t \cdot \omega^{f^\top t}, \forall f \in [p]^d$$

and the inverse Fourier transform is

$$x_t = \frac{1}{\sqrt{n}} \sum_{f \in [p]^d} \widehat{x}_f \cdot \omega^{-f^\top t}, \forall t \in [p]^d.$$

For any vector x and integer k, we denote x_{-k} to be the vector obtained by zeroing out the largest (in absolute value) k coordinates from x.

3.1.2 Our result

Apart from being dimension-independent and working for any universe size, our algorithm satisfies ℓ_∞/ℓ_2, which is the strongest guarantee out of the standard guarantees considered in compressed sensing tasks. A guarantee G_1 is stronger than guarantee G_2 if for any k-sparse recovery algorithm that satisfies G_1 we can obtain a $\Omega(k)$-sparse recovery algorithm that satisfies G_2. See also below for a comparison between ℓ_∞/ℓ_2 and ℓ_2/ℓ_2, the second stronger guarantee.

Previous work is summarized in Table 3.1. Our result is the following.

Theorem 3.1.1 (main result, informal version)**.** *Let $n = p^d$ where both p and d are positive integers. Let $x \in \mathbb{C}^{[p]^d}$. Let $k \in \{1, \dots, n\}$. Assume that $R^* \geq \|\widehat{x}\|_\infty / \|\widehat{x}_{-k}\|_2$ where $\log R^* = O(\log n)$ (signal-to-noise ratio). There is an algorithm that takes $O(k \log k \log n)$ samples from x, runs in $\widetilde{O}(n)$ time, and outputs a $O(k)$-sparse vector y such that*

$$\|\widehat{x} - y\|_\infty \leq \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2$$

holds with probability at least $1 - 1/\operatorname{poly}(n)$.

Comparison between ℓ_∞/ℓ_2 and ℓ_2/ℓ_2 (or ℓ_2/ℓ_1). For the sake of argument, we will consider only the ℓ_2/ℓ_2 guarantee which is stronger than ℓ_2/ℓ_1. The ℓ_2/ℓ_2 guarantee is the following: for $\widehat{x} \in \mathbb{C}^n$ one should output a z such that $\|\widehat{x} - z\|_2 \leq C\|\widehat{x}_{-k}\|_2$, where $C > 1$ is the approximation factor. Consider $C = 1.1$ [1], and think of the following signal: for a set S of size $0.05k$ we have $|\widehat{x}_i| = \frac{2}{\sqrt{k}} \|\widehat{x}_{\overline{S}}\|_2$. Then the all zeros vectors is a valid solution for the ℓ_2/ℓ_2 guarantee, since

$$\|\vec{0} - \widehat{x}\|_2^2 = \|\widehat{x}_S\|_2^2 + \|\widehat{x}_{\overline{S}}\|_2^2 = 0.05k \cdot \frac{4}{k} \|\widehat{x}_{\overline{S}}\|_2^2 + \|\widehat{x}_{\overline{S}}\|_2^2 = 1.2\|\widehat{x}_{\overline{S}}\|_2^2 < 1.1^2 \|\widehat{x}_{\overline{S}}\|_2^2.$$

It is clear that since $\vec{0}$ is a possible output, we may not recover any of the coordinates

[1] This is the case with the RIP based approaches, which obtain ℓ_2/ℓ_1. In fact many filter-based algorithms facilitate $(1+\epsilon)$ on the right hand side, with the number of measurements being multiplied by ϵ^{-1}. By enabling the same dependence on ϵ^{-1} our algorithm facilitates a multiplicative ϵ factor on right hand side of the ℓ_∞/ℓ_2, which makes it much stronger. Thus, a similar argument can go through.

Reference	**Samples**	**Time**	**Filter**	**RIP**	**Guarantee**
[GMS05]	$k \log^{O(d)} n$	$k \log^{O(d)} n$	Yes	No	ℓ_2/ℓ_2
[CT06]	$k \log^6 n$	$\mathrm{poly}(n)$	No	Yes	ℓ_2/ℓ_1
[RV08]	$k \log^2 k \log(k \log n) \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[HIKP12a]	$k \log^d n \log(n/k)$	$k \log^d n \log(n/k)$	Yes	No	ℓ_2/ℓ_2
[CGV13]	$k \log^3 k \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[IK14]	$2^{d \log d} k \log n$	$\widetilde{O}(n)$	Yes	No	ℓ_∞/ℓ_2
[Bou14]	$k \log k \log^2 n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[HR16]	$k \log^2 k \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[Kap16]	$2^{d^2} k \log n \log \log n$	$2^{d^2} k \log^{d+3} n$	Yes	No	ℓ_2/ℓ_2
[KVZ19]	$k^3 \log^2 k \log^2 n$	$k^3 \log^2 k \log^2 n$	Yes	Yes	Exactly k-sparse
Theorem 3.1.1	$k \log k \log n$	$\widetilde{O}(n)$	No	No	ℓ_∞/ℓ_2

Table 3.1: *$n = p^d$. We ignore the O for simplicity. The ℓ_∞/ℓ_2 is the strongest possible guarantee, with ℓ_2/ℓ_2 coming second, ℓ_2/ℓ_1 third and exactly k-sparse being the less strong. We note that [CT06, RV08, CGV13, Bou14, HR16] obtain a uniform guarantee, i.e. with $1 - 1/\mathrm{poly}(n)$ they allow reconstruction of all vectors; ℓ_∞/ℓ_2 and ℓ_2/ℓ_2 are impossible in the uniform case [CDD09]. We also note that [RV08, CGV13, Bou14, HR16] give improved analysis of the Restricted Isometry property; the algorithm is suggested and analyzed (modulo the RIP property) in [BD08]. The work in [HIKP12a] does not explicitly state the extension to the d-dimensional case, but can easily be inferred from the arguments. [HIKP12a, IK14, Kap16, KVZ19] work when the universe size in each dimension are powers of 2. We also assume that the signal-to-noise ratio is bounded by a polynomial of n, which is a standard assumption in the sparse Fourier transform literature [HIKP12a, IK14, Kap16, Kap17, LN19].*

in S, which is the set of "interesting" coordinates. On the other hand, the ℓ_∞/ℓ_2 guarantee does allow the recovery of every coordinate in S. This is a difference of recovering all $0.05k$ versus 0 coordinates. From the above discussion, one can conclude in the case where there is too much noise, ℓ_2/ℓ_2 becomes much weaker than ℓ_∞/ℓ_2, and can be even meaningless. Thus, ℓ_∞/ℓ_2 is highly desirable, whenever it is possible. The same exact argument holds for ℓ_2/ℓ_1.

3.1.3 Summary of previous Filter function based technique

One of the two ways to perform Fourier sparse recovery is by trying to implement arbitrary linear measurements, with algorithms similar to the ubiquitous CountSketch [CCF02]. In the general setting CountSketch hashes every coordinate to one of the $O(k)$ buckets, and repeats $O(\log n)$ times with fresh randomness. Then, it is guaranteed that every heavy coordinate will be isolated, and the contribution from non-heavy elements is small. To implement this in the Fourier setting becomes a highly non-trivial task however: one gets access only to the time-domain but not the frequency domain. One natural way to do this is to exploit the convolution theorem and find a function which is sparse in the time domain and approximates the indicator of an interval (rectangular pulse) in the frequency domain; these functions are called (bandpass) filters. Appropriate filters were designed in [HIKP12a, HIKP12b]: they were very good approximations of the rectangular pulse, i.e. the contribution from elements outside the passband zone contributed only by $1/\operatorname{poly}(n)$ their mass. These filters had an additional $\log n$ factor (in one dimension) in the sparsity of the time domain and they are sufficient for the purposes of [HIKP12a], but in high dimensions this factor becomes $\log^d n$. Filters based on the Dirichlet kernel give a better dependence in terms of sparsity and dimension (although still an exponential dependence on the latter), but the leak to subsequent buckets, i.e. coordinates outside the passband zone contribute a constant fraction of their mass, in contrast to the filter used in [HIKP12a]. Thus one should perform additional denoising, which is a non-trivial task. The seminal work of Indyk and Kapralov [IK14] was the first that showed how to perform sparse recovery with these filters, and then Kapralov [Kap16, Kap17] extended this result to run in sublinear time. We note, that any filter-based approach with filters which approximate the ℓ_∞ box, suffers from the curse of dimensionality. [KVZ19] devised an algorithm which avoids the curse of dimensionality by using careful aliasing, but it works in the noiseless case and has a cubic dependence on k.

3.1.4 RIP property-based algorithms: a quick overview

We say the matrix $A \in \mathbb{C}^{m\times n}$ satisfies RIP (Restricted Isometry Property [CT05]) of order k if for all k-sparse vectors $x \in \mathbb{C}^n$ we have $\|Ax\|_2^2 \approx \|x\|_2^2$. A celebrated result of Candes and Tao [CT06] shows that Basis Pursuit (ℓ_1 minimization) suffices for sparse recovery, as long as the samples from the time domain satisfy RIP. In [CT06] it was also proved using generic chaining that random sampling with oversampling factor $O(\log^6 n)$ gives RIP property for any orthonomal matrix with bounded entries by $1/\sqrt{n}$. Then [RV08] improved the bound to

$O(k \cdot \log^2 k \cdot \log(k \log n) \cdot \log n)$ and [CGV13] improved it to $O(k \cdot \log^3 k \cdot \log n)$. Subsequent improvement by Bourgain [Bou14] has lead to $O(k \log k \cdot \log^2 n)$ samples, improved by Haviv and Regev to $O(k \log^2 k \cdot \log n)$[HR16]. The fastest set of algorithms are iterative ones: for example Iterative Hard Thresholding [BD09a] or CoSaMP [NT09b] run $O(\log n)$ iterations[2] and each iteration takes $\widetilde{O}(n)$ time.

We note the very recent lower bound of [Rao19]: a subsampled Fourier matrix that satisfies the RIP properties should have $\Omega(k \log k \cdot d)$ rows[3]. This bound is particularly useful in high dimensions, since it deteriorates to a trivial bound in low dimensions. We still believe though that a bound of $\Omega(k \log k \log n)$ should hold in all dimensions. Thus, what remains is to obtain the ℓ_2/ℓ_2 guarantee by giving a tighter analysis, and removing the one $\log k$ factor to match the lower bound, but our algorithm already allows Fourier sparse recovery with these number of samples, even with a stronger guarantee.

3.1.5 Overview of our technique

Let $x \in \mathbb{C}^{[p]^d}$ denote our input signal in the time domain. In the following we assume the knowledge of $\mu = \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2$ and R^* which is an upper bound of $\|\widehat{x}\|_\infty / \mu$, and bounded by $\mathrm{poly}(n)$. These are standard assumption [HIKP12a, IK14, Kap16, Kap17, LN19] in the sparse Fourier transform literature. The bound on R^* is useful for bounding the running time and in any of [HIKP12a, IK14, Kap16, Kap17, LN19] a $\log n$ can be substituted by $\log R^*$ in the general case, which is also the case for our algorithm. We note that our algorithm will be correct with probability $1 - 1/\mathrm{poly}(n)$ whenever $R^* < 2^{n^{100}}$; this is fine for every reasonable application.

Consider the simplest scenario: $d = 1$, p is a prime number and a 1-sparse signal $\widehat{x}$ which is 1 on some frequency f^*. From a sample x_t in the time-domain what would be the

[2]To be precise, their running time is logarithmic in the signal-to-noise ratio, but we assumed throughtout this subchapter that this quantity is polynomial in n.

[3][BLLM19] independently gives a similar bound for $d = \log n$.

most reasonable way to find f^*? For every $f \in [p]$ we would compute

$$\sqrt{n}\omega^{ft}x_t = \sqrt{n}\omega^{ft} \cdot \frac{1}{\sqrt{n}} \sum_{f' \in [p]} \omega^{-f't} \widehat{x}_{f'} = \omega^{(f-f^*)t},$$

and keep, for $t \neq 0$, the frequency that gives a real number. Since $(f - f^*)t$ will be zero only for $f = f^*$, we are guaranteed correct recovery. In the noisy and multi-dimensional case or p is an arbitrary integer, however, this argument will not work, because of the presence of contribution from other elements and the fact that $(f - f^*)^\top t$ can be zero modulo p for other frequencies apart from f. However, we can take a number of samples t and average $\sqrt{n}\omega^{f^\top t}$, and hope that this will make the contribution from other frequencies small enough, so that we can infer whether f corresponds to a heavy coordinate or not. More specifically, we pick a list T of size $O(k)$ uniformly at random from $[p]^d$ and compute

$$\frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} x_t$$

for all frequencies f. We show that if $|T| = O(k)$ our estimator is good on average (and later we will maintain $O(\log n)$ independent instances and take the median to make sure with probability $1 - 1/\operatorname{poly}(n)$ the estimators for all the frequencies are good), and in fact behaves like a crude filter, similarly to the ones used in [IK14], in the sense that every coordinate contributes a non-trivial amount to every other coordinate. However, these estimators do not suffer from the curse of dimensionality and our case is a little bit different, requiring a quite different handling. The main reason is that in contrast to the filters used in [IK14], there is not an easy way to formulate an isolation argument from heavy elements that would allow easy measurement re-use, like Definition 5.2 and Lemma 5.4 from [IK14]. Buckets induced by filter functions have a property of locality, since they correspond to approximate ℓ_∞ boxes (with a polynomial decay outside of the box) in $[p]^d$: the closer two buckets are the more contribute the elements of one into the other. Our estimators on the other side do not enjoy such a property. Thus, one has to go via a different route.

In what follows, we will discuss how to combine the above estimators with an iterative loop that performs denoising, i.e. removes the contribution of every heavy element to other heavy elements.

We first implement a procedure which takes $O(k \log n)$ uniform random measurements from x and has the guarantee that for any $\nu \geq \mu$ any $y \in \mathbb{C}^{[p]^d}$ where $\|\widehat{x} - y\|_\infty \leq 2\nu$ and y is independent from the randomness of the measurements, the procedure outputs a $O(k)$-sparse $z \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y - z\|_\infty \leq \nu$ with probability $1 - 1/\operatorname{poly}(n)$.

Lemma 3.1.2 (LinfinityReduce procedure/data structure, informal). *Let $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$, and $\nu \geq \mu$. Let $\mathcal{T}^{(0)}$ be a list of $O(k \log n)$ i.i.d. elements in $[p]^d$. Let S be top $O(k)$ coordinates in $\widehat{x}$. There is a procedure that takes $\{x_t\}_{t \in \mathcal{T}}$, $y \in \mathbb{C}^{[p]^d}$ and ν as input, runs in $\widetilde{O}(n)$ time, and outputs $z \in \mathbb{C}^{[p]^d}$ so that if $\|\widehat{x} - y\|_\infty \leq 2\nu$, $\operatorname{supp}(y) \subseteq S$ and y is independent from the randomness of $\mathcal{T}^{(0)}$, then $\|\widehat{x} - y - z\|_\infty \leq \nu$ and $\operatorname{supp}(z) \subseteq S$ with probability $1 - 1/\operatorname{poly}(n)$ under the randomness of $\mathcal{T}^{(0)}$.*

Namely, we can take $O(k \log n)$ measurements and run the procedure in Lemma 3.1.2 to reduce (the upper bound of) the ℓ_∞ norm of the residual signal by half. We call the procedure in Lemma 3.1.2 LinfinityReduce procedure. More generally, we can take $O(H \cdot k \log n)$ measurements and run the LinfinityReduce procedure H times to reduce the ℓ_∞ norm of the residual signal to $1/2^H$ of its original magnitude, with failure probability at most $1/\operatorname{poly}(n)$. This is because if $\nu \geq 2^H \mu$ and $\|\widehat{x} - y\|_\infty \leq \nu$, then we can proceed in H iterations where in the h-th iteration ($h \in [H]$) we can take $O(k \log n)$ fresh measurements from x and run the LinfinityReduce procedure to make the ℓ_∞ norm of the residual signal at most $2^{-h}\nu$. Note that if we set $H = \log R^*$, we have already obtained a recovery algorithm taking $O(k \log n \log R^*)$ measurements, because we can drive down (the upper bound of) the ℓ_∞ norm of the residual signal from $\|\widehat{x}\|_\infty$ to μ in $\log R^*$ iterations.

$O(k \log n)$ measurements for $k = O(\log n)$

We first discuss a measurement reuse idea that leads us to a sparse recovery algorithm (Algorithm 8) taking $O(k \log n)$ measurements for $k = O(\log n)$. We set $H = 5$, and let $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}\}$, where each $\mathcal{T}^{(h)}$ is a list of $O(k \log n)$ i.i.d. elements in $[p]^d$. Note that

$\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}$ are independent. In our sparse Fourier recovery algorithm, we will measure x_t for all $t \in \mathcal{T}$.

In a nutshell, our approach finely discretizes the space of possible trajectories the algorithm could evolve, and carefully argues about the correctness of the algorithm by avoiding the intractable union-bound over all trajectories.

Recovery algorithm. The recovery algorithm proceeds in $\log R^* - H + 1$ iterations, where each iteration (except the last iteration) the goal is to reduce the upper bound of ℓ_∞ norm of the residual signal by half. Initially, the upper bound is R^*. It is important to note that we use the same measurements $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}\}$ in all of these $\log R^* - H + 1$ iterations.

In the following, we will describe one iteration of the recovery algorithm. Let $y \in \mathbb{C}^{[p]^d}$ denote the sparse vector recovered so far, and let the upper bound of $\|\widehat{x} - y\|_\infty$ be 2ν. Running the LinfinityReduce procedure H times where in the h-th time we use measurements in $\mathcal{T}^{(h)}$, we obtain a $O(k)$-sparse z such that with probability $1 - 1/\operatorname{poly}(n)$, $\|\widehat{x} - y - z\|_\infty \le 2^{1-H}\nu \le 0.1\nu$ (we call such z a desirable output by the LinfinityReduce procedure). Instead of taking $y + z$ as our newly recovered sparse signal, for each $f \in \operatorname{supp}(y+z)$, we project $y_f + z_f$ to the nearst points in $\mathcal{G}_{0.6\nu} := \{0.6\nu(x + y\mathbf{i}) : x, y \in \mathbb{Z}\}$ and assign to y'_f, where y' denotes our newly recovered sparse signal. For all $f \notin \operatorname{supp}(y+z)$, we let $y'_f = 0$.

To simplify our exposition, here we introduce some notations. We call $\mathcal{G}_{0.6\nu}$ a grid of side length 0.6ν, and we generalize the definition to any side length. Namely, for any $r_g > 0$, let grid $\mathcal{G}_{r_g} := \{r_g(x + y\mathbf{i}) : x, y \in \mathbb{Z}\}$. Moreover, we define $\Pi_{r_g} : \mathbb{C} \to \mathcal{G}_{r_g}$ to be the mapping that maps any element in $\mathbb{C}$ to the nearest element in $\mathcal{G}_{r_g}$. Now we can write y' as

$$y'_f = \begin{cases} \Pi_{0.6\nu}(y_f + z_f), & \text{if } f \in \operatorname{supp}(y+z); \\ 0, & \text{if } f \notin \operatorname{supp}(y+z). \end{cases}$$

At the end of each iteration, we assign y' to y, and shrink ν by half. In the last iteration, we will not compute y', instead we output $y + z$. We present the algorithm in Algorithm 8.

Algorithm 8 Fourier sparse recovery by projection, $O(k \log n)$ measurements when $k = O(\log n)$

```
1: procedure FourierSparseRecoveryByProjection($x, n, k, \mu, R^*$)    ▷ Section 3.1.5
2:     Require that $\mu = \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2$ and $R^* \geq \|\widehat{x}\|_\infty / \mu$
3:     $H \leftarrow 5, \nu \leftarrow \mu R^*/2,\ y \leftarrow \vec{0}$    ▷ $y \in \mathbb{C}^{[p]^d}$ refers to the sparse vector recovered so far
4:     Let $\mathcal{T} = \{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(H)}\}$ where each $\mathcal{T}^{(h)}$ is a list of i.i.d. uniform samples in $[p]^d$
5:     while true do
6:         $\nu' \leftarrow 2^{1-H}\nu$
7:         Use $\{x_t\}_{t \in \mathcal{T}}$ to run the LinfinityReduce procedure (in Lemma 3.1.2) $H$ times
   (use samples in $\mathcal{T}^{(h)}$ for each $h \in [H]$ ), and finally it finds $z$ so that $\|\widehat{x} - y - z\|_\infty \leq \nu'$
8:         if $\nu' \leq \mu$ then return $y + z$    ▷ We found the solution
9:         $y' \leftarrow \vec{0}$
10:        for $f \in \mathrm{supp}(y + z)$ do
11:            $y'_f \leftarrow \Pi_{0.6\nu}(y_f + z_f)$    ▷ We want $\|\widehat{x} - y'\|_\infty \leq \nu$ and the depend-
12:        end for    ▷ ence between $y'$ and $\mathcal{T}$ is under control
13:        $y \leftarrow y', \nu \leftarrow \nu/2$
14:    end while
15: end procedure
```

Analysis. We analyze y' conditioned on the event that $\|\widehat{x} - y - z\|_\infty \leq 0.1\nu$ (i.e. z is a desirable output by the LinfinityReduce procedure, which happens with probability $1 - 1/\mathrm{poly}(n)$). We will prove that y' has two desirable properties: (1) $\|\widehat{x} - y'\|_\infty \leq \nu$; (2) the dependence between y' and our measurements $\mathcal{T}$ is under control so that after taking y' as newly recovered sparse signal, subsequent executions of the LinfinityReduce procedure with measurements $\mathcal{T}$ still work with good probability. Property (1) follows from triangle inequality and the fact that $\|\widehat{x} - (y + z)\|_\infty \leq 0.1\nu$ and $\|(y + z) - y'\|_\infty \leq 0.6\nu$. We now elaborate on property (2). We can prove that for any $f \in [p]^d$,

$$y'_f \in \left\{\Pi_{0.6\nu}(\widehat{x}_f + 0.1\nu(\alpha + \beta\mathbf{i})) : \alpha, \beta \in \{-1, 1\}\right\}.$$

Let S denote top $26k$ coordinates (in absolute value) of $\widehat{x}$. We can further prove that for any $f \in \overline{S}$, $y'_f = 0$. Therefore, the total number of possible y' is upper bounded by $4^{|S|} = 4^{O(k)}$. If $k = O(\log n)$, we can afford union bounding all $4^{O(k)} = \mathrm{poly}(n)$ possible y', and prove that with probability $1 - 1/\mathrm{poly}(n)$ for all possible value of y' if we take y' as our newly recovered sparse signal then in the next iteration the LinfinityReduce procedure with measurements $\mathcal{T}$ gives us a desirable output.

Sufficient event. More rigorously, we formulate the event that guarantees successful execution of Algorithm 8. Let $\mathcal{E}_1$ be the event that for all $O(\log R^*)$ possible values of $\nu \in \{\mu\frac{R^*}{2}, \mu\frac{R^*}{4}, \dots, \mu 2^{H-1}\}$, for all possible vector y where $y_f = 0$ for $f \in \overline{S}$ and $y_f \in \{\Pi_{0.6\nu}(\widehat{x}_f + 0.1\nu(\alpha + \beta\mathbf{i})) : \alpha, \beta \in \{-1, 1\}\}$ for $f \in S$ (we also need to include the case that $y = \vec{0}$ for the success of the first iteration), running the LinfinityReduce procedure (in Lemma 3.1.2) H times (where in the h-th time measurements $\{x_t\}_{t \in \mathcal{T}^{(h)}}$ are used to reduce the error from $2^{2-h}\nu$ to $2^{1-h}\nu$) finally gives z so that $\|\widehat{x} - y - z\|_\infty \le 2^{1-H}\nu$. The randomness of $\mathcal{E}_1$ comes from $\mathcal{T} = \{\mathcal{T}^{(1)}, \dots, \mathcal{T}^{(H)}\}$.

First, event $\mathcal{E}_1$ happens with probability $1 - 1/\operatorname{poly}(n)$. This is because there are $4^{O(k)} \log R^*$ possible combinations of ν and y to union bound, and each has failure probability at most $1/\operatorname{poly}(n)$. For $k = O(\log n)$, and any $R^* < 2^{n^{100}}$ this gives the desired result. Second, conditioned on event $\mathcal{E}_1$ happens, Algorithm 8 gives correct output. This can be proved by a mathematical induction that in the t-th iteration of the while-true loop in Algorithm 8, $\|\widehat{x} - y\|_\infty \le 2^{-t}\mu R^*$.

$O(k \log k \log n)$ **measurements for arbitrary** k

Using random shift to reduce projection size. We remark that in the analysis of the previous recovery algorithm, if we can make sure that every $y_f + z_f$ has only one possible outcome when projecting to the grid $\mathcal{G}_{0.6\nu}$, then we no longer need to union bound $4^{O(k)}$ events. However, if $\widehat{x}_f$ is very close to a grid point in $\mathcal{G}_{0.6\nu}$ (or $\widehat{x}_f \in \mathcal{G}_{0.6\nu}$), then no matter how close $y_f + z_f$ and $\widehat{x}_f$ are, $\Pi_{0.6\nu}(y_f + z_f)$ will have 4 possible values.

To address this, we introduce random shift, whose property is captured by Lemma 3.1.3. To simplify notation, for any $r_b > 0$ and $c \in \mathbb{C}$ we define box $\mathcal{B}_\infty(c, r_b) := \{c + r_b(x + y\mathbf{i}) : x, y \in [-1, 1]\}$. For any $S \subseteq \mathbb{C}$, let $\Pi_{r_g}(S) = \{\Pi_{r_g}(c) : c \in S\}$.

Lemma 3.1.3 (property of a randomly shifted box, informal)**.** *If we take a box of side length $2r_b$ and shift it randomly by an offset in $\mathcal{B}_\infty(0, r_s)$ (or equivalently, $[-r_s, r_s] \times [-r_s, r_s]$) where $r_s \ge r_b$,* *and next we round every point inside that shifted box to the closest point in G_{r_g} where $r_g \geq 2r_s$, then with probability at least $(1 - r_b/r_s)^2$ everyone will be rounded to the same point.*

In the following, we present a sparse Fourier recovery algorithm that incorporates the random shift idea. The algorithm takes $O(k \log k \log n)$ measurements. We set $H = O(\log k)$ and take measurements of $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}\}$, where $\mathcal{T}^{(h)}$ is a list of $O(k \log n)$ i.i.d elements in $[p]^d$. Note that $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}$ are independent, and the choice of H is different from Section 3.1.5.

In a nutshell, our approach finely discretizes the space of possible trajectories the algorithm could evolve; in contrast to the case of $k = O(\log n)$, the number of trajectories becomes much larger. For that, we perform random shifting after the samples are taken, such that the number of trajectories is pruned, and we need to argue for a much smaller collection of events. We note that we make the decoding algorithm be randomized: the randomness in previous algorithms was present only when taking samples, and the rest of the algorithm was deterministic. However, here we need randomness in both cases, and that helps us prune the number of possible trajectories. To the best of our knowledge, this is a novel argument and approach, and might be helpful for future progress in the field.

Recovery algorithm. Similar to the $k = O(\log n)$ case (Section 3.1.5), we assume that we have already obtained a $O(k)$-sparse $y \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y\|_\infty \leq 2\nu$ and y is "almost" independent from $\mathcal{T}$. We show how to obtain $y' \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y'\|_\infty \leq \nu$ with probability $1 - 1/\operatorname{poly}(n)$ and y' is "almost" independent from $\mathcal{T}$. The main idea is we first run LinfinityReduce procedure $H = O(\log k)$ times to get an $O(k)$-sparse $z \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y - z\|_\infty \leq \frac{1}{2^{20}k}\nu$. Then we repeatedly sample a uniform random shift $s \in \mathbb{C}$ (where $\|s\|_\infty \leq 10^{-3}\nu$; here we consider complex numbers as 2D vectors) until for every $f \in \operatorname{supp}(y + z)$, all the points (or complex numbers) of the form $y_f + z_f + s + a + b\mathbf{i}$ where $a, b \in [-\frac{\nu}{2^{20}k}, \frac{\nu}{2^{20}k}]$ round to the same grid point in $\mathcal{G}_{0.04\nu}$. Finally, for every $f \in \operatorname{supp}(y + z)$, we assign $\Pi_{0.04\nu}(y_f + z_f + s)$ to y'_f; all remaining coordinates in y' will be assigned to 0. We present an informal version of our algorithm in Algorithm 9, and defer its formal version to

Algorithm 9 Fourier sparse recovery by random shift and projection (informal version)

```
1: procedure FourierSparseRecovery($x, n, k, \mu, R^*$)        ▷ Theorem 3.1.1, $n = p^d$
2:     Require that $\mu = \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2$ and $R^* \geq \|\widehat{x}\|_\infty / \mu$
3:     $H \leftarrow O(\log k)$, $\nu \leftarrow \mu R^*/2$, $y \leftarrow \vec{0}$   ▷ $y \in \mathbb{C}^{[p]^d}$ refers to the sparse vector recovered so far
4:     Let $\mathcal{T} = \{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(H)}\}$ where each $\mathcal{T}^{(h)}$ is a list of i.i.d. uniform samples in $[p]^d$
5:     while true do
6:         $\nu' \leftarrow \frac{1}{2^{20}k} \nu$
7:         Use $\{x_t\}_{t \in \mathcal{T}}$ to run the LinfinityReduce procedure (in Lemma 3.1.2) $H$ times (use samples in $\mathcal{T}^h$ for each $h \in [H]$ ), and finally it finds $z$ so that $\|\widehat{x} - y - z\|_\infty \leq \nu'$
8:         if $\nu' \leq \mu$ then return $y + z$        ▷ We found the solution
9:         repeat
10:            Pick $s \in \mathcal{B}_\infty(0, 10^{-3}\nu)$ uniformly at random
11:        until $\forall f \in \mathrm{supp}(y+z), |\Pi_{0.04\nu}(\mathcal{B}_\infty(y_f + z_f + s, \nu'))| = 1$
12:        $y' \leftarrow \vec{0}$
13:        for $f \in \mathrm{supp}(y+z)$ do
14:            $y'_f \leftarrow \Pi_{0.04\nu}(y_f + z_f + s)$        ▷ We want $\|\widehat{x} - y'\|_\infty \leq \nu$ and the depend-
15:        end for                                   ▷ ence between $y'$ and $\mathcal{T}$ is under control
16:        $y \leftarrow y'$, $\nu \leftarrow \nu/2$
17:    end while
18: end procedure
```

the appendix.

Analysis. Now we analyze the above approach. First, we have the guarantee that $\|\widehat{x} - y'\|_\infty \leq \nu$. Moreover, note that by our choice of s, for every $f \in \mathrm{supp}(y+z)$, $y_f + z_f + s$ and $\widehat{x}_f + s$ round to the same grid point in $\mathcal{G}_{0.04\nu}$. Therefore, for the new vector y' we have recovered, we "hide" the randomness in $\mathcal{T}$, and the randomness only leaks from failed attempts of the shifts. In the following, we show that each attempt of shift succeeds with probability $\frac{1}{2}$.

We can restate the procedure of choosing s to be:

$$\textbf{repeatedly} \text{ sample } s \sim \mathcal{B}_\infty(0, 10^{-3}\nu),$$
$$\textbf{until} \text{ for all } f \in \mathrm{supp}(y+z), \left|\Pi_{0.04\nu}\left(\mathcal{B}_\infty(y_f + z_f + s, \frac{\nu}{2^{20}k})\right)\right| = 1.$$

Note that $|\mathrm{supp}(y+z)| = O(k)$. Let us say that we can always guarantee that $|\mathrm{supp}(y + z)| \leq 50k$. By Lemma 3.1.3 where we let $r_b = \frac{\nu}{2^{20}k}$, $r_s = 10^{-3}\nu$ and $r_g = 0.04\nu$, for

$f \in \mathrm{supp}(y+z)$,

$$\Pr\left[\left|\Pi_{0.04\nu}\left(\mathcal{B}_\infty(y_f+z_f+s, \frac{\nu}{2^{20}k})\right)\right| = 1\right] \geq (1-\frac{r_b}{r_s})^2 \geq 1-\frac{1}{100k}.$$

By a union bound over $f \in \mathrm{supp}(y+z)$, the probability is at least $\frac{1}{2}$ that for all $f \in \mathrm{supp}(y+z)$, $|\Pi_{0.04\nu}(\mathcal{B}_\infty(y_f+z_f+s, \frac{\nu}{2^{20}k}))| = 1$.

Therefore, with probability $1 - 1/\operatorname{poly}(n)$, we will only try $O(\log n)$ shifts. We can apply a union bound over $O(\log n)$ possible shifts, and prove that with probability $1-1/\operatorname{poly}(n)$ if taking y' as our new y, and shrinking ν by half, the LinfinityReduce procedure will work as desired as if there is no dependence issue.

Sufficient event. Let S be top $O(k)$ coordinates in $\widehat{x}$. Let $L = O(\log R^*)$ denote the number of iterations in Algorithm 9. For $\ell \in [L]$, let $\nu_\ell = 2^{-\ell}\mu R^*$. For $\ell \in [L-1]$, let $s_\ell^{(a)}$ be the a-th uniform randomly sampled from $\mathcal{B}_\infty(0, 10^{-3}\nu_\ell)$ as appeared on Line 10 in Algorithm 9. For the sake of analysis, we assume that Algorithm 9 actually produces an infinite sequence of shifts $s_\ell^{(1)}, s_\ell^{(2)}, \ldots$. We formulate the event that guarantees successful execution of Algorithm 9. We define event $\mathcal{E}_2$ to be all of the following events hold.

1. For all $\ell \in [L-1]$, there exists $a \in [10\log n]$ so that for all $f \in S$,

$$\left|\Pi_{0.04\nu_\ell}\left(\mathcal{B}_\infty(\widehat{x}_f + s_\ell^{(a)}, \frac{1}{100k}\nu_\ell)\right)\right| = 1.$$

2. For $\ell = 1$, if we run the LinfinityReduce procedure H times with $y = \vec{0}$ and measurements in $\mathcal{T}$, we get z such that $\|\widehat{x} - z\|_\infty \leq 2^{1-H}\nu_1$ and $\mathrm{supp}(z) \subseteq S$.

3. For all $\ell \in \{2, \ldots, L\}$, for all $a \in [10\log n]$, if we run the LinfinityReduce procedure H times with $y = \xi$ where

$$\xi_f = \begin{cases} \Pi_{0.04\nu_\ell}(\widehat{x}_f + s_{\ell-1}^{(a)}), & \text{if } f \in S; \\ 0, & \text{if } f \in \overline{S}. \end{cases}$$

then we get z such that $\|\widehat{x} - y - z\|_\infty \leq 2^{1-H}\nu_\ell$ and $\mathrm{supp}(y+z) \subseteq S$.

We can prove that event $\mathcal{E}_2$ happens with probability $1 - 1/\operatorname{poly}(n)$. Moreover, we can prove that conditioned on event $\mathcal{E}_2$ Algorithm 9 gives correct output. We defer both proofs in the appendix.

3.1.6 Algorithm for d-dimensional Sparse Fourier Transform

In this section, we will give a Fourier sparse recovery algorithm that takes $O(k \log k \log n)$ measurements with "ℓ_∞/ℓ_2" guarantee. We assume the knowledge of $\mu = \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2$. In fact, a constant factor approximation suffices, but we prefer to assume exact knowledge of it in order to simplify exposition. All of the arguments go through in the other case, with minor changes in constants. We also assume we know R^* so that $R^* \geq \|\widehat{x}\|_\infty / \mu$. We assume that $\log R^* = O(\log n)$. For larger $\log R^* = O(\operatorname{poly}(n))$, our algorithm will still work, but the decoding time will be worse by a factor of $\frac{\log R^*}{\log n}$. Note that our assumptions on μ and R^* are standard. For example, [IK14] make the same assumption. We assume that we can measure the signal x in the time domain, and we want to recover the signal $\widehat{x}$ in the frequency domain.

In our algorithm, we will use μ as a threshold for noise, and we will perform $\log R^*$ iterations, where in each iteration the upper bound of ℓ_∞ norm of the residual signal (in the frequency domain) shrinks by half. In Section 3.1.7, we give some definitions that will be used in the algorithm. Then we present our new algorithm for d-dimension Fourier sparse recovery in Section 3.1.8. In Section 3.1.9, we prove the correctness of the proposed algorithm.

3.1.7 Notations

For a subset of samples (or measurements) $\{x_t\}_{t \in T}$ from the time domain, where T is a list of elements in $[p]^d$, we define $\widehat{x}^{[T]}$ in Definition 3.1.4 as our estimation to $\widehat{x}$.

Definition 3.1.4 (Fourier transform of a subset of samples)**.** *Let $x \in \mathbb{C}^{[p]^d}$. For any T which is*

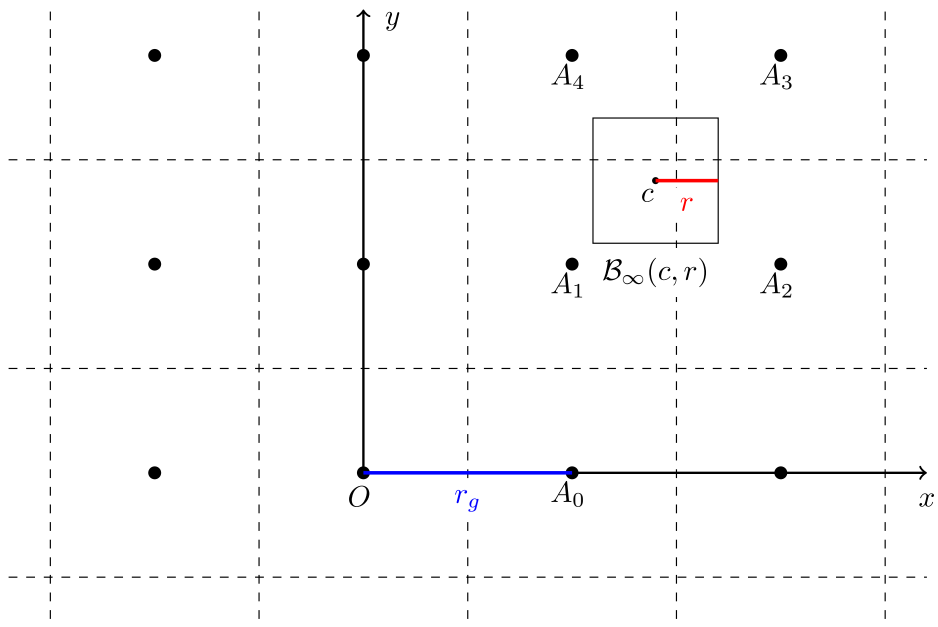


Figure 3.1: *Illustration of box $\mathcal{B}_\infty(c,r)$ and grid $\mathcal{G}_{r_g}$. Box $\mathcal{B}_\infty(c,r)$ refers to all the points in the square centered at c with side length $2r$. Grid $\mathcal{G}_{r_g}$ refers to all the solid round points, and the distance between origin O and A_0 is r_g. Note that the dashed lines are decision boundaries of the projection Π_{r_g}, and all the points inside a minimum cell separated by the dashed lines are mapped (by Π_{r_g}) to the same grid point in $\mathcal{G}_{r_g}$ (which is the center of the cell). We have $\Pi_{r_g}(c) = A_1$ and $\Pi_{r_g}(\mathcal{B}_\infty(c,r)) = \{A_1, A_2, A_3, A_4\}$.*

a list of elements in $[p]^d$, for any $f \in [p]^d$, we define

$$\widehat{x}_f^{[T]} = \frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} x_t.$$

In order to reuse samples across different iterations where we drive down the upper bound of the residual signal by half, in each iteration after we obtain estimations to heavy hitters (or equivalently large coordinates), instead of subtracting the estimates directly, we need to "hide" the randomness leaked by the samples. We interpret each estimate (which is a complex number) as a point on a 2-dimension plane, and hide the randomness by rounding the estimate to the nearest grid point (where the side length of the grid is chosen to be a small constant fraction of the target ℓ_∞ norm of the residual signal in the frequency domain), which we call "projection onto grid". In Definition 3.1.5, we formally define box and grid, and in Definition 3.1.6 we define projection to grid. We illustrate these two definitions in Figure 3.1.

Definition 3.1.5 (box and grid)**.** *For any $c \in \mathbb{C}$ and $r \geq 0$, we define box $\mathcal{B}_\infty(c,r) \subseteq \mathbb{C}$ as*

$$\mathcal{B}_\infty(c,r) = \{c + x + y\mathbf{i} : x, y \in [-r, r]\}.$$

Namely, if we consider complex numbers as points on 2D plane, box $\mathcal{B}_\infty(c,r)$ refers to ℓ_∞ ball with radius r centered at c.

For any $r > 0$, we define grid $\mathcal{G}_r \subseteq \mathbb{C}$ as

$$\mathcal{G}_r = \{xr + yr\mathbf{i} : x, y \in \mathbb{Z}\}.$$

Definition 3.1.6 (projection onto grid)**.** *For any $r > 0$, we define Π_r to be a maping from $\mathbb{C}$ to $\mathcal{G}_r$, so that for any $c \in \mathbb{C}$,*

$$\Pi_r(c) = \arg\min_{c' \in \mathcal{G}_r} |c - c'|,$$

where we break the tie by choosing the one with minimum $|c'|$. As a natural generalization, For $C \subseteq \mathbb{C}$, we define

$$\Pi_r(C) = \{\Pi_r(c) : c \in C\}.$$

3.1.8 Algorithm

We present our new sparse Fourier recovery algorithm in Algorithm 10. Its auxiliary function LinfinityReduce is in Algorithm 11. Important constants are summarized in Table 3.2.

In Algorithm 10, we define "bucket size" $B = O(k)$ and number of repetitions $R = O(\log n)$. For each $r \in [R]$, we choose $\mathcal{T}_r$ to be a list of B independent and uniformly random elements in $[p]^d$. We will measure x_t for all $t \in \cup_{r \in [R]} \mathcal{T}_r$, and use LinfinityReduce in Algorithm 11 to locate and estimate all the "heavy hitters" of the residual signal so that if we substract them then the ℓ_∞ norm of the new residual signal shrinks by half. The input to LinfinityReduce is a signal $x \in \mathbb{C}^{[p]^d}$ in the time domain (but we can only get access to x_t where $t \in \cup_{r \in [R]} \mathcal{T}_r$), a sparse vector $y \in \mathbb{C}^{[p]^d}$ in the frequency domain that we have recovered so far, and $\nu \geq \mu$ such that $\|\widehat{x} - y\|_\infty \leq 2\nu$ where we will refer $\widehat{x} - y$ as the currect residual signal (in the frequency domain). It is guaranteed that LinfinityReduce$(x, n, y, \{\mathcal{T}_r\}_{r=1}^R, \nu)$

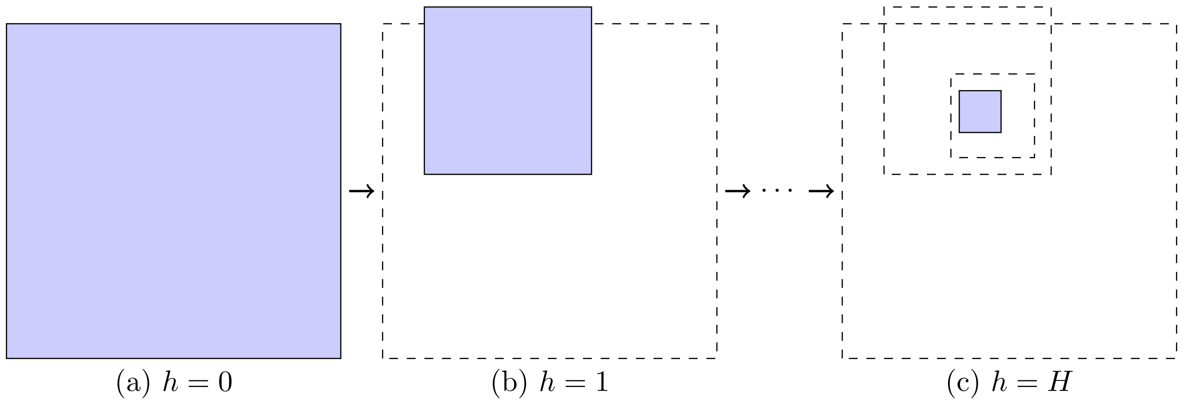


Figure 3.2: *Illustration of the behavior of Line 16 to Line 20 in Algorithm 10. For any $f \in [p]^d$, we draw box $\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f, 2^{1-h}\nu_\ell)$ after h iterations of the for loop between Line 17 and Line 19 in Algorithm 10, where $h \in \{0, 1, \dots, H\}$. Conditioned on* LinfinityReduce *is correct, for every $h \in \{0, 1, \dots, H\}$, after h-th iteration we have $\widehat{x}_f \in \mathcal{B}_\infty(y_f^{(\ell-1)} + z_f, 2^{1-h}\nu_\ell)$. When $h = 0$, i.e. before the loop between Line 17 and Line 19 starts, we know that $\widehat{x}_f \in \mathcal{B}_\infty(y_f^{(\ell-1)}, 2\nu_\ell)$ as depicted by (a). After each iteration in h, the radius of the box shrinks by half (and its center might change). Finally after H iterations, as depicted by (c), we obtain $z^{(\ell-1)}$ such that $\widehat{x}_f \in \mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)}, 2^{1-H}\nu_\ell)$.*

returns a $O(k)$-sparse z so that $\|\widehat{x} - y - z\| \leq \nu$ with probability $1 - 1/\operatorname{poly}(n)$.

Algorithm 10 in total maintains $H = O(\log k)$ independent copies of such error-reduce data structures, where in the h-th copy it measures $\mathcal{T}^{(h)} = \{\mathcal{T}_r^{(h)}\}_{r \in [R]}$ for $h \in [H]$. We denote $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{r \in [R]}$. If $\log R^* \leq H$, then we can simply use different $\mathcal{T}^{(h)}$ in different iterations. In that case $L = 1$ and $H = \log R^*$ in Algorithm 10. We will get $z^{(1)}$ on Line 20 such that $\|\widehat{x} - y^{(0)} - z^{(1)}\|_\infty \leq \mu$ (we will prove in the analysis this holds with probability $1 - 1/\operatorname{poly}(n)$) where $y^{(0)} = 0$, and return $z^{(1)} + y^{(0)}$ on Line 22.

If $\log R^* > H$, we have to reuse the samples. We proceed in L iterations (in the loop between Line 14 and Line 33 in Algorithm 10), where $L = \log R^* - H + 1$. For $\ell \in [L]$, as defined in Line 15, $\nu_\ell = 2^{-\ell}\mu R^*$ refers to the target ℓ_∞ of the residual signal in the ℓ-th iteration (namely, for $\ell \in [L-1]$ we want to obtain $y^{(\ell)}$ so that $\|\widehat{x} - y^{(\ell)}\|_\infty \leq \nu_\ell$). In the ℓ-th iteration where $\ell \in [L]$, by using the samples in $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h \in H}$ (Line 16 to Line 20), the algorithm tries to get $z^{(\ell)}$ so that $\|\widehat{x} - y^{(\ell-1)} - z^{(\ell)}\|_\infty \leq 2^{1-H}\nu_\ell$. The intuition on the behavior of Line 16 to Line 20 is depicted in Figure 3.2.

If $\ell = L$ the algorithm will return $y^{(L-1)} + z^{(L)}$ as in Line 22; otherwise, the algorithm will

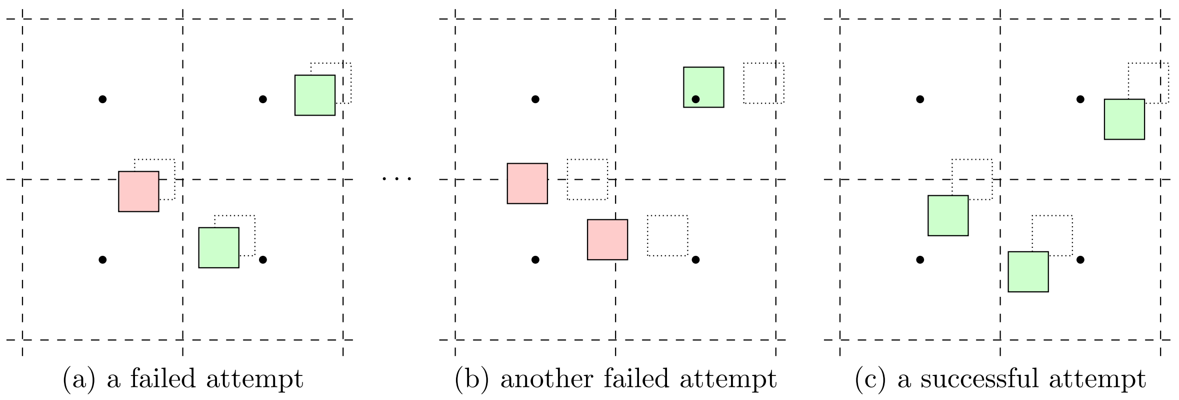


Figure 3.3: *Illustration of the iteration between Line 25 and Line 28 in Algorithm 10. The round solid points represent grid points in $\mathcal{G}_{\beta\nu}$, and the dashed lines represent decision boundaries of $\Pi_{\beta\nu_\ell}$. In this example we have $|\operatorname{supp}(y^{(\ell-1)} + z^{(\ell)})| = 3$, and the dotted squares represent boxes $\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)}, 2^{1-H}\nu_\ell)$ for $f \in \operatorname{supp}(y^{(\ell-1)} + z^{(\ell)})$. The algorithm repeatedly samples a random shift $s \sim \mathcal{B}_\infty(0, \alpha\nu_\ell)$, until all the shifted boxes $\{\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)}, 2^{1-H}\nu_\ell) + s\}_{f\in\operatorname{supp}(y^{(\ell-1)}+z^{(\ell)})}$ do not intersect with the dashed lines (i.e. decision boundaries of $\Pi_{\beta\nu_\ell}$). In the figure, we color a shifted box in green if it does not intersect with dashed lines, and color in red otherwise. After a series of failed attempts from (a) to (b), we finally have a successful attempt in (c).*

try to compute $y^{(\ell)}$ based on $y^{(\ell-1)} + z^{(\ell)}$. In Line 25 to Line 28, the algorithm repeatedly samples a uniform random shift $s_\ell \in \mathcal{B}_\infty(0, \alpha\nu_\ell)$ (where $\alpha \in (0,1)$ is a small constant chosen in Table 3.2) until the shift is good, where shift s_ℓ is good if and only if for each $f \in \operatorname{supp}(y^{(\ell-1)} + z^{(\ell)})$, all the points in $\mathcal{B}_\infty(y^{(\ell-1)} + z^{(\ell)} + s_\ell, 2^{1-H}\nu_\ell)$ (i.e. the box obtained by applying shift s_ℓ to the box $\mathcal{B}_\infty(y^{(\ell-1)} + z^{(\ell)}, 2^{1-H}\nu_\ell)$) project to the same grid point in $\mathcal{G}_{\beta\nu_\ell}$. We depict the process of obtaining the shift s_ℓ in Figure 3.3. It is crucial to note that if the shift s_ℓ is good and the vector $z^{(\ell)}$ we get is desirable (namely $\|\widehat{x} - y^{(\ell-1)} - z^{(\ell)}\|_\infty \le 2^{1-H}\nu_\ell$), then for each $f \in \operatorname{supp}(y^{(\ell-1)} + z^{(\ell)})$, $\Pi_{\beta\nu_\ell}(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell) = \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell)$.

On Line 31, we assign $\Pi_{\beta\nu_\ell}(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell)$ to $y_f^{(\ell)}$. Because β is a small constant, we still have the guarantee that $\|\widehat{x} - y^{(\ell)}\|_\infty \le \nu_\ell$. Moreover, by assigning $\Pi_{\beta\nu_\ell}(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell) = \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell)$ to $y_f^{(\ell)}$, we "hide" the randomness in $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h\in[H]}$. Now the randomness in $\mathcal{T}$ only leaks from failed attempts of the shifts. For analysis purpose, we maintain a counter a_ℓ for $\ell \in [L-1]$ recording the number of attempts until we have sampled a good one. By our choice of parameters, we can prove that with high probability $a_\ell \le 10 \log n$ for each $\ell \in [L-1]$. Thus intuitively the leaked randomness is under control,

Algorithm 10 Fourier sparse recovery by random shift and projection

```
procedure FourierSparseRecovery($x, n, k, \mu, R^*$)    ▷ Theorem 3.1.19, $n = p^d$
    Require that $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$ and $R^* \geq \|\widehat{x}\|_\infty / \mu$    ▷ $R^*$ is a power of 2
    $B \leftarrow C_B \cdot k$    ▷ $C_B$ is a constant defined in Table 3.2
    $R \leftarrow C_R \cdot \log n$    ▷ $C_R$ is a constant defined in Table 3.2
    $H \leftarrow \min\{\log k + C_H, \log R^*\}$    ▷ $C_H$ is a constant defined in Table 3.2
    for $h = 1 \to H$ do
        for $r = 1 \to R$ do
            $\mathcal{T}_r^{(h)} \leftarrow$a list of $B$ i.i.d elements in $[p]^d$
        end for
        $\mathcal{T}^{(h)} \leftarrow \{\mathcal{T}_r^{(h)}\}_{r=1}^R$
    end for    ▷ We will measure $x_t$ for $t \in \cup_{h\in[H], r\in[R]}\mathcal{T}_r^{(h)}$
    $y^{(0)} \leftarrow \vec{0}$    ▷ $y^{(0)} \in \mathbb{C}^n$
    $L \leftarrow \log R^* - H + 1$
    for $\ell = 1 \to L$ do
        $\nu_\ell \leftarrow 2^{-\ell}\mu R^*$    ▷ Target $\ell_\infty$ of the residual signal in iteration $t$
        $z \leftarrow \vec{0}$    ▷ $z$ is a temporary variable used to compute $z^{(\ell)}$
        for $h = 1 \to H$ do
            $z \leftarrow z + $ LinfinityReduce($x, n, y^{(\ell-1)} + z, \mathcal{T}^{(h)}, 2^{1-h}\nu_\ell$)
        end for
        $z^{(\ell)} \leftarrow z$    ▷ We want $\|\widehat{x} - y^{(\ell-1)} - z^{(\ell)}\|_\infty \leq 2^{1-H}\nu_\ell$
        if $\ell = L$ then
            return $y^{(L-1)} + z^{(L)}$
        end if
        $a \leftarrow 0$    ▷ A temporary counter maintained for analysis purpose only
        repeat
            Pick $s_\ell \in \mathcal{B}_\infty(0, \alpha\nu_\ell)$ uniformly at random    ▷ $\alpha \in (0,1)$ is a small constant
            $a \leftarrow a + 1$    ▷ $\beta$ in the next line is a small constant where $\alpha < \beta < 0.1$
        until $\forall f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)}), |\Pi_{\beta\nu_\ell}(\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell, 2^{1-H}\nu_\ell))| = 1$
        $a_\ell \leftarrow a$
        for $f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)})$ do
            $y_f^{(\ell)} \leftarrow \Pi_{\beta\nu_\ell}(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell)$    ▷ We want $\|\widehat{x} - y^{(\ell)}\|_\infty \leq \nu_\ell$
        end for
    end for
end procedure
```

and we can formally apply a union bound to prove that with good probability all possible invocations of LinfinityReduce by our FourierSparseRecovery produce desirable output.

3.1.9 Analysis

In order to analyze the algorithm, let $S \subseteq [n]$ be top $C_S k$ coordinates of $\widehat{x}$ where $C_S = 26$, and let $\overline{S} = [n] \setminus S$. In order to analyze the performance of LinfinityReduce in Algorithm 11, we need the following definition.

Definition 3.1.7 (uniform sample)**.** *We say t is sampled from $[p]^d$ uniformly at random if for each $i \in [d]$, we independently sample t_i from $[p]$ uniformly at random. We use $t \sim [p]^d$ to denote it.*

Fact 3.1.8. *Let $\omega = e^{2\pi \mathbf{i}/p}$ where p is any positive integer. For a fixed $f \in [p]^d \setminus \{\vec{0}\}$, $\mathbf{E}_{t\sim[p]^d}[\omega^{f^\top t}] = 0$.*

Proof. Note that $\mathbf{E}_{t\sim[p]^d}[\omega^{f^\top t}] = \prod_{i\in[d]} \mathbf{E}_{t_i\sim[p]}[\omega^{f_i t_i}]$ by the fact that $t_1, \dots, t_d$ are independent. Because $f \neq \vec{0}$, there exists $i \in [d]$ so that $f_i \neq 0$. We have

$$\begin{aligned}
\mathop{\mathbf{E}}_{t_i\sim[p]}[\omega^{f_i t_i}] &= \frac{1}{p}\sum_{j=0}^{p-1} (\omega^{f_i})^j \\
&= \frac{1}{p} \cdot \frac{(\omega^{f_i})^0 (1 - (\omega^{f_i})^p)}{1 - \omega^{f_i}} \\
&= 0,
\end{aligned}$$

where the second step follows from the sum of geometry series where $\omega^{f_i} \neq 1$, adn the third step follow from $(\omega^{f_i})^p = e^{2\pi \mathbf{i} f_i} = 1$. Therefore, $\mathbf{E}_{t\sim[p]^d}[\omega^{f^\top t}] = 0$.

□

We define measurement coefficient as follows:

Definition 3.1.9 (measurement coefficient)**.** *For any $f \in [p]^d$ and any T which is a list of elements in $[p]^d$, we define*

$$c_f^{[T]} = \frac{1}{|T|} \sum_{t\in T} \omega^{f^\top t}.$$

By definition of $c_f^{[T]}$ and d-dimension Fourier transform, we can decompose $\widehat{x}_f^{[T]}$ as follows.

Notation	Choice	Statement	Parameter
C_B	10^6	Lemma 3.1.18	B
C_R	10^3	Lemma 3.1.17	R
C_H	20	Algorithm 10	H
α	10^{-3}	Algorithm 10 Line 26	shift range
β	0.04	Algorithm 10 Line 28	grid size
C_S	26	Lemma 3.1.17, Lemma 3.1.18	$\|S\|$

Table 3.2: *Summary of important constants.*

Lemma 3.1.10 (measurement decomposition)**.** *For any $f \in [p]^d$ and any T which is a list of elements in $[p]^d$,*

$$\widehat{x}_f^{[T]} = \sum_{f' \in [p]^d} c_{f-f'}^{[T]} \widehat{x}_{f'}.$$

Proof. We have

$$\begin{aligned}
\widehat{x}_f^{[T]} &= \frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} x_t \\
&= \frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} \frac{1}{\sqrt{n}} \sum_{f' \in [p]^d} \omega^{-f'^\top t} \widehat{x}_{f'} \\
&= \frac{\sqrt{n}}{|T|} \sum_{t \in T} \frac{1}{\sqrt{n}} \sum_{f' \in [p]^d} \omega^{(f-f')^\top t} \widehat{x}_{f'} \\
&= \sum_{f' \in [p]^d} \left(\frac{1}{|T|} \sum_{t \in T} \omega^{(f-f')^\top t} \right) \widehat{x}_{f'} \\
&= \sum_{f' \in [p]^d} c_{f-f'}^{[T]} \widehat{x}_{f'},
\end{aligned}$$

where the first step follow by the definition of $\widehat{x}_f^{[T]}$ in Definition 3.1.4, second step follows by the definition of inverse d-dimensional Fourier transform (see Section 3.1.1), third and forth step follow by rearranging terms, last step follows by the definition of measurement coefficients c in Definition 3.1.9. □

Let T be a list of i.i.d. samples from $[p]^d$, then the coeffcients $c_f^{[T]}$ defined in Definition 3.1.9 have the following property.

Lemma 3.1.11 (properties of coeffcient c)**.** *Let T be a list of B independent and uniform random*

Lemma	Meaning
Lemma 3.1.10	measurement decomposition
Lemma 3.1.11	properties of coefficient
Lemma 3.1.12	noise bound
Lemma 3.1.13	guarantee of LINFINITYREDUCE
Lemma 3.1.15	property of a randomly shifted box
Lemma 3.1.17	event $\mathcal{E}$ happens
Lemma 3.1.18	correctness of our algorithm

Table 3.3: *Summary of Lemmas.*

elements in $[p]^d$. Then we have

1. $c_0^{[T]} = 1$.

2. For any $f \in [p]^d \setminus \{0\}$, $\mathbf{E}_T \left[|c_f^{[T]}|^2 \right] = \frac{1}{B}$.

3. For any $f, f' \in [p]^d$, $f \neq f'$, $\mathbf{E}_T \left[c_f^{[T]} \cdot \overline{c_{f'}^{[T]}} \right] = 0$.

Proof. **Part 1.** By definition of $c_0^{[T]}$,

$$c_0^{[T]} = \frac{1}{|T|} \sum_{t \in T} \omega^{0 \cdot t} = 1.$$

Part 2. Let $T = \{t_1, \dots, t_B\}$, where t_i is independently and uniformly chosen from $[p]^d$. For any $f \in [p]^d \setminus \{0\}$,

$$\begin{aligned}
\mathop{\mathbf{E}}_T \left[|c_f^{[T]}|^2 \right] &= \mathop{\mathbf{E}}_T \left[c_f^{[T]} \cdot \overline{c_f^{[T]}} \right] \\
&= \frac{1}{|T|^2} \mathop{\mathbf{E}}_T \left[\sum_{i,j \in [B]} \omega^{f^\top (t_i - t_j)} \right] \\
&= \frac{1}{|T|^2} \left(|T| + \mathop{\mathbf{E}}_T \left[\sum_{i,j \in [B], i \neq j} \omega^{f^\top (t_i - t_j)} \right] \right) \\
&= \frac{1}{|T|} + \frac{1}{|T|^2} \sum_{i,j \in [B], i \neq j} \mathop{\mathbf{E}}_T \left[\omega^{f^\top (t_i - t_j)} \right] \\
&= \frac{1}{|T|} - \frac{1}{|T|^2} \cdot 0 \\
&= \frac{1}{|T|} = \frac{1}{B},
\end{aligned}$$

where the forth step follows by $\mathbf{E}_T[\omega^{f^\top (t_i - t_j)}] = \mathbf{E}_{t \sim [p]^d}[\omega^{f^\top t}] = 0$, in which $\mathbf{E}_T[\omega^{f^\top (t_i - t_j)}] =$

$\mathbf{E}_{t\sim[p]^d}[\omega^{f^\top t}]$ because $i \neq j$, t_i, t_j are independent and uniformly random distributed in $[p]^d$, $t_i - t_j \sim [p]^d$; $\mathbf{E}_{t\sim[p]^d}[\omega^{f^\top t}] = 0$ follows by by Fact 3.1.8 and f is not a zero vector.

Part 3. For any $f, f' \in [p]^d$, $f \neq f'$,

$$\begin{aligned}
\mathop{\mathbf{E}}_{T}\left[c_f^{[T]} \cdot \overline{c_{f'}^{[T]}}\right] &= \frac{1}{|T|^2} \mathop{\mathbf{E}}_{T}\left[\sum_{i,j\in[B]} \omega^{f^\top t_i - f'^\top t_j}\right] \\
&= \frac{1}{|T|^2}\left(\sum_{i,j\in[B], i\neq j} \mathop{\mathbf{E}}_{T}\left[\omega^{f^\top t_i - f'^\top t_j}\right] + \sum_{i\in[B]} \mathop{\mathbf{E}}_{T}\left[\omega^{(f-f')^\top t_i}\right]\right) \\
&= \frac{1}{|T|^2}\left(\sum_{i,j\in[B], i\neq j} \mathop{\mathbf{E}}_{t_i\sim[p]^d}\left[\omega^{f^\top t_i}\right] \mathop{\mathbf{E}}_{t_j\sim[p]^d}\left[\omega^{-f'^\top t_j}\right] + \sum_{i\in[B]} \mathop{\mathbf{E}}_{t_i\sim[p]^d}\left[\omega^{(f-f')^\top t_i}\right]\right) \\
&= 0,
\end{aligned}$$

where the second step follows from separating diagonal term and off-diagonal terms, the third step follows from t_i and t_j are independent, the last step follows from Fact 3.1.8 where $f - f' \neq \vec{0}$, and at least one of f and f' is not $\vec{0}$.

□

Let T be a list of independent and uniformly random elements from $[p]^d$. We are going to measure x_t for $t \in T$, and take $\widehat{x}_f^{[T]}$ (recall its definition in Definition 3.1.4) as estimate to $\widehat{x}_f$. By Lemma 3.1.10, $\widehat{x}_f^{[T]} = \sum_{f'\in[p]^d} c_{f-f'}^{[T]} \widehat{x}_{f'}$. The following lemma bounds the contribution of coordinates from V where $V \subseteq [p]^d \setminus \{f\}$, namely $|\sum_{f'\in V} c_{f-f'}^{[T]} \widehat{x}_{f'}|$. When analyzing the quality of $\widehat{x}_f^{[T]}$ as an approximation to $\widehat{x}_f$, we consider coordinates in V as noise, and we usually set $V = [p]^d \setminus \{f\}$.

Lemma 3.1.12 (noise bound). *For any $f \in [p]^d$, T which is a list of B i.i.d. samples from $[p]^d$ and $V \subseteq [n]$ such that $f \notin V$,*

$$\Pr_T\left[\left|\sum_{f'\in V} c_{f-f'}^{[T]} \widehat{x}_{f'}\right| \geq \frac{10}{\sqrt{B}} \|\widehat{x}_V\|_2\right] \leq \frac{1}{100}.$$

Proof. First, we can prove that $\mathbf{E}_T \left[\left| \sum_{f' \in V} c^{[T]}_{f-f'} \widehat{x}_{f'} \right|^2 \right] = \frac{1}{B} \| \widehat{x}_V \|_2^2$, because

$$\begin{aligned}
\mathop{\mathbf{E}}_{T} \left[\left| \sum_{f' \in V} c^{[T]}_{f-f'} \widehat{x}_{f'} \right|^2 \right] &= \mathop{\mathbf{E}}_{T} \left[\sum_{f_1, f_2 \in V} (c^{[T]}_{f-f_1} \widehat{x}_{f_1}) \overline{(c^{[T]}_{f-f_2} \widehat{x}_{f_2})} \right] \\
&= \sum_{f_1, f_2 \in V} \mathop{\mathbf{E}}_{T} \left[c^{[T]}_{f-f_1} \overline{c^{[T]}_{f-f_2}} \right] \widehat{x}_{f_1} \overline{\widehat{x}_{f_2}} \\
&= \sum_{f' \in V} \mathop{\mathbf{E}}_{T} \left[\left| c^{[T]}_{f-f'} \right|^2 \right] |\widehat{x}_{f'}|^2 \\
&= \frac{1}{B} \| \widehat{x}_V \|_2^2,
\end{aligned}$$

where the third step follows from Lemma 3.1.11 that for $f - f_1 \neq f - f_2$, $\mathbf{E}_T \left[c^{[T]}_{f-f_1} \overline{c^{[T]}_{f-f_2}} \right] = 0$, and the last step follows from $\mathbf{E}_T \left[\left| c^{[T]}_{f-f'} \right|^2 \right] = 1/B$ in Lemma 3.1.11.

Then the lemma follows from Chebyshev Inequality and the fact that

$$\mathop{\mathbf{Var}}_{T} \left[\left| \sum_{f' \in V} c^{[T]}_{f-f'} \widehat{x}_{f'} \right| \right] \leq \mathop{\mathbf{E}}_{T} \left[\left| \sum_{f' \in V} c^{[T]}_{f-f'} \widehat{x}_{f'} \right|^2 \right] = \frac{1}{B} \| \widehat{x}_V \|_2^2.$$

□

In the next lemma, we show the guarantee of LinfinityReduce in Algorithm 11.

Lemma 3.1.13 (guarantee of LinfinityReduce in Algorithm 11). *Let $x \in \mathbb{C}^{[p]^d}$, and $n = p^d$. Let $R = C_R \log n$, and $B = C_B k$. Let $C_B \geq 10^6$ and $C_R \geq 10^3$. Let $\mu = \frac{1}{\sqrt{k}} \| \widehat{x}_{-k} \|_2$, and $\nu \geq \mu$. For $r \in [R]$, let $\mathcal{T}_r$ be a list of B i.i.d. elements in $[p]^d$. Let $z \in \mathbb{C}^n$ denote the output of*

$$\textsc{LinfinityReduce}(x, n, y, \{\mathcal{T}_r\}_{r=1}^R, \nu).$$

Let S be top $C_S k$ coordinates in $\widehat{x}$, where $C_S = 26$. If $\| \widehat{x} - y \|_\infty \leq 2\nu$, $\mathrm{supp}(y) \subseteq S$ and y is independent from the randomness of $\{\mathcal{T}_r\}_{r=1}^R$, then with probability $1 - 1/\mathrm{poly}(n)$ under the randomness of $\{\mathcal{T}_r\}_{r=1}^R$, $\| \widehat{x} - y - z \|_\infty \leq \nu$ and $\mathrm{supp}(z) \subseteq S$. Moreover, the running time of LinfinityReduce *is $O(n \log^2 n)$.*

Algorithm 11 Procedure for reducing ℓ_∞ norm of the residual signal

```
1: procedure LINFINITYREDUCE($x, n, y, \{\mathcal{T}_r\}_{r=1}^R, \nu$)            ▷ Lemma 3.1.13
2:     Require that $\|\widehat{x} - y\|_\infty \le 2\nu$
3:     Let $w$ be inverse Fourier transform of $y$                       ▷ We have $\widehat{w} = y$
4:     for $r = 1 \to R$ do
5:         for $f = 1 \to n$ do          ▷ Implemented by FFT which takes $O(n \log n)$ time
6:             $u_{f,r} \leftarrow \frac{\sqrt{n}}{|\mathcal{T}_r|} \sum_{t \in \mathcal{T}_r} \omega^{f^\top t}(x_t - w_t)$      ▷ $\omega = e^{2\pi \mathbf{i}/p}, u_{f,r} = \widehat{(x-w)}_f^{[\mathcal{T}_r]}$
7:         end for
8:     end for
9:     for $f = 1 \to n$ do
10:        $\eta = \mathrm{median}_{r \in [R]}\{u_{f,r}\}$                ▷ Take the median coordinate-wise
11:        if $|\eta| \le \nu/2$ then
12:            $z_f \leftarrow \eta$
13:        else
14:            $z_f \leftarrow 0$
15:        end if
16:    end for
17:    return $z$                                          ▷ Guarantee $\|\widehat{x} - y - z\|_\infty \le \nu$
18: end procedure
```

Proof. Note that $\forall f \in \overline{S}$,

$$|\widehat{x}_f| \le \sqrt{\frac{\|\widehat{x}_{-k}\|_2^2}{C_S k - k}} = \frac{1}{5}\mu,$$

where the last step follows from choice of C_S.

Let w denote the inverse Fourier transform of y. Note that on Line 6 in Algorithm 11, for any $f \in [p]^d$ and $r \in [R]$,

$$\begin{aligned} u_{f,r} &= \frac{\sqrt{n}}{|\mathcal{T}_r|} \sum_{t \in \mathcal{T}_r} \omega^{f^\top t}(x_t - w_t) \\ &= \widehat{(x-w)}_f^{[\mathcal{T}_r]} \\ &= \sum_{f' \in [n]} c_{f-f'}^{[\mathcal{T}_r]}(\widehat{x}_{f'} - y_{f'}), \end{aligned}$$

where the second step follows by the notation in Definition 3.1.4, and the third step follows by Lemma 3.1.10. Therefore,

$$\widehat{x}_f - y_f = u_{f,r} - \sum_{f' \in [p]^d \setminus \{f\}} c_{f-f'}^{[\mathcal{T}_r]}(\widehat{x}_{f'} - y_{f'}), \tag{3.1}$$

By Lemma 3.1.12,

$$\Pr_{\mathcal{T}_r}\left[\left|\sum_{f'\in[p]^d\backslash\{f\}} c^{[\mathcal{T}_r]}_{f-f'}(\widehat{x}_{f'}-y_{f'})\right| \geq \frac{10}{\sqrt{B}}\|(\widehat{x}-y)_{[p]^d\backslash\{f\}}\|_2\right] \leq \frac{1}{100}. \tag{3.2}$$

We have

$$\begin{aligned}
\frac{10}{\sqrt{B}}\|(\widehat{x}-y)_{[p]^d\backslash\{f\}}\|_2 &\leq \frac{10}{\sqrt{B}}\left(\|(\widehat{x}-y)_{S\backslash\{f\}}\|_2+\|(\widehat{x}-y)_{\overline{S}\backslash\{f\}}\|_2\right)\\
&\leq \frac{10}{\sqrt{B}}\left(\|\widehat{x}-y\|_\infty\cdot\sqrt{|S|}+\|\widehat{x}_{\overline{S}\backslash\{f\}}\|_2\right)\\
&\leq \frac{10}{\sqrt{B}}\left(2\nu\cdot\sqrt{26k}+\sqrt{k}\mu\right)\\
&\leq \frac{1}{100\sqrt{k}}\left(2\nu\cdot\sqrt{26k}+\sqrt{k}\mu\right)\\
&< 0.12\nu, \qquad (3.3)
\end{aligned}$$

where the first step following by triangle inequality, the second step follows by the assumption that $\operatorname{supp}(y)\subseteq S$, the forth step follows by $C_B \geq 10^6$, the last step follows by $\mu \leq \nu$.

Therefore,

$$\begin{aligned}
\Pr_{\mathcal{T}_r}[|u_{f,r}-(\widehat{x}_f-y_f)|\leq 0.12\nu] &= \Pr_{\mathcal{T}_r}\left[\left|\sum_{f'\in[p]^d\backslash\{f\}} c^{[\mathcal{T}_r]}_{f-f'}(\widehat{x}_{f'}-y_{f'})\right|\leq 0.12\nu\right]\\
&= 1-\Pr_{\mathcal{T}_r}\left[\left|\sum_{f'\in[p]^d\backslash\{f\}} c^{[\mathcal{T}_r]}_{f-f'}(\widehat{x}_{f'}-y_{f'})\right|> 0.12\nu\right]\\
&\geq 1-\Pr_{\mathcal{T}_r}\left[\left|\sum_{f'\in[p]^d\backslash\{f\}} c^{[\mathcal{T}_r]}_{f-f'}(\widehat{x}_{f'}-y_{f'})\right|\geq \frac{10}{\sqrt{B}}\|(\widehat{x}-y)_{[p]^d\backslash\{f\}}\|_2\right]\\
&\geq 1-\frac{1}{100},
\end{aligned}$$

where the first step follows by (3.1), the third step follows by (3.3), and the last step follows by (3.2).

Thus we have

$$\Pr_{\mathcal{T}_r}[u_{f,r}\in\mathcal{B}_\infty(\widehat{x}_f-y_f,0.12\nu)] \geq \Pr_{\mathcal{T}_r}[|u_{f,r}-(\widehat{x}_f-y_f)|\leq 0.12\nu] \geq 1-\frac{1}{100}.$$

Let $\eta_f = \text{median}_{r\in[R]} u_{f,r}$ as on Line 10 in Algorithm 11. By Chernoff bound, with probability $1 - 1/\operatorname{poly}(n)$, more than $\frac{1}{2}R$ elements in $\{u_{f,r}\}_{r=1}^{R}$ are contained in box $\mathcal{B}_\infty(\widehat{x}_f - y_f, 0.12\nu)$, so that $\eta_f \in \mathcal{B}_\infty(\widehat{x}_f - y_f, 0.12\nu)$.

Therefore, we have

$$\Pr[|\eta_f - (\widehat{x}_f - y_f)| \le 0.17\nu] \ge \Pr[|\eta_f - (\widehat{x}_f - y_f)| \le \sqrt{2}\cdot 0.12\nu] \ge 1 - 1/\operatorname{poly}(n).$$

Let E be the event that for all $f \in [p]^d$, $|\eta_f - (\widehat{x}_f - y_f)| \le 0.17\nu$. By a union bound over $f \in [p]^d$, event E happens with probability $1 - 1/\operatorname{poly}(n)$. In the rest of the proof, we condition on event E.

(**Case 1**) For $f \in \overline{S}$, note that

$$|\eta_f| \le 0.17\nu + |\widehat{x}_f - y_f| = 0.17\nu + |\widehat{x}_f| \le 0.17\nu + 0.2\nu = 0.37\nu.$$

According to the if statement between Line 11 and Line 15 in Algorithm 11, z_f will be assigned to 0. Thus $\operatorname{supp}(z) \subseteq S$. In addition, $|\widehat{x}_f - y_f - z_f| = |\widehat{x}_f| \le \mu \le \nu$.

(**Case 2**) For $f \in S$, we have two cases. We prove that $|(\widehat{x}_f - y_f) - z_f| \le \nu$ for both cases.

(**Case 2.1**) $|\eta_f| \le 0.5\nu$. z_f is assigned as 0. Because

$$|\eta_f - (\widehat{x}_f - y_f)| \le 0.17\nu, \quad |\widehat{x}_f - y_f| \le |\eta_f| + 0.17\nu \le 0.67\nu.$$

Therefore,

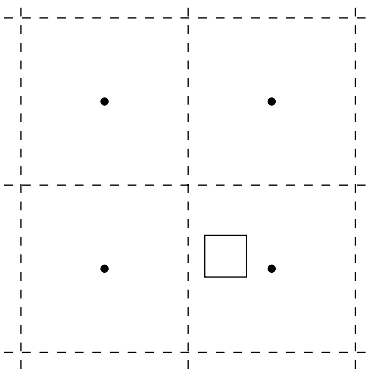
$$|(\widehat{x}_f - y_f) - z_f| \le 0.67\nu \le \nu.$$

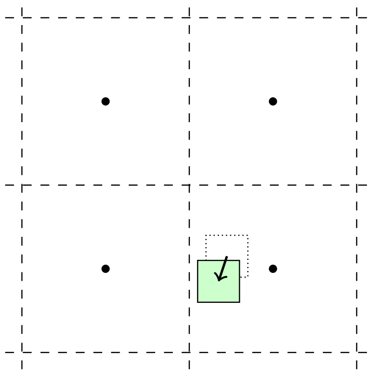
(**Case 2.2**) $|\eta_f| > 0.5\nu$. z_f is assigned as η_f. We have

$$|(\widehat{x}_f - y_f) - z_f| = |(\widehat{x}_f - y_f) - \eta_f| \le 0.17\nu \le \nu.$$

We thus have obtained that with probability $1 - 1/\operatorname{poly}(n)$, $\|(\widehat{x} - y) - z\|_\infty \le \nu$ and $\operatorname{supp}(z) \subseteq S$.

The running time of LinfinityReduce is dominated by the loop between Line 4 and Line 8, which takes $O(R \cdot n \log n) = O(n \log^2 n)$ by FFT. $\square$


(a) a box and grid


(b) a good shift

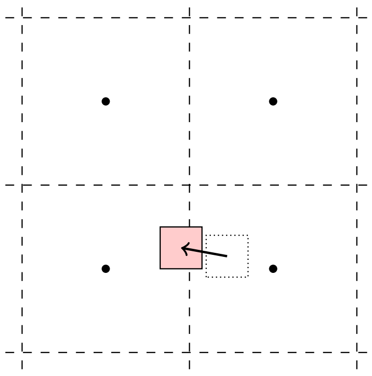

(c) a bad shift

Figure 3.4: *Illustration of good and bad shifts in Definition 3.1.14. In (a), the small square represents box $\mathcal{B}_\infty(c, r_b)$, and the dashed lines represent the decision boundary of Π_{r_g}. The arrows in (b) and (c) represent two different shifts, where the shift in (b) is an example of good shift, since the shifted box does not intersect with the decision boundaries of Π_{r_g}, while the shift in (c) is an example of bad shift, since the shifted box intersects with the decision boundaries of Π_{r_g}.*

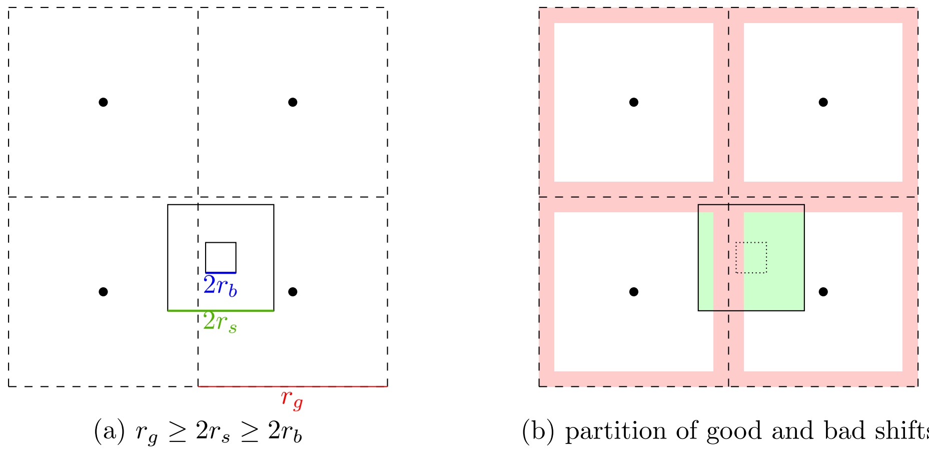
For a given box $\mathcal{B}_\infty(c, r)$ and grid $\mathcal{G}_{r_g}$, we say a shift $s \in \mathbb{C}$ is good if after applying the shift, all the points in the shifted box $\mathcal{B}_\infty(c, r) + s$ are mapped to the same point by Π_{r_g} (recall that Π_{r_g} projects any point to the nearst grid point in $\mathcal{G}_{r_g}$). We formulate the notation of a good shift in the following definition, and illustrate in Figure 3.4.

Definition 3.1.14 (good shift)**.** *For any r_g, r_b, and any $c \in \mathbb{C}$, we say shift $s \in \mathbb{C}$ is a good shift if*

$$\left|\Pi_{r_g}(\mathcal{B}_\infty(c, r_b) + s)\right| = 1.$$

The following lemma intuitively states that if we take a box of radius r_b (or equivalently, side length $2r_b$) and shift it randomly by an offset in $\mathcal{B}_\infty(0, r_s)$ (or equivalently, $[-r_s, r_s] \times [-r_s, r_s]$) where $r_s \geq r_b$, and next we round everyone inside that shifted box to the closest point in G_{r_g} where $r_g \geq 2r_s$, then with probability at least $(1 - r_b/r_s)^2$ everyone will be rounded to the same point. In other words, let $s \sim \mathcal{B}_\infty(0, r_s)$, for box $\mathcal{B}_\infty(c, r_b)$ and grid $\mathcal{G}_{r_g}$, s is a good shift with probability at least $(1 - r_b/r_s)^2$. We illustrate the lemma in Figure 3.5.

Lemma 3.1.15 (property of a randomly shifted box)**.** *For any r_g, r_s, r_b so that $r_g/2 \geq r_s \geq r_b >$*



$2r_b$
$2r_s$
r_g
(a) $r_g \geq 2r_s \geq 2r_b$
(b) partition of good and bad shifts

Figure 3.5: *Illustration of Lemma 3.1.15. In (a) the smallest square represents box $\mathcal{B}_\infty(c, r_b)$, the medium-sized square represents $\mathcal{B}_\infty(c, r_s)$, and the dashed lines represent decision boundaries of Π_{r_g}. Note that for $s \sim \mathcal{B}_\infty(0, r_s)$, the center of the shifted box $s + \mathcal{B}_\infty(c, r_b)$ is $s + c \sim \mathcal{B}_\infty(c, r_s)$. Shift s is good (recall in Definition 3.1.14) for box $\mathcal{B}_\infty(c, r_b)$ and grid $\mathcal{G}_{r_g}$ if and only if the distance between $s + c$ and decision boundaries of Π_{r_g} is greater than r_b. In (b), we draw in red the set of points which are within distance at most r_b to the decision boundaries of Π_{r_g}. Then in (b) the red part inside $\mathcal{B}_\infty(c, r_s)$ corresponds to bad shifts (plus c), and the green part corresponds to good shifts (plus c). Intuitively, the fraction of the green part is at least $(1 - r_b/r_s)^2$ because the vertical red strips can cover a width of at most $2r_b$ on the x-axis of $\mathcal{B}_\infty(c, r_s)$ (whose side length is $2r_s$), and the horizontal red strips can cover a width of at most $2r_b$ on the y-axis.*

0 and any $c \in \mathbb{C}$, let $s \in \mathbb{C}$ be uniform randomly chosen in $\mathcal{B}_\infty(0, r_s)$, then

$$\Pr_{s \sim \mathcal{B}_\infty(0, r_s)} \left[\left| \Pi_{r_g}(\mathcal{B}_\infty(c, r_b) + s) \right| = 1 \right] \geq \left(1 - \frac{r_b}{r_s}\right)^2,$$

where we refer r_g, r_s, r_b as the radius of grid, shift and box respectively, and we use notation $C + s$ to refer to $\{c + s : c \in C\}$.

Proof. We consider complex numbers as points in 2D plane, where the real part is the coordinate on x-axis, and the imaginary part is the coordinate on y-axis. Note that the "decision boundary" of projection Π_{r_g} from $\mathbb{C}$ onto grid $\mathcal{G}_{r_g}$ consists of verticle lines of form $x = (m + \frac{1}{2})r_g$ and horizontal lines of form $y = (m + \frac{1}{2})r_g$, where $m \in \mathbb{Z}$. $\left|\Pi_{r_g}(\mathcal{B}_\infty(c, r_b) + s)\right| = 1$ if and only if the shifted box $\mathcal{B}_\infty(c, r_b) + s$ does not intersect with the "decision boundary".

Let $s = s_x + s_y\mathbf{i}$ and $c = c_x + c_y\mathbf{i}$. Then the shifted box does not intersect with the "decision boundary" if and only if both the interval

$$[c_x - r_b + s_x, c_x + r_b + s_x] \text{ and } [c_y - r_b + s_y, c_y + r_b + s_y]$$

do not intersect with $\{(m + \frac{1}{2})r_g : m \in \mathbb{Z}\}$. The probability of each one is at least $1 - \frac{r_b}{r_s}$, and two events are independent. Therefore, we get the claimed result. □

In the following, we define event $\mathcal{E}$, which is a sufficient condition for the correctness of Algorithm 10. Event $\mathcal{E}$ consists of three parts. Part 1 of $\mathcal{E}$ is used to prove that $a_\ell \le 10 \log n$ for $\ell \in [L-1]$ on Line 29 in Algorithm 10. Part 2 and Part 3 of $\mathcal{E}$ are used to prove that Line 15 to Line 20 in Algorithm 10 give a desirable $z^{(\ell)}$ for $\ell \in [L]$.

Definition 3.1.16 (sufficient condition for the correctness of Algorithm 10)**.** *For input signal $x \in \mathbb{C}^n$, let $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$ and R^* is an upper bound of $\|\widehat{x}\|_\infty/\mu$. Let S be top $C_S k$ coordinates in $\widehat{x}$. Let $H = \min\{\log k + C_H, \log R^*\}$, and $L = \log R^* - H + 1$. For $\ell \in [L]$, let $\nu_\ell = 2^{-\ell}\mu R^*$. For $\ell \in [L-1]$, let $s_\ell^{(a)}$ be the a-th uniform randomly sampled from $\mathcal{B}_\infty(0, \alpha\nu_\ell)$ as appeared on Line 26 in Algorithm 10 (i.e. $s_\ell^{(1)}, \dots, s_\ell^{(a_\ell)}$ are sampled, and $s_\ell^{(a_\ell)}$ is the first that satisfies the condition on Line 28). For the sake of analysis, we assume that Algorithm 10 actually produces an infinite sequence of shifts $s_\ell^{(1)}, s_\ell^{(2)}, \dots$, and chooses the smallest a_ℓ so that $s_\ell^{(a_\ell)}$ satisfies $\forall f \in \operatorname{supp}(y^{(\ell-1)} + z^{(\ell)})$, $|\Pi_{\beta\nu_\ell}(\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}, 2^{1-H}\nu_\ell))| = 1$ on Line 28.*

For $\ell \in [L-1]$, we define random variable a'_ℓ to be the smallest a' such that for all $f \in S$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty(\widehat{x}_f + s_\ell^{(a')}, 2^{3-H}\nu_\ell)\right)\right| = 1.$$

We define event $\mathcal{E}$ to be all of the following events hold.

1. For all $\ell \in [L-1]$, $a'_\ell \le 10 \log n$.

2. For $\ell = 1$, if we execute Line 15 to Line 20 in Algorithm 10 with $y^{(0)} = 0$, we get $z^{(1)}$ such that $\|\widehat{x} - z^{(1)}\|_\infty \le 2^{1-H}\nu_1$ and $\operatorname{supp}(z^{(1)}) \subseteq S$.

3. For all $\ell \in \{2, \dots, L\}$, for all $a \in [10 \log n]$, if we execute Line 15 to Line 20 in Algorithm 10 *with* $y^{(\ell-1)} = \xi$ *where*

$$\xi_f = \begin{cases} \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_{\ell-1}^{(a)}), & \text{if } f \in S; \\ 0, & \text{if } f \in \overline{S}. \end{cases}$$

then we get $z^{(\ell)}$ *such that* $\|\widehat{x} - y^{(\ell-1)} - z^{(\ell)}\|_\infty \le 2^{1-H}\nu_\ell$ *and* $\mathrm{supp}(y^{(\ell-1)} + z^{(\ell)}) \subseteq S$.

In the following, we will prove that for fixed x, under the randomness of $\{s_\ell^{(a)}\}_{\ell\in[L-1],a\in\{1,\ldots\}}$ and $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h\in[H]}$, event $\mathcal{E}$ (defined in Definition 3.1.16) happens with probability at least $1 - 1/\mathrm{poly}(n)$. Moreover, we will prove that event $\mathcal{E}$ is a sufficient condition for the correctness of Algorithm 10. Namely, conditioned on event $\mathcal{E}$, Algorithm 10 gives a desirable output.

Lemma 3.1.17 (event $\mathcal{E}$ happens with high probability)**.** *Let* $\mathcal{E}$ *in Definition 3.1.16. For any fixed* $x \in \mathbb{C}^n$*, under the randomness of shifts* $\{s_\ell^{(a)}\}_{\ell\in[L-1],a\in\{1,\ldots\}}$ *and* $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h\in[H]}$,

$$\Pr[\mathcal{E}] \ge 1 - 1/\mathrm{poly}(n).$$

Proof. We bound the failure probability of each parts in event $\mathcal{E}$ respectively as follows, and $\Pr[\mathcal{E}] \ge 1 - 1/\mathrm{poly}(n)$ follows by a union bound.

Part 1. If $H = \log R^*$, then $L = 1$ and it is trivially true that "for all $\ell \in [L-1], a'_\ell \le 10\log n$". Otherwise, we have $H = \log k + C_H$. By Lemma 3.1.15, for any $\ell \in [L-1]$, for any $f \in S$,

$$\Pr_{s\sim\mathcal{B}_\infty(0,\alpha\nu_\ell)}\left[\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(\widehat{x}_f, 2^{3-H}\nu_\ell\right) + s\right)\right| = 1\right] \ge \left(1 - \frac{2^{3-H}\nu_\ell}{\alpha\nu_\ell}\right)^2 = \left(1 - \frac{2^{3-H}}{\alpha}\right)^2,$$

where $(1 - 2^{3-H}/\alpha)^2 \ge 1 - 2^{4-C_H-\log k}/\alpha \ge 1 - \frac{1}{100k}$ by our choice of α and C_H in Table 3.2.

For each $\ell \in [L-1]$, by a union bound over all f in S, the probability is at least $1 - \frac{C_S k}{100k} = 1 - \frac{26k}{100k} \ge \frac{1}{2}$ that for all $f \in S$, $|\Pi_{\beta\nu_\ell}(\mathcal{B}_\infty(\widehat{x}_f + s, 2^{3-H}\nu_\ell))| = 1$ where $s \sim \mathcal{B}_\infty(0, \alpha\nu_\ell)$. Formally, we get

$$\Pr_{s\sim\mathcal{B}_\infty(0,\alpha\nu_\ell)}\left[\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(\widehat{x}_f, 2^{3-H}\nu_\ell\right) + s\right)\right| = 1, \forall f \in S\right] \ge 1/2.$$

Therefore, by definition of a'_ℓ in Definition 3.1.16,

$$\Pr[a'_\ell \leq 10 \log n] \geq 1 - (1/2)^{10 \log n} = 1 - 1/n^{10}.$$

By a union bound over all $\ell \in [L-1]$, the probability is at least $1 - L/n^{10} = 1 - 1/\operatorname{poly}(n)$ that for all $\ell \in [L-1]$, $a'_\ell \leq 10 \log n$.

Part 2. By Lemma 3.1.13 and a union bound over all $h \in [H]$, the failure probability is at most $H/\operatorname{poly}(n) = 1/\operatorname{poly}(n)$, where $H = O(\log k)$ and so $H/\operatorname{poly}(n)$ is still $1/\operatorname{poly}(n)$.

Part 3. For each $\ell \in \{2, \ldots, L\}$ and $a \in [10 \log n]$, similar to the above argument, each has failure probability at most $1/\operatorname{poly}(n)$. By a union bound, the failure probability is at most

$$(L-1) \cdot (10 \log n)/\operatorname{poly}(n) = 1/\operatorname{poly}(n).$$

$\square$

In the following lemma, we show that if event $\mathcal{E}$ (defined in Definition 3.1.16) happens, then Algorithm 10 gives a desirable output.

Lemma 3.1.18 (correctness of Algorithm 10 conditioned on $\mathcal{E}$)**.** *Let $n = p^d$, and let $k \in [n]$. Let $x \in \mathbb{C}^n$ be input signal. Let $\mu = \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2$. Let $R^* \geq \|\widehat{x}\|_\infty / \mu$ and R^* is a power of 2. Let $H = \min\{\log k + C_H, \log R^*\}$. Let $L = \log R^* - H + 1$. For $\ell \in [L-1]$, let $y^{(\ell)}$ be the vector obtained on Line 31 of Algorithm 10. For $\ell \in [L]$, let $z^{(\ell)}$ be the vector obtained on Line 20. Note that $y^{(0)} = 0$, and $y^{(L-1)} + z^{(L)}$ is the output of* FourierSparseRecovery(x, n, k, R^*, μ) *in Algorithm 10. Conditioned on the event $\mathcal{E}$ (defined in Definition 3.1.16) happens, we have*

$$\|\widehat{x} - y^{(L-1)} - z^{(L)}\|_\infty \leq \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2.$$

Proof. We first discuss the case that $H = \log R^*$. In that case, $L = 1$. Conditioned on the event $\mathcal{E}$ (Part 2 of $\mathcal{E}$), $z^{(1)}$ obtained through Line 15 to Line 20 in Algorithm 10 satisfies $\|\widehat{x} - z^{(1)}\|_\infty \leq 2^{1-H} \nu_1 = 2^{1-H}(2^{-1} \mu R^*) = \mu$.

In the rest of the proof, we discuss the case that $H > \log R^*$. For $\ell \in [L]$, let $\nu_\ell = 2^{-\ell} \mu R^*$. For $\ell \in [L-1]$, let $s_\ell^{(a_\ell)} \in \mathcal{B}_\infty(0, \alpha \nu_\ell)$ denote the first $s_\ell^{(a)}$ on Line 26 in Algorithm 10 such that for all $f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)})$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a)}, 2^{1-H}\nu_\ell\right)\right)\right| = 1.$$

For $\ell \in [L-1]$, we define $\xi^{(\ell)} \in \mathbb{C}^{[p]^d}$ as follows

$$\xi_f^{(\ell)} = \begin{cases} \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell^{(a_\ell)}), & \text{if } f \in S; \\ 0, & \text{if } f \in \overline{S}. \end{cases}$$

We also define $\xi^{(0)} = 0$, $s_0^{(a)} = 0$ for $a \in \{1, \dots\}$ and $a_0 = 1$.

(**Goal: Inductive Hypothesis**) We are going to prove that conditioned on event $\mathcal{E}$ (defined in Definition 3.1.16), for all $\ell \in \{0, \dots, L-1\}$,

$$y^{(\ell)} = \xi^{(\ell)} \quad \text{and} \quad a_\ell \leq 10 \log n.$$

(**Base case**) Note that $y^{(0)} = \xi^{(0)} = 0$ and $a_0 = 1 \leq 10 \log n$.

(**Inductive step**) We will prove that conditioned on event $\mathcal{E}$, if $y^{(\ell-1)} = \xi^{(\ell-1)}$ and $a_{\ell-1} \leq 10 \log n$ for $\ell \in [L-1]$, then $y^{(\ell)} = \xi^{(\ell)}$ and $a_\ell \leq 10 \log n$.

(**Proving** $a_\ell \leq 10 \log n$) Conditioned on event $\mathcal{E}$ (if $L = 1$ then from Part 2 of $\mathcal{E}$, otherwise from Part 3 of $\mathcal{E}$ and by the fact that $a_{\ell-1} \leq 10 \log n$), $z^{(\ell)}$ obtained through Line 15 to Line 20 in Algorithm 10 satisfies $\|\widehat{x} - \xi^{(\ell-1)} - z^{(\ell)}\|_\infty \leq 2^{1-H}\nu_\ell$ and $\mathrm{supp}(z^{(\ell)}) \subseteq S$. Namely, for all $f \in [p]^d$, $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$. Recall the definition of a'_ℓ in Definition 3.1.16. We can prove that $a_\ell \leq a'_\ell$ because if for all $f \in S$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(\widehat{x}_f + s_\ell^{(a'_\ell)}, 2^{3-H}\nu_\ell\right)\right)\right| = 1,$$

then for all $f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)})$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a'_\ell)}, 2^{1-H}\nu_\ell\right)\right)\right| = 1$$

where

$$\mathcal{B}_\infty\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a'_\ell)}, 2^{1-H}\nu_\ell\right) \subseteq \mathcal{B}_\infty\left(\widehat{x}_f + s_\ell^{(a'_\ell)}, 2^{3-H}\nu_\ell\right)$$

which follows by $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$. Therefore, conditioned on $\mathcal{E}$ (Part 1 of $\mathcal{E}$), $a_\ell \le a'_\ell \le 10\log n$.

(**Proving** $y_f^{(\ell)} = \xi_f^{(\ell)}$) For $f \in [p]^d$, we will prove that $y_f^{(\ell)} = \xi_f^{(\ell)}$ in two cases.

(**Case 1**) If $f \in \operatorname{supp}(y^{(\ell-1)} + z^{(\ell)}) \subseteq S$. We have

$$\begin{aligned} y_f^{(\ell)} &= \Pi_{\beta\nu_\ell}\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}\right) \\ &= \Pi_{\beta\nu_\ell}\left(\xi_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}\right). \end{aligned}$$

Because $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$, we have $\xi_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)} \in \mathcal{B}_\infty(\widehat{x}_f + s_\ell^{(a_\ell)}, 2^{1-H}\nu_\ell)$. By the choice of $s_\ell^{(a_\ell)}$, $\Pi_{\beta\nu_\ell}(\xi_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}) = \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell^{(a_\ell)})$. Thus $y_f^{(\ell)} = \xi_f^{(\ell)}$.

(**Case 2**) If $f \notin \operatorname{supp}(y^{(\ell-1)} + z^{(\ell)})$. We have $y_f^{(\ell)} = 0$. Because $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$, we have $|\widehat{x}_f| < 2^{2-H}\nu_\ell < 0.1\beta\nu_\ell$ by our choice of H. We can easily prove that $\xi_f^{(\ell)} = 0 = y_f^{(\ell)}$ in the following two cases:

(**Case 2.1**) If $f \in S$, we have $\xi_f^{(\ell)} = \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell^{(a_\ell)}) = 0$ because

$$|\widehat{x}_f| + |s_\ell^{(a_\ell)}| < 0.1\beta\nu_\ell + 2\alpha\nu_\ell < 0.5\beta\nu_\ell.$$

(**Case 2.2**) If $f \in \overline{S}$, $\xi_f^{(\ell)} = 0$ by definition of $\xi^{(\ell)}$.

Therefore, for all $\ell \in [L-1]$, $y^{(\ell)} = \xi^{(\ell)}$ and $a_\ell \le 10\log n$. Again conditioned on event $\mathcal{E}$ (Part 3 of $\mathcal{E}$), $z^{(L)}$ obtained through Line 15 to Line 20 in Algorithm 10 satisfies

$$\|\widehat{x} - y^{(L-1)} - z^{(L)}\|_\infty \le 2^{1-H}\nu_L = 2^{1-H}(2^{-(\log R^* - H + 1)}\mu R^*) = \mu.$$

Therefore, $y^{(L-1)} + z^{(L)}$ on Line 22 gives a desirable output.

□

Now we present our main theorem, which proves the correctness of Algorithm 10, and shows its sample complexity and time complexity.

Theorem 3.1.19 (main result, formal version). *Let $n = p^d$ where both p and d are positive integers. Let $x \in \mathbb{C}^{[p]^d}$. Let $k \in \{1, \ldots, n\}$. Assume we know $\mu = \frac{1}{k}\|\widehat{x}_{-k}\|_2$ and $R^* \ge \|\widehat{x}\|_\infty/\mu$ where $\log R^* = O(\log n)$. There is an algorithm (Algorithm 10) that takes $O(k\log k\log n)$ samples* *from x, runs in $O(n \log^3 n \log k)$ time, and outputs a $O(k)$-sparse vector y such that*

$$\|\widehat{x} - y\|_\infty \le \frac{1}{\sqrt{k}} \min_{k-\text{sparse } x'} \|\widehat{x} - x'\|_2$$

holds with probability at least $1 - 1/\operatorname{poly}(n)$.

Proof. The correctness of Algorithm 10 follows directly from Lemma 3.1.17 and Lemma 3.1.18. The number of samples from x is

$$B \cdot R \cdot H = O(k \cdot \log n \cdot \log k) = O(k \log k \log n).$$

Its running time is dominated by $L \cdot H = O(\log k \log n)$ invocations of LinfinityReduce (in Algorithm 11). By Lemma 3.1.13, the running time of LinfinityReduce is $O(n \log^2 n)$. Therefore, the running time of Algorithm 10 is

$$O(L \cdot H \cdot n \log^2 n) = O(\log k \cdot \log n \cdot n \log^2 n) = O(n \log^3 n \log k).$$

□

3.2 Deterministic Sparse Fourier Transform with an ℓ_∞ Guarantee

3.2.1 Technical Results

Preliminaries

For a positive integer n, we define $[n] = \{0, 1 \ldots, n-1\}$ and we shall index the coordinates of a n-dimensional vector or the rows/columns of an $n \times n$ matrix from 0 to $n-1$. Let $\omega = e^{-2\pi\sqrt{-1}/n}$ and we define the Discrete Fourier Transform Matrix $F \in \mathbb{C}^{n \times n}$ to be the unitary matrix such that $F_{ij} = \frac{1}{\sqrt{n}} e^{-2\pi\sqrt{-1} \cdot ij/n}$, and the discrete Fourier Transform (DFT) of a vector $x \in \mathbb{C}^n$ to be $\widehat{x} = Fx$.

For a set $S \subseteq [n]$ we define x_S to be the vector obtained from x after zeroing out the coordinates not in S. We also define $H(x, k)$ to be the set of the indices of the largest k coordinates (in magnitude) of x, and $x_{-k} = x_{[n] \setminus H(x,k)}$. We say x is k-sparse if $x_{-k} = 0$. We also define $\|x\|_p = \left(\sum_{i=0}^{n-1} |x_i|^p \right)^{1/p}$ for $p > 1$ and $\|x\|_0$ to be the number of nonzero coordinates of x.

For a matrix $F \in \mathbb{C}^{n\times n}$ and subsets $S, T \subseteq [n]$, we define $F_{S,T}$ to be the submatrix of F indexed by rows in S and columns in T.

The median of a collection of complex numbers $\{z_i\}$ is defined to be $\mathrm{median}_i\, z_i = \mathrm{median}_i\, \Re(z_i) + \sqrt{-1}\,\mathrm{median}_i\, \Im(z_i)$, i.e., taking the median of the real and the imaginary component separately.

ℓ_∞/ℓ_1 **Gurantee and incoherent matrices**

A sparse recovery scheme consists of a measurement matrix $\Phi \in \mathbb{C}^{m\times n}$ and a recovery algorithm $\mathcal{R}$ such that for any given $x \in \mathbb{C}^n$, the scheme approximates x by $\widehat{x} = \mathcal{R}(\Phi x)$. The quality of the approximation is usually measured in different error metrics, and the main recovery guarantee we are interested in is called the ℓ_∞/ℓ_1 guarantee, defined as follows.

Definition 3.2.1 (ℓ_∞/ℓ_1 guarantee). *A sparse recovery scheme is said to satisfy the ℓ_∞/ℓ_1 guarantee with parameter k, if given access to vector x, it outputs a vector $\widehat{x}'$ such that*

$$\|\widehat{x} - \widehat{x}'\|_\infty \le \frac{1}{k}\|\widehat{x}_{-k}\|_1. \tag{3.4}$$

Such scheme is also called a ℓ_∞/ℓ_1 scheme.

Other types of recovery guarantee, such as the ℓ_∞/ℓ_2, the ℓ_2/ℓ_2 and the ℓ_2/ℓ_1, are defined similarly, where (3.4) is replaced with the respective expression in Table 3.4. Note that these are definitions of the error guarantee per se and do not have algorithmic requirements on the scheme.

Highly relevant with the ℓ_∞/ℓ_1 guarantee is a matrix condition which we call incoherence.

Definition 3.2.2 (Incoherent Matrix). *A matrix $A \in \mathbb{C}^{m\times n}$ is called ϵ-incoherent if $\|A_i\|_2 = 1$ for all i (where A_i denotes the i-th column of A) and $|\langle A_i, A_j\rangle| \le \epsilon$.*

Lemma 3.2.3 ([NNW14]). *There exist an absolute constant $c > 0$ such that for any (c/k)-incoherent matrix A, there exists a ℓ_∞/ℓ_1-scheme which uses A as the measurement matrix and*

Guarantee	Formula	Deterministic Lower Bound
ℓ_∞/ℓ_2	$\|\widehat{x}-\widehat{x}'\|_\infty \le \|\widehat{x}_{-k}\|_2/\sqrt{k}$	$\Omega(n)$ [CDD09]
ℓ_2/ℓ_2	$\|\widehat{x}-\widehat{x}'\|_2 \le C\|\widehat{x}_{-k}\|_2$	$\Omega(n)$ [CDD09]
ℓ_∞/ℓ_1	$\|\widehat{x}-\widehat{x}'\|_\infty \le \|\widehat{x}_{-k}\|_1/k$	$\Omega(k^2 + k\log n)$ [Gan08, FPRU10]
ℓ_2/ℓ_1	$\|\widehat{x}-\widehat{x}'\|_2 \le \|\widehat{x}_{-k}\|_1/\sqrt{k}$	$\Omega(k\log(n/k))$ [Gan08, FPRU10]

Table 3.4: *Common guarantees of sparse recovery. Only the ℓ_2/ℓ_2 case requires a parameter $C > 1$. The guarantees are listed in the descending order of strength.*

whose recovery algorithm runs in polynomial time.

The Restrictred Isometry Property and its connection with incoherence

Another highly relevant condition is called the renowned restricted isometry property, introduced by Candès et al. in [CRT06]. We show how incoherent matrices are connected to it.

Definition 3.2.4 (Restricted Isometry Property). *A matrix $A \in \mathbb{C}^{m\times n}$ is said to satisfy the (k,ϵ) Restricted Isometry Property, if for all $x \in \mathbb{C}^n$ with $\|x\|_0 \le k$, its holds that $(1-\epsilon)\|x\|_2 \le \|Ax\|_2 \le (1+\epsilon)\|x\|_2$.*

Candès et al. proved in ther breakthrough paper [CRT06] that any RIP matrix can be used for sparse recovery with the ℓ_2/ℓ_1 error guarantee. The following formulation comes from [FR13, Theorem 6.12].

Lemma 3.2.5. *Given a $(2k,\epsilon)$-RIP matrix A with $\epsilon < 4/\sqrt{41}$, we can design a ℓ_2/ℓ_1-scheme that uses A as the measurement matrix and has a recovery algorithm that runs in polynomial time.*

Although randomly subsampling the DFT matrix gives an RIP matrix with $k\log^2 k\log n$ rows [HR16], no algorithm for finding this rows in polynomial time is known; actually, even for $o(k^2)\cdot \mathrm{poly}(\log n)$ rows the problem remains wide open [4] It is a very important

[4]In fact, one of the results of our paper gives the state of the art result even for this problem, with $O(k^2\log n)$ rows, see Theorem 3.2.11.

	Samples	Run-time	Guarantee	Explicit Construction	Lower Bound
[HR16]	$k \log^2 k \log n$	$\mathrm{poly}(n)$	ℓ_2/ℓ_1	No	$k \log(n/k)$
[MZIC17]	$k^2 \log^{5.5} n$	$k^2 \log^{5.5} n$	ℓ_2/ℓ_1	Yes	$k \log(n/k)$
Theorem 3.2.8	$k^2 \log n$	$nk \log^2 n$	ℓ_∞/ℓ_1	Yes	$k^2 + k \log n$[NNW14]
Theorem 3.2.9	$k^2 \log^2 n$	$k^2 \log^3 n$	ℓ_∞/ℓ_1	Yes	$k^2 + k \log n$ [NNW14]

Table 3.5: *Comparison of our results and the previous results. All O- and Ω-notations are suppressed. The result in the first row follows from Lemma 3.2.5 and the RIP matrix in [HR16].Our algorithms adopt the common assumption in the sparse FT literature that the signal-to-noise ratio is bounded by n^c for some absolute constant $c > 0$.*

and challenging problem whether one can have explicit construction of RIP matrices from Fourier measurements that break the quadratic barrier on k.

We state the following two folklore results, connecting the two different guarantees, and their associated combinatorial objects. This indicates the importance of incoherent matrices for the field of compressed sensing.

Proposition 3.2.6. *An ℓ_∞/ℓ_1 scheme with a measurement matrix of m rows and recovery time T induces an ℓ_2/ℓ_1 scheme of a measurement matrix of $O(m)$ rows and recovery time $O(T + \|\widehat{x}'\|_0)$, where $\widehat{x}'$ is the output of the ℓ_∞/ℓ_1 scheme.*

Proposition 3.2.7. *A (c/k)-incoherent matrix is also a (k, c)-RIP matrix.*

Our results

Theorem 3.2.8. *Let n be a power of 2. There exist a set $S \subseteq [n]$ with $|S| = O(k^2 \log n)$ and an absolute constant $c > 0$ such that the following holds. For any vector $x \in \mathbb{C}^n$ with $\|\widehat{x}\|_\infty \leq n^c \|\widehat{x}_{-k}\|_1/k$, one can find an $O(k)$-sparse vector $\widehat{x}' \in \mathbb{C}^n$ such that*

$$\|\widehat{x} - \widehat{x}'\|_\infty \leq \frac{1}{k} \|\widehat{x}_{-k}\|_1,$$

in time $O(nk \log^2 n)$ by accessing $\{x_i\}_{i \in S}$ only. Moreover, the set S can be found in $\mathrm{poly}(n)$ time.

Theorem 3.2.9. *Let n be a power of 2. There exist a set $S \subseteq [n]$ with $|S| = O(k^2 \log^2 n)$ and an absolute constant $c > 0$ such that the following holds. For any vector $x \in \mathbb{C}^n$ with*

$\|\widehat{x}\|_\infty \le n^c \|\widehat{x}_{-k}\|_1/k$, one can find an $O(k)$-sparse vector $\widehat{x}' \in \mathbb{C}^n$ such that

$$\|\widehat{x} - \widehat{x}'\|_\infty \le \frac{1}{k}\|\widehat{x}_{-k}\|_1,$$

in time $O(k^2 \log^3 n)$ by access to $\{x_i\}_{i\in S}$ only. Moreover, the set S can be found in $\mathrm{poly}(n)$ time.

Remark 3.2.10. *The condition $\|\widehat{x}\|_\infty \le n^c \|\widehat{x}_{-k}\|_1/k$ may seem strange at the first glance, but it upper bounds the "signal-to-noise ratio", a common measure in engineering that compares the level of a desired signal to the level of the background noise. This is a common assumption in most algorithms in the Sparse Fourier Transform literature, see, e.g. [HIKP12a, IK14, Kap16, CKSZ17, Kap17], where the ℓ_2-norm variant $\|\widehat{x}\|_\infty \le n^c \|\widehat{x}_{-k}\|_2/\sqrt{k}$ was assumed.*

We also give the following result on incoherent matrices. The proof can be found in Section 3.2.6.

Theorem 3.2.11. *There exists a set $S \subseteq [n]$ with $|S| = m = O(k^2 \log n)$ such that the matrix $\sqrt{\frac{m}{n}} F_{S,[n]}$ is $(1/k)$-incoherent. Moreover, S can be found in $\mathrm{poly}(n)$ time.*

In Section 3.2.7, we give a construction of explicit $(1/k)$-incoherent matrices via character sums. The number of rows matches the constructions in [DeV07, AM11, DeV07] which were obtained via Gelfand widths, BCH codes and Reed-Solomon codes respectively.

Comparing ℓ_2/ℓ_1 with ℓ_∞/ℓ_1

4 In this subsection we elaborate why ℓ_∞/ℓ_1 is much stronger than ℓ_2/ℓ_1, and not just a guarantee that implies ℓ_2/ℓ_1. Let $\gamma < 1$ be a constant and consider the following scenario. There are three sets A, B, C of size γk, $(1-\gamma)k$, $n-k$ respectively, and for every coordinate in A we have $|\widehat{x}_i| = \frac{2}{k}\|\widehat{x}_C\|_1 = \frac{2}{k}\|\widehat{x}_{-k}\|_1$, while every coordinate in B, C is equal in magnitude. The following relation between the ℓ_1 masses of $\widehat{x}_{B\cup C}$ and $\widehat{x}_C$ is immediate:

$$\|\widehat{x}_C\|_1 = \frac{n-k}{n-\gamma k}\|\widehat{x}_{B\cup C}\|_1.$$

Now assume that $k \le \gamma n$, then $(n-\gamma k)/(n-k) \le 1+\gamma$. We claim that the zero vector is a valid solution for the ℓ_2/ℓ_1 guarantee, since

$$\begin{aligned}\|\vec{0}-\widehat{x}\|_2^2 &= \|\widehat{x}_A\|_2^2 + \|\widehat{x}_{B\cup C}\|_2^2 \\ &\leq \gamma k \cdot \frac{4}{k^2}\|\widehat{x}_{-k}\|_1^2 + \frac{1}{(n-\gamma k)}\|\widehat{x}_{B\cup C}\|_1^2 \\ &\leq \frac{4\gamma}{k}\|\widehat{x}_{-k}\|_1^2 + \frac{n-\gamma k}{(n-k)^2}\|\widehat{x}_C\|_1^2 \\ &\leq \left(\frac{4\gamma}{k}+\frac{1+\gamma}{n-k}\right)\|\widehat{x}_{-k}\|_1^2 \\ &\leq \frac{5\gamma}{k}\|\widehat{x}_{-k}\|_1^2,\end{aligned}$$

where the last inequality follows provided it further holds that $k \leq \gamma n/(2\gamma+1)$. Hence when $\gamma \leq 1/5$, we see that the zero vector satisfies the ℓ_2/ℓ_1 guarantee.

Since $\vec{0}$ is a possible output, we may not recover any of the coordinates in S, which is the set of "interesting" coordinates. On the other hand, the ℓ_∞/ℓ_1 guarantee does allow the recovery of **every** coordinate in S. This is a difference of recovering all γk versus 0 coordinates. We conclude from the discussion above that in the case of too much noise, the ℓ_2/ℓ_1 guarantee becomes much weaker than the ℓ_∞/ℓ_1, possibly giving meaningless results in some cases.

We wish to stress the following remarks, which might help the reader obtain a more complete picture of the results, as well as the difficulties needed to overcome.

Remark 3.2.12. *An inspection of Table 3.5 shows that our sublinear-time algorithm for ℓ_∞/ℓ_1 (Theorem 3.2.9) is closer to its corresponding lower bound than the previous attempts for ℓ_2/ℓ_1. In fact, previous algorithms are at least $k \cdot \mathrm{poly}(\log n)$ factors away from the lower bound while we are only $\log^2 n$ factors away. Ideally, one desires to have an $O(k \cdot \mathrm{poly}(\log n))$ deterministic algorithm for ℓ_2/ℓ_1; however, with the current state of the art techniques, this seems to be way out of reach, see also next remark.*

Remark 3.2.13. *Even if one cared only about sample complexity and not running time, our result is the state-of-the-art for deterministic ℓ_2/ℓ_1. Even with arbitrary measurements, only a mild (but highly technical and important) deterministic construction of ℓ_2/ℓ_1 schemes is known: The breakthrough work of Bourgain et al. uses roughly $O(k^{2-\epsilon_0}\log n)$ measurements [BDF+11], where*

ϵ_0 can be chosen to be approximately $4 \cdot 5.5169 \cdot 10^{-28}$ and has been improved to $4 \cdot 4.4466 \cdot 10^{-24}$ following a sequel of works [Mix].

3.2.2 Technical Toolkit

Properties of the Fourier Transform

Definition 3.2.14 (Convolution of two vectors). *Let $x, y \in \mathbb{R}^N$. The convolution $v = x \star y$ of x, y is the N-dimensional vector defined as $v_i = \sum_{j \in [N]} x_j y_{i-j}$, where the indices of the vectors are taken modulo N, i.e. $y_{-j} = y_{(N-j) \bmod N}$.*

Theorem 3.2.15 (convolution-multiplication duality). *Let $x, y \in \mathbb{C}^N$. It holds that $\widehat{x \star y} = \widehat{x} \odot \widehat{y}$ and $\widehat{x \odot y} = \widehat{x} \star \widehat{y}$, where $\odot$ represents the coordinate-wise product of two vectors, i.e. $(x \odot y)_i = x_i y_i$.*

Hash Functions

Definition 3.2.16 (Frequency domain hashings π, h, o). *Given $\sigma, b \in [n]$, we define a function $\pi_{\sigma,b} : [n] \to [n]$ to be $\pi_{\sigma,b}(f) = \sigma(f - b) \pmod{n}$ for all $f \in [n]$. Define a hash function $h_{\sigma,b} : [n] \to [B]$ as $h_{\sigma,b}(f) = \text{round}((B/n)\pi_{\sigma,b}(f))$ and the off-set functions $o_{f,\sigma,b} : [n] \to [n/B]$ as $o_{f,\sigma,b}(f') = \pi_{\sigma,b}(f') - (n/B)h_{\sigma,b}(f)$. When it is clear from context, we will omit the subscripts σ, b from the above functions.*

In what follows, we might use the notation $H = (\sigma, a, b)$ to denote a tuple of values along with the associated hash function from Definition 3.2.16. Below we define a pseudorandom permutation in the frequency domain.

Definition 3.2.17 ($P_{\sigma,a,b}$). *Suppose that $\sigma^{-1} \mod n$ exists. For $a, b \in [n]$, we define the pseudo-random permutation $P_{\sigma,a,b}$ by $(P_{\sigma,a,b}x)_t = x_{\sigma(t-a)}\omega^{t\sigma b}$.*

Proposition 3.2.18 ([HIKP12a, Claim 2.2]). $(\widehat{P_{\sigma,a,b}x})_{\pi_{\sigma,b}(f)} = \widehat{x}_f \omega^{a\sigma f}$.

Definition 3.2.19 (Sequence of Hashings). *A sequence of d hashings is specified by d tuplets $\{(\sigma_r, a_r, b_r)\}_{r \in [d]}$. For a fixed $r \in [d]$, we will also set π_r, h_r, o_r to be the functions defined in*

Definition 3.2.16, and P_r to be the pseudorandom permutation defined in Definition 3.2.17, by setting $a = a_r, b = b_r, \sigma = \sigma_r$.

Filter Functions

Definition 3.2.20 (Flat filter with B buckets and sharpness F [Kap17])**.** *A sequence $\widehat{G} \in \mathbb{R}^n$ symmetric about zero with Fourier transform $G \in \mathbb{R}^n$ is called a flat filter with B buckets and sharpness F if*
(1) $\widehat{G}_f \in [0,1]$ for all $f \in [n]$;
(2) $\widehat{G}_f \geq 1 - (1/4)^{F-1}$ for all $f \in [n]$ such that $|f| \leq \frac{n}{2B}$;
(3) $\widehat{G}_f \leq (1/4)^{F-1}(\frac{n}{B|f|})^{F-1}$ for all $f \in [n]$ such that $|f| \geq \frac{n}{B}$.

Lemma 3.2.21 (Compactly supported flat filter with B buckets and sharpness F [Kap17])**.** *Fix the integers (n, B, F) with n a power of two, integers $B < n$, and $F \geq 2$ an even integer. There exists an (n, B, F)-flat filter $\widehat{G} \in \mathbb{R}^n$, whose inverse Fourier transform G is supported on a length-$O(FB)$ window centered at zero in time domain.*

Lemma 3.2.22 ([HIKP12b, Lemma 3.6], [HIKP12a, Lemma 2.4], [IK14, Lemma 3.2])**.** *Let $f, f' \in [n]$. Let σ be uniformly random odd number between 1 and $n-1$. Then for all $d \geq 0$ we have $\Pr[|\sigma(f - f')|_\circ \leq d] \leq 4d/n$.*

Formulas for Estimation

Definition 3.2.23 (Measurement)**.** *For a signal $\widehat{x} \in \mathbb{C}^n$, a hashing $H = (\sigma, a, b)$, integers B and F, a measurement vector $m_H \in \mathbb{C}^B$ is the B-dimensional complex-valued vector such that*

$$(m_H)_s = \sum_{f \in [n]} \widehat{G}_{\pi(f)-(n/B)\cdot s} \omega^{a\sigma f} \cdot \widehat{x}_f \in \mathbb{C}$$

for $s \in [B]$. Here $\widehat{G}$ is a filter with B buckets and sharpness F constructed in Definition 3.2.20.

The following lemma provides a HashToBins procedure, which computes the bucket values of the residual $\widehat{x} - \widehat{z}$, where $\widehat{z}$ is also provided as input.

Lemma 3.2.24 (HashToBins [Kap17, Lemma 2.8]). *Let $H = (\sigma, a, b)$ and parameters B, F such that B is a power of 2, and F is an even integer. There exists a deterministic procedure* HashToBins$(x, \widehat{z}, H)$ *which computes $u \in \mathbb{C}^B$ such that for any $f \in [n]$,*

$$u_{h(f)} = \Delta_{h(f)} + \sum_{f' \in [n]} \widehat{G}_{o_f(f')}(\widehat{x} - \widehat{z})_{f'} \omega^{a\sigma f'},$$

where $\widehat{G}$ is the filter defined in Definition 3.2.20, and $\Delta_{h(f)}$ is a negligible error term satisfying $|\Delta_{h(f)}| \le \|z\|_2 \cdot n^{-c}$ for $c > 0$ an arbitrarily large absolute constant. It takes $O(BF)$ samples, and $O(F \cdot B \log B + \|\widehat{z}\|_0 \cdot \log n)$ time.

For a hashing $H = (\sigma, a, b)$, values B, F, and the associated measurement m_H, one has

$$\widehat{G}^{-1}_{o_f(f)} (m_H)_{h(f)} \omega^{-a\sigma f} = \widehat{x}_f + \underbrace{\widehat{G}^{-1}_{o_f(f)} \sum_{f' \in [n] \setminus \{f\}} \widehat{G}_{o_f(f')} \widehat{x}_{f} \omega^{a\sigma(f'-f)}}_{\text{noise term}}. \tag{3.5}$$

A trivial but useful lemma

The following is a basic fact of complex numbers, which will be crucially used in our sublinear-time algorithm, for estimating the phase of a heavy coordinate.

Proposition 3.2.25. *Let $x, y \in \mathbb{C}$ with $|y| \le |x|/3$, then $|\arg(x+y) - \arg x| \le \pi/8$.*

Proof. The worst case occurs when y is orthogonal to x, and thus $|\arg(x+y) - \arg x| \le \arctan(1/3) < \pi/8$. □

3.2.3 Overview

We first show how to obtain for-all schemes, i.e., schemes that allow universal reconstruction of all vectors, and then derandomize them. Similarly to previous work [HIKP12b, IK14, Kap17], we hash, with the filter in [Kap17], the spectrum of x to $O(k)$ buckets using pseudorandom permutations, and repeat $k \log n$ times with fresh randomness. The main part of the algorithm is to show that for any vector $\widehat{x} \in \mathbb{C}^n$ and any set $S \subseteq [n]$ with $|S| \le k$, each $i \in S$, in a constant fraction of the repetitions, receives "low noise" from all other elements, due to this hashing. We show that this boils down to a set of $\Theta(n^2)$ inequalities, which invoke the filter and the pseudorandom permutations. We prove these inequalities with full randomness, and then derandomize the scheme using the method of conditional expectations. For that we choose the pseudorandom permutations one by one, and keep a (rather intricate) pessimistic estimator, which we update accordingly. Our arguments highly extend arguments in [NNW14] and [PR08].

Our sublinear-time algorithm is obtained by bootstrapping the above scheme with an identification procedure in each bucket, as most previous algorithms have done. In contrast to previous approaches, e.g. [HIKP12a], our identification procedure has to be deterministic. We show an explicit set of samples that allow the implementation of the desired routine. To illustrate our idea, let us focus on the following 1-sparse case: $\widehat{x} \in \mathbb{C}^n$ and $|\widehat{x}_{i^*}| \geq 3\|\widehat{x}_{[n]\setminus i^*}\|_1$ for some i^*, which we want to locate. Let

$$\theta_j = \left(\frac{2\pi}{n}j\right) \bmod 2\pi,$$

and consider the $\log n$ samples $x_0, x_1, x_2, x_4, \ldots, x_{2^{r-1}}, \ldots$.

Observe that (ignoring $1/\sqrt{n}$ factors) since

$$x_\beta = \widehat{x}_{i^*} e^{\sqrt{-1}\beta\theta_{i^*}} + \sum_{j \neq i^*} \widehat{x}_j e^{\sqrt{-1}\beta\theta_j},$$

we can find $\beta\theta_{i^*} + \arg \widehat{x}_{i^*}$ up to $\pi/8$, just by estimating the phase of x_β and Fact 3.2.25. Thus we can estimate $\beta\theta_{i^*}$ up to $\pi/4$ from the phase of x_β/x_0. If $i^* \neq j$, then there exists a $\beta \in \{1, 2, 2^2, \ldots, 2^{r-1}, \ldots\}$ such that $|\beta\theta_{i^*} - \beta\theta_j|_o > \pi/2$, and so $\beta\theta_j$ will be more than $\pi/4$ away from the phase of the measurement. Thus, by iterating over all $j \in [n]$, we keep the index j for which $\beta\theta_j$ is close to the $\arg(x_\beta/x_0)$ by $\pi/4$, for every β that is a power of 2 in $\mathbb{Z}_n$.

Unfortunately, although this is a deterministic collection of $O(\log n)$ samples, the above argument gives only $O(n \log n)$ time. For sublinear-time decoding we use x_1/x_0 to find a sector S_0 of the unit circle of length $\pi/4$ that contains θ_{i^*}. Then, from x_2/x_0 we find two sectors of length $\pi/8$ each, the union of which contains θ_{i^*}. Because these sectors are antipodal on the unit circle, the sector S_0 intersects exactly one of those, let the intersection be S_1. The intersection is again a sector of length at most $\pi/8$. Proceeding iteratively, we halve the size of the sector at each step, till we find θ_{i^*}, and infer i^*. Plugging this idea in the whole k-sparse recovery scheme, yields the desired result. Our argument crucially depends on the fact that in the ℓ_1 norm phase of θ_{i^*} will always dominate the phase of all samples we take. On contrast, it totally fails for the ℓ_2 norm, since there is a decent chance (which can even be $1 - 1/n$) that i^* is drowned in the error coming from coordinates $j \neq i^*$.

Our result for incoherent matrices is more general and works for any matrix that has orthonormal columns and entries bounded by $O(1/\sqrt{n})$. We subsample the matrix, invoke a Chernoff bound and Bernstein's inequality to show the small incoherence of the subsampled matrix. We follow a mazy derandomization procedure, which essentially mimics the proof of Bernstein's inequality, by keeping a pessimistic estimator which corresponds to the sum of the generating functions of the probabilities of all events we want to hold, evaluated at specific points. Our second construction of incoherent matrices, involves the use of the infamous Weil bound on character sums. Our construction and its analysis, modulo the (rather complicated) proof of the Weil bound, are much simpler than previous constructions [DeV07, AM11].

3.2.4 Linear-Time Algorithm

Our first step is to obtain a condition that allows us to approximate every coordinate of $x \in \mathbb{C}^n$. This condition corresponds to a set of $n(n-1)$ inequalities. In this section we often considers a sequence of hashings $\{H_r\}_{r\in[d]} = \{(\sigma_r, a_r, b_r)\}_{r\in[d]}$ and for notational simplicity we shall abbreviate $o_{f,\sigma_r,b_r}(f')$ as $o_{f,r}(f')$.

Lemma 3.2.26. *Fix B and F. Let a sequence of hashings $\{H_r\}_{r\in[d]} = \{(\sigma_r, a_r, b_r)\}_{r\in[d]}$ and $x \in \mathbb{C}^n$. We let $o_{f,r} = o_{f,\sigma_r,b_r}$If for all $f, f' \in [n]$ with $f \neq f'$ it holds that*

$$\sum_{r\in[d]} \widehat{G}^{-1}_{o_{f,r}(f)} \widehat{G}_{o_{f,r}(f')} \leq \frac{2d}{B}, \tag{3.6}$$

then for every vector $x \in \mathbb{C}^n$ and every $f \in [n]$, for at least $8d/10$ indices $r \in [d]$ we have that

$$\left|\widehat{x}_f - \widehat{G}^{-1}_{o_{f,r}(f)}(m_{H_r})_{h_r(f)}\right| \le \frac{10}{B}\|\widehat{x}_{[n]\setminus\{f\}}\|_1. \tag{3.7}$$

Proof. We have that

$$\begin{aligned}
\sum_{r\in[d]}\left|\widehat{x}_f - \widehat{G}^{-1}_{o_{f,r}(f)}(m_{H_r})_{h_r(f)}\right| &= \sum_{r\in[d]}\left|\widehat{G}^{-1}_{o_{f,r}(f)}\sum_{f'\in[n]\setminus\{f\}}\widehat{G}_{o_{f,r}(f')}\widehat{x}_{f'}\omega^{a_r\sigma_r(f'-f)}\right| \qquad \text{(by (3.5))}\\
&\le \sum_{r\in[d]}\widehat{G}^{-1}_{o_{f,r}(f)}\sum_{f'\in[n]\setminus\{f\}}\widehat{G}_{o_{f,r}(f')}|\widehat{x}_{f'}|\\
&= \sum_{f'\in[n]\setminus\{f\}}|\widehat{x}_{f'}|\sum_{r\in[d]}\widehat{G}^{-1}_{o_{f,r}(f)}\widehat{G}_{o_{f,r}(f')}\\
&\le \sum_{f'\in[n]\setminus\{f\}}|\widehat{x}_{f'}|\frac{2d}{B}.
\end{aligned}$$

Hence there can be at most $2d/10$ indices $r \in [d]$ for which the estimate $|\widehat{x}_f - \widehat{G}_{o_{f,r}}(f)\cdot m_r(h_r(f))|$ is more than $(10/B)\|\widehat{x}_{[n]\setminus\{f\}}\|_1$, otherwise the leftmost-hand side would be at least $(2d/10+1)\cdot(10/B)\|\widehat{x}_{[n]\setminus\{f\}}\|_1 > 2(d/B)\|\widehat{x}_{[n]\setminus\{f\}}\|_1$. □

The lemma above implies that for every $f \in [n]$ we can find an estimate of $\widehat{x}_f$ up to $\frac{10}{B}\|x_{[n]\setminus\{f\}}\|_1$ in time $O(d)$, by taking the median of all values $m_r(h_r(f))$ for $r \in [d]$. In what follows, we prove the first part of Theorem 3.2.8 (existence of S) assuming that the conditions of Lemma 3.2.26 hold.

For notational simplicity, let $\epsilon = (1/4)^{F-1}$ so the filter $\widehat{G}$ satisfies that $\widehat{G}_{f'} \ge 1-\epsilon$ for all $f' \in [-\frac{n}{2B}, \frac{n}{2B}]$ and $\widehat{G}_{f'} \le \epsilon$ for all $f' \in [n] \setminus (-\frac{n}{B}, \frac{n}{B})$. In the rest of the section, we choose $B = 10(1-\epsilon)^{-1}\beta k$ for some constant β to be determined.

As in previous Fourier sparse recovery papers [HIKP12a, IK14, Kap16, Kap17], we assume that we have the knowledge of $\mu = \|\widehat{x}_{-k}\|_1/k$ (or a constant factor upper bound) and that the signal-to-noise ratio $R^* = \|\widehat{x}\|_1/\mu \le n^\alpha$. Our estimation algorithm is similar to that in [IK14]. The main algorithm is Algorithm 12. It recovers the heavy coordinates of $\widehat{x}$ in increasing magnitude by repeatedly calling the subroutine Algorithm 13, which recovers the heavy coordinates of the residual spectrum above certain threshold.

The following lemmata are analogous to Lemmata 6.1 and 6.2 in [IK14], and their proofs

Algorithm 12 Overall algorithm

```
$T \leftarrow \log_\gamma R^*$
$\widehat{z} \leftarrow 0$
$\nu^{(0)} \leftarrow C\mu$
for $t = 0$ to $T-1$ do
    $\widehat{z} \leftarrow \widehat{z} + \textsc{SubRecovery}(x, \widehat{z}^{(t+1)}, \nu^{(t)})$
    $\nu^{(t+1)} \leftarrow \gamma\nu^{(t)}$
end for
return $\widehat{z}$
```

Algorithm 13 Linear-time Sparse Recovery for $\widehat{x} - \widehat{z}$

```
function $\textsc{SubRecovery}(x, \widehat{z}, \nu)$
    $S \leftarrow \emptyset$
    for $r = 1$ to $d$ do
        $u_r \leftarrow \textsc{HashToBins}(x, \widehat{z}, (\sigma_r, 0, b_r))$
    end for
    for $f \in [n]$ do
        $\widehat{x}'_f = \text{median}_{r\in[d]}\, \widehat{G}^{-1}_{o_{f,r}(f)}(u_r)_{h_r(f)}$        ▷ $o_{f,r} = o_{f,\sigma_r,b_r}$
        if $|\widehat{x}'_f| > \nu/2$ then
            $S \leftarrow S \cup \{f\}$
        end if
    end for
    return $\widehat{x}'_S$
end function
```

are postponed to Section 3.2.8. The first lemma states that Algorithm 13 will recover all the coordinates in the residual spectrum that are at least ν and it will not mistake a small coordinate for a large one.

Lemma 3.2.27. *Suppose that $x, \widehat{z}, \nu$ be the input to Algorithm 13. Let $w = \widehat{x} - \widehat{z}$. When $\nu \geq \frac{16}{\beta k}\|\widehat{w}\|_1$, the output $\widehat{w}'$ of Algorithm 13 satisfies*

1. $|\widehat{w}_f| \geq (7/16)\nu$ *for all* $f \in \text{supp}(\widehat{w}')$.
2. $|\widehat{w}_f - \widehat{w}'_f| \leq |\widehat{w}_f|/7$ *for all* $i \in \text{supp}(\widehat{w}')$;
3. $\text{supp}(\widehat{w}')$ *contains all* f *such that* $|\widehat{w}_f| \geq \nu$;

Next we turn to the analysis of Algorithm 12. Let $H = H(\widehat{x}, k)$ and $I = \{f : |\widehat{x}_f| \geq \frac{1}{\rho k}\|\widehat{x}_{-k}\|_1\}$ for some constant ρ to be determined. By the SNR assumption of $\widehat{x}$, we have that $\|\widehat{x}_H\|_1 \leq k\|\widehat{x}\|_\infty \leq R^*\|\widehat{x}_{-k}\|_1$ and thus $\|\widehat{x}\|_1 \leq (R^*+1)\|\widehat{x}_{-k}\|_1$. In Algorithm 12, the threshold in the t-th step is

$$\nu^{(t)} = C\mu\gamma^{T-t},$$

where $C \geq 1, \gamma > 1$ are constants to be determined. Let $r^{(t)}$ be the residual vector at the beginning of the t-th step in the iteration. We can show that the coordinates we shall ever identify are all heavy (contained in I) and we always have good estimates of them.

Lemma 3.2.28. *There exist C, β, ρ, γ such that it holds for all $0 \leq t \leq T$ that*

1. $\widehat{x}_f = r_f^{(t)}$ *for all* $f \notin I$;
2. $|r_f^{(t)}| \leq |\widehat{x}_f|$ *for all* f.
3. $\|r_I^{(t)}\|_\infty \leq \nu^{(t)}$;

Now we are ready to show the first part of Theorem 3.2.8, which is one of our main results. We shall choose $d = O(k\log n)$ such that (3.2.26) holds. The hashings $\{H_r\}_{r\in[d]}$ can be chosen deterministically, which we shall prove in the rest of the section after this proof; this will complete the full proof.

Proof of Theorem 3.2.8. The recovery guarantee follows immediately from Lemma 3.2.28, as

$$\|r^{(T)}\|_\infty \leq \max\{\|r_I^{(T)}\|_\infty, \|r_{I^c}^{(T)}\|_\infty\} \leq \max\{\nu^{(T)}, \|\widehat{x}_{I^c}\|_\infty\} \leq \max\{2\mu, (1/\rho)\mu\} = 2\mu. \quad (3.8)$$

Computing the measurements in SubRecovery requires $O(k)$ measurements (Lemma 3.2.24). These measurements are reused throughout the iteration in the overall algorithm, hence there are $O(kd) = O(k \cdot k\log n) = O(k^2\log n)$ measurements in total.

Each call to SubRecovery runs in time $O(d(B\log B + \|\widehat{z}\|_0 \log n) + nd) = O(k^2 \log k \log n + k\|\widehat{z}\|_0 \log^2 n + nk\log n)$. By Lemma 3.2.28(a), we know that $\|\widehat{z}\|_0 \leq |I| = O(k)$. The overall runtime is therefore $O(k^2\log k\log n + nk\log^2 n + k^2\log^2 n) = O(k^2\log^2 n + nk\log^2 n) = O(nk\log^2 n)$.

To obtain the ℓ_∞/ℓ_1 error guarantee, or μ on the right-hand side of (3.8) we can just replace k with $2k$ throughout our construction and analysis. □

Derandomization: Pessimistic Estimator

The rest of the section is devoted to finding $\{(\sigma_r, a_r, b_r)\}_{r\in[d]}$ such that (3.6) holds for all pairs $f \neq f'$. It will be crucial for the next section that we can choose a_r freely; that means the inequaltities depend solely on σ_r, b_r. Note that $o_{f,r}(f) \in [-\frac{n}{2B}, \frac{n}{2B}]$ and thus $\widehat{G}_{o_{f,r}(f)} \in [1-\epsilon, 1]$, it suffices to find $\{(\sigma_r, b_r)\}_{r\in[d]}$ such that it holds for all $f \neq f'$ that

$$\sum_{r\in[d]} \widehat{G}_{o_{f,r}(f')} \leq \frac{2}{1+\epsilon} \cdot \frac{d}{B}.$$

To proceed, we derandomize using the method of conditional expectations.

Definition 3.2.29 (Bad Events)**.** *Let $C = 2/(1+\epsilon)$ and $\beta = Cd/B$. Let $A_{f,f'}$ denote the event $\sum_{r=1}^{d} \widehat{G}_{o_{f,r}(f')} \geq \beta$.*

The derandomization proceeds as follows: find a pessimistic estimator $h_r(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r)$ for each r with the first r hash functions fixed by $(\sigma_1, b_1), \ldots, (\sigma_r, b_r)$ such that the following holds:

$$\Pr\left(A_{f,f'} | \sigma_1, b_1, \ldots, \sigma_r, b_r\right) \leq h_r(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r) \tag{3.9}$$

$$\sum_{f\neq f'} h_0(f, f') < 1 \tag{3.10}$$

$$h_r(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r) \geq \mathop{\mathbf{E}}_{\sigma_{r+1}, b_{r+1}} h_{r+1}(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r, \sigma_{r+1}, b_{r+1}) \tag{3.11}$$

The algorithm will start with $r = 0$. At the r-th step, it chooses σ_{r+1}, b_{r+1} to minimize

$$\sum_{f\neq f'} h_{r+1}(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r, \sigma_{r+1}, b_{r+1}).$$

By (3.11), this sum keeps decreasing as r increases. At the end of step $d-1$, all hash functions are fixed, and by (3.9) and (3.10), we have $\sum_{f\neq f'} \Pr(A_{f,f'} | \sigma_1, b_1, \ldots, \sigma_d, b_d) < 1$. Since $A_{f,f'}$ is a deterministic event conditioned on all d hash functions, the conditional probability is either 0 or 1. The inequality above implies that all conditional probabilities are 0, i.e., none of the bad events $A_{f,f'}$ happens, as desired.

We first define our pessimistic estimator. In what follows, we shall be dealing with numbers that might have up to $O(n)$ digits. Manipulating numbers of that length can be done in polynomial time. We will not bother with determining the exact exponent in the polynomial or optimizing it, which we leave to future work.

Definition 3.2.30 (Pessimistic Estimator)**.** *Let $\lambda > 0$ to be determined. Define*

$$h_r(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r) = e^{-\lambda\beta} \exp\left(\lambda \sum_{\ell=1}^{r} \widehat{G}_{o_{f,\ell}(f')}\right) (M(\lambda))^{d-r},$$

where

$$M(\lambda) = e^{\lambda\epsilon}\left[\left(\frac{2}{B} + \frac{1}{n}\right)(e^{\lambda(1-\epsilon)} - 1) + 1\right].$$

This function can be evaluated in $\widetilde{O}(r) \cdot \mathrm{poly}(n)$ time for each pair $f \neq f'$ and thus the algorithm runs in time $\widetilde{O}(n^2 d^2)$.

To complete the proof, we shall verify (3.9)–(3.11) below.

Distribution of Offset Function

This subsection prepares auxiliary lemmata which will be used to verify the derandomization inequalities. In this subsection we focus on the distribution of the offset $o_{f,\sigma,b}(f')$ for $f' \neq f$ and appropriately random σ and b.

Lemma 3.2.31. *Suppose that n, B are powers of 2, σ is uniformly random on the odd integers in $[n]$ and b is uniformly random in $[n]$. For any fixed pair $f \neq f'$ it holds that*

1. *When $(n/B) \nmid (f - f')$, $o_{f,\sigma,b}(f')$ is uniformly distributed on $[n]$;*
2. *When $(f - f')/(n/B)$ is even, $\Pr\{o_{f,\sigma,b}(f') = \ell\} = 0$ for all $\ell \in [-\frac{n}{B}, \frac{n}{B}]$.*
3. *When $(f - f')/(n/B)$ is odd, $\Pr\{o_{f,\sigma,b}(f') = \ell\} = 0$ for $\ell \in [-\frac{n}{2B}, \frac{n}{2B})$ and $\Pr\{o_{f,\sigma,b}(f') = \ell\} = \frac{2}{n}$ for $\ell \in [-\frac{n}{B}, -\frac{n}{2B}) \cup [\frac{n}{2B}, \frac{n}{B}]$.*

Proof. First observe that

$$o_f(f') \equiv \sigma(f' - f) + \sigma(f - b) - \frac{n}{B}\,\mathrm{round}\left(\frac{B}{n}\sigma(f - b)\right) \pmod{n}.$$

For a fixed σ, let

$$Z_\sigma = \sigma(f - b) - \frac{n}{B}\,\mathrm{round}\left(\frac{B}{n}\sigma(f - b)\right).$$

Note that $\sigma(f-b) \bmod n$ is uniform on $[n]$ (for random b), it is easy to see that Z_σ is uniform on its support, which is $[-\frac{n}{2B}, \frac{n}{2B})$.

Suppose that $f'-f \equiv 2^s K \pmod n$, where $K \geq 1$ is an odd integer. It is clear that $\sigma(f'-f)$ is uniform on its support $T = \{2^s \ell \bmod n : \ell \text{ is odd}\}$, which consists of equidistant points. Since Z_σ is always uniform (regardless of σ), and the distribution of $o_f(f') = \sigma(f-f') + Z_\sigma$ is the convolution of two distributions. Suppose that $n = 2^r$ and $B = 2^b$.

When $(n/B) \nmid (f'-f)$, it holds that $r-b \geq s+1$, and thus n/B is an integer multiple of the distance between two consecutive distance in T. In this case it is easy to see that $o_f(f')$ is uniform on $[n]$.

When $(f'-f)/(n/B)$ is even, it must hold that $r-b \leq s-1$ and thus $n/B \leq 2^{s-1}$. The support of $o_f(f')$ is

$$\bigcup_{\text{odd } \ell} \left[2^s\ell - \frac{n}{2B}, 2^s\ell + \frac{n}{2B}\right)$$

which leaves a gap of width at least $2n/B$ in the middle between two consecutive points in T.

When $(f'-f)/(n/B)$ is odd, it must hold that $r-b = s$ and thus $n/B = 2^s$. The support of $o_f(f')$ therefore leaves a gap of width at least n/B in the middle between two consecutive points in T. It is easy to see that $o_f(f')$ is uniform on its support. □

The next theorem, which bounds the moment generating function of $\widehat{G}_{o_f(f')}$, is a straightforward corollary of Lemma 3.2.31.

Lemma 3.2.32. *Let n, σ and b be as in Lemma 3.2.31. When $f \neq f'$, $\mathbf{E}\exp(\lambda \widehat{G}_{o_{f,\sigma,b}(f')}) \leq M(\lambda)$.*

Proof. When $(n/B) \nmid (f-f')$,

$$\mathbf{E}\, e^{\lambda \widehat{G}_{o_f(f')}} \leq \left(\frac{2}{B} + \frac{1}{n}\right) e^{\lambda} + \left(1 - \frac{2}{B} - \frac{1}{n}\right) e^{\lambda\epsilon} = e^{\lambda\epsilon}\left[\left(\frac{2}{B} + \frac{1}{n}\right)(e^{\lambda(1-\epsilon)} - 1) + 1\right],$$

where the inequality follows from the fact that $\widehat{G}$ is at most 1 on $[-n/B, n/B]$ as at most ϵ elsewhere (recall Definition 3.2.20), and the equality from rearranging the terms.

When $f'-f \equiv k(n/B) \pmod n$ for even k,

$$\mathbf{E}\, e^{\lambda \widehat{G}_{o_f(f')}} \leq e^{\lambda\epsilon},$$

since the filter $\widehat{G}$ is at most ϵ outside of $[-n/B, n/B]$ and the distribution $o_f(f')$ is not supported on that interval by Lemma 3.2.31.

When $f' - f \equiv k(n/B) \pmod{n}$ for odd k,

$$\mathbf{E}\, e^{\lambda \widehat{G}_{o_f(f')}} \le \left(\frac{2}{B} + \frac{1}{n}\right) e^{\lambda} + \left(1 - \frac{2}{B} - \frac{1}{n}\right) e^{\lambda\epsilon} = e^{\lambda\epsilon}\left[\left(\frac{2}{B} + \frac{1}{n}\right)(e^{\lambda(1-\epsilon)} - 1) + 1\right],$$

where the inequality follows again by combining Lemma 3.2.31(iii) and the bounds on $\widehat{G}$ from Definition 3.2.20, and the equality is just a rearrangement of terms. □

Finishing the Derandomization

We are now ready to verify (3.9)–(3.11).

Lemma 3.2.33 (Pessimistic Estimation). *It holds that*

$$h_r(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r) \ge \Pr\left(A_{f,f'} | \sigma_1, b_1, \ldots, \sigma_r, b_r\right).$$

Proof. Let $z = \sum_{\ell=1}^{r} G_{o_{f,\ell}(f')}$. Then

$$\begin{aligned}
\Pr\left(A_{f,f'} | \sigma_1, b_1, \ldots, \sigma_r, b_r\right) &= \Pr\left(z + \sum_{\ell=r+1}^{d} G_{o_{f,\ell}(f')} > \beta\right) \\
&= \Pr\left(\exp\left\{\lambda\left(z + \sum_{\ell=r+1}^{d} G_{o_{f,\ell}(f')}\right)\right\} > e^{\lambda\beta}\right) \\
&\le e^{-\lambda\beta} e^{\lambda z}\, \mathbf{E} \exp\left(\lambda \sum_{\ell=r+1}^{d} G_{o_{f,\ell}(f')}\right) \\
&= e^{-\lambda\beta} e^{\lambda z} (\mathbf{E} \exp(\lambda G_{o_f(f')}))^{d-r} \\
&\le e^{-\lambda\beta} e^{\lambda z} (M(\lambda))^{d-r},
\end{aligned}$$

where the last inequality follows from Lemma 3.2.32. □

Lemma 3.2.34 (Initial constraint). *It holds that*

$$\sum_{f \ne f'} h_0(f, f') < 1.$$

Proof. It follows from Lemma 3.2.32 that

$$\begin{aligned}
(M(\lambda))^d &\le \exp\left\{d\left(\lambda\epsilon + \ln\left(1+\frac{3}{B}(e^{\lambda(1-\epsilon)}-1)\right)\right)\right\} \\
&\le \exp\left\{d\left(\lambda\epsilon + \frac{3}{B}(e^{\lambda(1-\epsilon)}-1)\right)\right\} \\
&\le \exp\left\{d\lambda\left(\epsilon + \frac{3}{B}(1-\epsilon)\right)\right\} \\
&\le \exp(3d\lambda).
\end{aligned}$$

Recall that we choose $B = \Theta(k)$ and $d = O(k \log n)$. It follows that

$$\begin{aligned}
\sum_{f \neq f'} h_0(f, f') &= e^{-\lambda\beta} \sum_{f \neq f'} (M(\lambda))^d \\
&\le n^2 \exp\left\{-C\frac{d}{B} + 3d\lambda\right\} \\
&\le n^2 \exp(-cd/B) \qquad \text{(by choosing } \lambda = c''/B \text{ for } c'' \text{ small enough)} \\
&< 1.
\end{aligned}$$

□

Lemma 3.2.35 (Derandomization step)**.** *It holds that*

$$h_r(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r) \ge \mathop{\mathbf{E}}_{\sigma_{r+1}, b_{r+1}} h_{r+1}(f, f'; \sigma_1, b_1, \ldots, \sigma_r, b_r, \sigma_{r+1}, b_{r+1})$$

Proof. Let $z = \sum_{\ell=1}^{r} G^{(\ell)}_{o_{f,r}(f')}$. The proposition is equivalent to

$$\exp(\lambda z)\,(M(\lambda))^{d-r} \ge \mathop{\mathbf{E}}_{\sigma,q} \exp\left(\lambda\left(z + G_{o_{f,r}(f')}\right)\right)(M(\lambda))^{d-r-1},$$

This clearly holds by Lemma 3.2.32. □

3.2.5 Sublinear-Time Algorithm

In this section, we take the pseudorandom hashings $\{H_r\}_{r\in[d]}$ to be as in Lemma 3.2.26 and assume that (3.6) holds.

The first lemma concerns 1-sparse recovery, because, as in earlier works, we shall create k subsignals using hashing, most of which are 1-sparse.

Lemma 3.2.36. *Suppose that n is a power of 2. Let $Q = \{0, 1, 2, 4, \dots, n/2\} \subseteq [n]$. Then the following holds: Let $x \in \mathbb{C}^n$ and suppose that $|\widehat{x}_f| \geq 3\|\widehat{x}_{[n]\setminus\{f\}}\|$ for some $f \in [n]$. Then one can recover the frequency f from the samples x_Q in $O(\log n)$ time.*

Proof. Define $\theta_{f'} = \left(\frac{2\pi}{n} f'\right) \bmod 2\pi$. Observe that

$$x_q = \frac{1}{\sqrt{n}} \left(\widehat{x}_f e^{\sqrt{-1} q \theta_f} + \sum_{f' \neq f} \widehat{x}_{f'} e^{\sqrt{-1} q \theta_{f'}} \right), \quad q \in [n],$$

It follows from Proposition 3.2.25 that $|\arg x_q - (\arg x_f + q\theta_f)| \leq \pi/8$. When $q = 0$, one has $|\arg x_0 - \arg x_f| \leq \pi/8$, and thus $|\arg(x_q/x_0) - q\theta_f| \leq \pi/4$.

Hence,

$$\theta_f \in I_q, \quad \text{where } I_q := \bigcup_{\ell=0}^{q-1} \left[\frac{2\ell\pi + \arg(x_q/x_0)}{q} - \frac{\pi}{4q}, \frac{2\ell\pi + \arg(x_q/x_0)}{q} + \frac{\pi}{4q} \right].$$

Note that I_q is the union of q disjoint intervals of length $\pi/(2q)$. We may view these intervals as arcs on the unit circle, each arc being of length $\pi/(2q)$, and the left endpoints of every two consecutive arcs having distance $2\pi/q$.

Define a series of intervals $\{S_r\}$ for $r = 0, 1, \dots, \log n - 1$ recursively as

$$S_0 = I_1,$$
$$S_{r+1} = S_r \cap I_{2^{r+1}}.$$

It is easy to see, via an inductive argument, that $\theta_f \in S_r$ for all $0 \leq r \leq \log n - 1$, and $|S_r| \leq \frac{\pi}{2^{r+1}}$. In the end, $S_{\log n - 1}$ is an interval of length $\pi/(2n)$, which can contain only one $\theta_{f'}$, and thus we can recover i.

Each S_r can be computed in $O(1)$ time from S_{r-1} and thus the overall runtime is $O(\log n)$. □

Now we move to develop our sublinear-time algorithm. The following is an immediate corollary of Lemma 3.2.26.

Lemma 3.2.37. *For each f, it holds for at least $8d/10$ indices $r \in [d]$ that*

$$\left| \sum_{f' \in [n] \setminus \{f\}} \widehat{G}_{o_{f,r}(f')} \widehat{x}_f \right| \leq \frac{10}{(1-\epsilon)B} \left\| \widehat{x}_{[n]\setminus\{f\}} \right\|_1 .$$

Proof. It follows from Lemma 3.2.26, Eq. (3.5) and the observation that $G_{o_{f,r}(f)} \in [1-\epsilon, 1]$. □

As before, we choose $B = 10(1-\epsilon)^{-1}\beta k$ for some constant β to be determined. The following is a lemma for Algorithm 14, which gives the same guarantees as Lemma 3.2.27.

Lemma 3.2.38. *Suppose that $x, \widehat{z}, \nu$ be the input to Algorithm 14. Let $w = \widehat{x} - \widehat{z}$. When $\nu \geq \frac{16}{\beta k}\|\widehat{w}\|_1$, the output $\widehat{w}'$ of Algorithm 14 satisfies*

1. $|\widehat{w}_f| \geq (7/16)\nu$ *for all* $f \in \operatorname{supp}(\widehat{w}')$.
2. $|\widehat{w}_f - \widehat{w}'_f| \leq |\widehat{w}_f|/7$ *for all* $i \in \operatorname{supp}(\widehat{w}')$;
3. $\operatorname{supp}(\widehat{w}')$ *contains all f such that* $|\widehat{w}_f| \geq \nu$;

Proof. The proof of (i) and (ii) are the same as the proof of Lemma 3.2.27. Next we prove (iii). When $|\widehat{w}_f| \geq \nu$, we have

$$|\widehat{G}_{o_{f,r}(i)} \widehat{w}_f| \geq (1-\epsilon)\nu \geq \frac{16(1-\epsilon)}{\beta k} \|\widehat{w}_{[n]\setminus\{f\}}\|_1.$$

Hence for the signal $y_r \in \mathbb{C}^n$ defined via its Fourier coefficients as

$$(\widehat{y}_r)_{f'} = \widehat{G}_{o_{f,r}(f')} \widehat{x}_{f'},$$

By Lemma 3.2.37, since $16(1-\epsilon) \geq 3$, we see that y_r with frequency f satisfies the condition of Lemma 3.2.36 and thus it will be recovered in at least $8d/10$ repetitions $r \in [d]$. The measurements are exactly $(m_H)_{h(f)}$ with $q \in Q$. The thresholding argument is the same as in the proof of Lemma 3.2.27. □

Observe that Lemma 3.2.28 continues to hold if we replace Algorithm 13 with Algorithm 14 and Lemma 3.2.27 with Lemma 3.2.38. Now we are ready to prove our main theorem, Theorem 3.2.9, on the sublinear-time algorithm.

Proof of Theorem 3.2.9. The recovery guarantee follows identically as in the proof of Theorem 3.2.8.

The measurements are u_q for $q \in Q$ in each of the d repetitions, and calculating each u_q requires $O(k)$ measurements (Lemma 3.2.24). There measurements are reused throughout the iteration in the overall algorithm, hence there are $O(kd|Q|) = O(k \cdot k \log n \cdot \log n) = O(k^2 \log^2 n)$ measurements in total.

Each call to SubRecovery runs in time $O(d(B \log B + \|\widehat{z}\|_0 \log n + B \log n) + kd) = O(k^2 \log^2 n + k\|\widehat{z}\|_0 \log^2 n) = O(k^2 \log^2 n)$, where we use the fact that $\|\widehat{z}\| = O(k)$ from Lemma 3.2.28(a). The overall runtime is therefore $O(k^2 \log^3 n)$. □

Algorithm 14 Sublinear-time Sparse Recovery for $\widehat{x} - \widehat{z}$

```
procedure SubRecovery($x, \widehat{z}, \nu$)
    $\Lambda = \emptyset$
    for $r = 1$ to $d$ do
        for each $q \in Q$ do                                  ▷ $Q$ as in Lemma 3.2.36
            $u_q \leftarrow$ HashToBins$(x, \widehat{z}, (\sigma_r, q, b_r))$
        end for
        for $b = 1$ to $B$ do
            $f \leftarrow$ OneSparseRecovery$(\{(u_q)_b\}_{q \in Q})$
            $\Lambda = \Lambda \cup \{f\}$
            $v_{f,r} = (u_0)_{h_r(f)}$
        end for
    end for
    $\widehat{w}' \leftarrow 0$
    for each $f \in \Lambda$ do
        $v_f \leftarrow \text{median}_r\, v_{f,r}$          ▷ median is taken over all $r$ such that $v_{f,r}$ exists
        if $|v_f| \geq \nu/2$ then
            $\widehat{w}'_f \leftarrow v_f$
        end if
    end for
    return $\widehat{w}'$
end procedure
```

3.2.6 Incoherent Matrices via Subsampling DFT Matrix

Consider an $N \times N$ unitary matrix A and assume that $|A_{i,j}| \leq C/\sqrt{n}$ for all i, j. Our goal in this section is to show how to sample deterministically $m = C_m k^2 \log n$ rows of A, and re-weight them by $\sqrt{\frac{n}{m}}$, obtaining a matrix B, such that $|\langle B_i, B_j\rangle| \leq 1/k$ for all pairs $i \neq j$.

Let $\delta_1, \dots, \delta_n$ be i.i.d. Bernoulli variables with $\Pr(\delta_\ell = 1) = p$, for some $p = m/n$. Let $i, j \in [n]$ such that $i \neq j$, then

$$\langle B_i, B_j\rangle = \sum_\ell \delta_\ell A_{\ell,i}\overline{A}_{\ell,j}.$$

Let $z_\ell = A_{\ell,i}\overline{A}_{\ell,j}$, then $|z_\ell| \leq \eta$, where $\eta = C^2/n$. We consider the real and the imaginary parts separately, since for a complex random variable Z,

$$\Pr(|Z| > t) \leq \Pr\left(|\Re Z| > \frac{t}{\sqrt{2}}\right) + \Pr\left(|\Im Z| > \frac{t}{\sqrt{2}}\right).$$

Hence it suffices to consider the real variable problem as follows. Suppose that $a_1, \dots, a_n \in \mathbb{R}$ satisfy $|a_i| \leq \eta$, and consider the centred sum $S = \sum_i (\delta_i - p)a_i$. We wish to find $\delta_1, \dots, \delta_n$ deterministically such that $|S| \leq cm/(kn)$, where $c > 0$ is an absolute constant to be determined.

Define the pessimistic estimator to be

$$f_r(\delta_1, \dots, \delta_r) = e^{-\lambda t}\left(e^{\lambda \sum_{i=1}^r (\delta_i - p)a_i} \prod_{i=r+1}^n M_i(\lambda) + e^{-\lambda \sum_{i=1}^r (\delta_i - p)a_i} \prod_{i=r+1}^n M_i(-\lambda)\right)$$

The moment generating function of $(\delta_i - p)a_i$ is

$$M_i(\lambda) = pe^{\lambda(1-p)a_i} + (1-p)e^{-\lambda p a_i}, \quad i = 1, \dots, n.$$

Pessimistic Estimation Let $w = \sum_{i=1}^r (\delta_i - p)a_i$, where $\delta_1, \dots, \delta_r$ have been fixed.

$$\begin{aligned}
\Pr(|S| > t|\delta_1, \dots, \delta_r) &= \Pr(S > t|\delta_1, \dots, \delta_r) + \Pr(-S > t|\delta_1, \dots, \delta_r) \\
&= \Pr(e^{\lambda S} > e^{\lambda t}|\delta_1, \dots, \delta_r) + \Pr(e^{-\lambda S} > e^{\lambda t}|\delta_1, \dots, \delta_r) \\
&\leq e^{-\lambda t}\,\mathbf{E}(e^{\lambda S} + e^{-\lambda S}|\delta_1, \dots, \delta_r) \\
&= e^{-\lambda t}\left(e^{\lambda w} \prod_{i=r+1}^n M_i(\lambda) + e^{-\lambda w} \prod_{i=r+1}^n M_i(-\lambda)\right) \\
&= f_r(\delta_1, \dots, \delta_r).
\end{aligned}$$

Derandomization step One can show first that

$$f_r(\delta_1,\ldots,\delta_r) = pf_r(\delta_1,\ldots,\delta_r,1) + (1-p)f_r(\delta_1,\ldots,\delta_r,0), \tag{3.12}$$

which is equivalent to

$$e^{\lambda w}M_{r+1}(\lambda)\prod_{i=r+2}^{n} M_i(\lambda) + e^{\lambda w}M_{r+1}(-\lambda)\prod_{i=r+2}^{n} M_i(-\lambda) = pM' + (1-p)M'', \tag{3.13}$$

where

$$\begin{aligned} M' &= e^{\lambda(w+(1-p)a_i)}\prod_{i=r+2}^{n} M_i(\lambda) + e^{-\lambda(w+(1-p)a_i)}\prod_{i=r+2}^{n} M_i(-\lambda), \\ M'' &= e^{\lambda(w-pa_i)}\prod_{i=r+2}^{n} M_i(\lambda) + e^{-\lambda(w-pa_i)}\prod_{i=r+2}^{n} M_i(-\lambda). \end{aligned}$$

It is now clear that the left-hand side of (3.13) is $pM' + (1-p)M''$, and therefore (3.12) holds. This implies that

$$f_r(\delta_1,\ldots,\delta_r) \geq \min\{f_r(\delta_1,\ldots,\delta_r,1), f_r(\delta_1,\ldots,\delta_r,0)\}.$$

Initial condition This is a standard argument for Bernstein's inequality. For notational convenience, let $\phi(x) = (e^{\lambda x} - \lambda x - 1)/x^2$. Note that $\phi(x)$ is increasing on $(0,\infty)$. Using Taylor's expansion, one can bound that (see [BLM13, p35])

$$M_i(\lambda) \leq \exp\left(\phi(|a_i|)p(1-p)a_i^2\right) \leq \exp\left(\phi(\eta)p(1-p)a_i^2\right).$$

and (see [Tro15, p98])

$$\phi(\eta) \leq \frac{\lambda^2/2}{1-\lambda\eta/3}, \quad \lambda < \frac{3}{\eta}.$$

It then follows (see [Tro15, p98]) that

$$\begin{aligned} \Pr(|S| > t) &\leq 2e^{-\lambda t}e^{\phi(\eta)p(1-p)\sum_i |a_i|^2} \\ &\leq 2\exp\left(-\lambda t + n\eta^2 p(1-p)\frac{\lambda^2/2}{1-\lambda\eta/3}\right) \\ &\leq 2\exp\left(-\frac{t^2/2}{n\eta^2 p(1-p) + t\eta/3}\right), \end{aligned}$$

provided that $\lambda = t/(n\eta^2 p(1-p) + t\eta/3) \in (0, 3/\eta)$.

When $t = m/(kn)$, $p = m/n$ and $\eta = C^2/n$, $\lambda \simeq \log n < 3/\eta$ and the above probability is at most

$$2\exp\left(-\frac{1}{2(C^4 + \frac{c}{3})} \cdot \frac{m}{k^2}\right) \le 2\exp(-c' C_m \log n) \le \frac{1}{n^3},$$

provided that C_m is large enough.

Therefore at step r, the algorithm minimizes $f_{r+1}(\delta_1, \ldots, \delta_{r+1})$ by choosing δ_{r+1}, and at the end of step $r+1$, all $\delta_1, \ldots, \delta_r$ have been fixed and such that $|\sum_i (\delta_i - p)a_i| \le t$.

Now we return to the original incoherence problem in the complex case. We can define $2n(n-1)$ events, $E_{i,j}$ and $F_{i,j}$, for every pair $i \ne j$ as

$$E_{i,j} = \left\{ |\Re\langle B_i, B_j\rangle| > t \right\}, \quad F_{i,j} = \left\{ |\Im\langle B_i, B_j\rangle| > t \right\}$$

For each pair of $i \ne j$, using the preceding argument, we have pessimistic estimators $f_r^1(i,j;\delta_1, \ldots, \delta_r)$ by setting $a_\ell = \Re B_{i,\ell}\overline{B_{\ell,i}}$ and $f_r^2(i,j;\delta_1, \ldots, \delta_r)$ by setting $a_\ell = \Im B_{i,\ell}\overline{B_{\ell,j}}$ such that

- (pessimistic estimation)

$$f_r^1(i,j;\delta_1, \ldots, \delta_r) \ge \Pr(E_{i,j}|\delta_1, \ldots, \delta_r)$$
$$f_r^2(i,j;\delta_1, \ldots, \delta_r) \ge \Pr(F_{i,j}|\delta_1, \ldots, \delta_r)$$

- (derandomization step)

$$f_r^s(i,j;\delta_1, \ldots, \delta_r) = p f_{r+1}^s(i,j;\delta_1, \ldots, \delta_r, 1) + (1-p) f_{r+1}^s(i,j;\delta_1, \ldots, \delta_r, 0), \quad s = 1,2 \tag{3.14}$$

- (initial condition)

$$\sum_{i \ne j} f_0^1(i,j) + f_0^2(i,j) < \frac{1}{n}.$$

Note that (3.14) implies

$$\sum_{i\neq j}\left[f_r^1(i,j;\delta_1,\ldots,\delta_r)+f_r^2(i,j;\delta_1,\ldots,\delta_r)\right]$$
$$\geq \min_{\delta_{r+1}\in\{0,1\}}\sum_{i\neq j}\left[f_r^1(i,j;\delta_1,\ldots,\delta_r,\delta_{r+1})+f_r^2(i,j;\delta_1,\ldots,\delta_r,\delta_{r+1})\right].$$

In addition, we also need to control the number of δ_i's which take value 1; we want this number to be $O(m)$. This can be achieved by combining another derandomization procedure on $\sum_i \delta_i$ using one-sided Chernoff bounds. Define the event $G=\{\sum_i \delta_i > 2m\}$. Then

$$\begin{aligned}\Pr(G|\delta_1,\ldots,\delta_r) &\leq \exp\left(-2m\kappa+\kappa\sum_{i=1}^r \delta_i\right)\prod_{i=r+1}^n \mathbf{E}\, e^{\kappa\delta_i}\\ &= \exp\left(-2m\kappa+\kappa\sum_{i=1}^r \delta_i\right)(M(\kappa))^{n-r},\end{aligned}$$

where

$$M(\kappa)=\mathbf{E}\, e^{\kappa\delta_i}=pe^p+1-p$$

is the moment generating function of δ_i. Define our pessimistic estimator to be

$$g_r(\delta_1,\ldots,\delta_r)=\exp\left(-2m\kappa+\kappa\sum_{i=1}^r \delta_i\right)(M(\kappa))^{n-r},$$

then, similar to the proof in Section 3.2.4, we have

- (pessimistic estimation)

$$g(\delta_1,\ldots,\delta_r)\geq \Pr(G|\delta_1,\ldots,\delta_r),$$

- (derandomization step)

$$g_r(i,j;\delta_1,\ldots,\delta_r)\geq pg_{r+1}(i,j;\delta_1,\ldots,\delta_r,1)+(1-p)g_{r+1}(i,j;\delta_1,\ldots,\delta_r,0),$$

- (initial condition)

$$g_0<\frac{1}{2}.$$

Overall, our standard derandomization procedure, which at step r chooses $\delta_{r+1} \in \{0,1\}$ that minimizes

$$\sum_{i \neq j} \left[f^1_{r+1}(i,j;\delta_1,\ldots,\delta_r,\delta_{r+1}) + f^2_{r+1}(i,j;\delta_1,\ldots,\delta_r,\delta_{r+1}) \right] + g_{r+1}(\delta_1,\ldots,\delta_r,\delta_{r+1})$$

will find $\delta_1,\ldots,\delta_r$ such that none of $E_{i,j}$ and $F_{i,j}$ and G holds, which implies that $|\langle B_i, B_j \rangle| \leq t$ for all $i \neq j$ and $\sum \delta_i \leq 2m$. That is, we have chosen $2m$ rows of A, obtaining a matrix B of incoherence at most $m/(kn)$.

3.2.7 Incoherent Matrices via the Weil bound

We shall use the following classical bound of Weil on character sums (see, e.g. [Sch76, p44]).

Theorem 3.2.39 (Weil Bound)**.** *Let q be a prime number and let $\mathbb{F}_q$ be the finite field of order q. Let $g(x)$ be a polynomial of degree $d > 0$ and $\psi : \mathbb{F}_q \to \mathbb{C}^*$ be a nontrivial additive character. If $d < q$ and* $\gcd(d,q) = 1$ *then*

$$\left| \sum_{x \in \mathbb{F}_q} \psi(g(x)) \right| \leq (d-1)\sqrt{q}$$

The function $\psi(x) = e^{(-2\pi\sqrt{-1}rx)/q}$ is a nontrivial additive character for $\mathbb{F}_q = \mathbb{Z}_q$, for each $1 \leq r < q$.

Consider a collection $\mathcal{P}$ of non-zero polynomials of degree at most $d < 1$. The size of $\mathcal{P}$ is $q^{d+1} - 1$. For every such polynomial $g \in \mathcal{P}$ define vector

$$(v_g)_x = \psi(g(x)),$$

for $x \in \mathbb{F}_q$. Observe that the inner product between two vectors v_g and $v_{g'}$ equals to

$$\sum_{x \in \mathbb{F}_q} \psi(g(x))\overline{\psi'(g(x))} = \sum_{x \in \mathbb{F}_q} \psi(g(x) - g'(x)),$$

which, by applying Weil's bound, is bounded in magnitude by $(d-1)\sqrt{q}$.

We construct a matrix $A \in \mathbb{C}^{m \times n}$ whose columns are the vectors $\{v_g\}_{g \in \mathcal{P}}$, re-weighted by $1/\sqrt{m}$, so that they have the unit norm. We set

$$d = \left\lceil c \frac{\log n}{\log k + \log \log n} \right\rceil$$

for some absolute constant c, to obtain

$$m = O\left(k^2 \left(\frac{\log n}{\log k + \log \log n}\right)^2\right).$$

Now, $(d-1)/q = \Theta(1/k)$, and thus the matrix A is (c'/k)-incoherent for some absolute constant c'. We now finish the proof by rescaling k.

Remark 3.2.40. *As we have already mentioned, the lower bound on the number of rows of any incoherent matrix is $\Omega(k \log_k n)$, due to Alon [Alo09]. Our construction above uses $O(k^2 (\log_k n)^2)$ samples. In the regime where k is a fractional power of n, this is always better than the random construction (Theorem 3.2.11) and matches the aforementioend lower bound.*

Remark 3.2.41. *Our construction above has a similar spirit to the Reed-Solomon construction in [NNW14]; both identify the columns of the matrix with the set of all low-degree polynomials over a finite field. However, we need a much stronger and deep result from number theory in order to show the incoherence result, in contrast to [NNW14].*

3.2.8 Reduction of the ℓ_∞ norm

Lemma 3.2.42. *Suppose that $x, \widehat{z}, \nu$ be the input to Algorithm 14. Let $w = \widehat{x} - \widehat{z}$. When $\nu \geq \frac{16}{\beta k}\|\widehat{w}\|_1$, the output $\widehat{w}'$ of Algorithm 14 satisfies*

1. *$|\widehat{w}_f| \geq (7/16)\nu$ for all $i \in \operatorname{supp}(\widehat{w}')$.*

2. *$|\widehat{w}_f - \widehat{w}'_f| \leq |\widehat{w}_f|/7$ for all $i \in \operatorname{supp}(\widehat{w}')$;*

3. *$\operatorname{supp}(\widehat{w}')$ contains all i such that $|\widehat{w}_f| \geq \nu$;*

Proof. By the recovery guarantee we know that

$$|\widehat{w}_f - \widehat{w}'_f| \leq \frac{\|w\|_1}{\beta k} \leq \frac{\nu}{16}.$$

By thresholding, it must hold for $i \in \operatorname{supp}(\widehat{w}')$ that $|\widehat{w}'_f| \geq \nu/2$ and thus

$$|\widehat{w}_f| \geq \frac{\nu}{2} - \frac{\nu}{16} = \frac{7}{16}\nu,$$

which proves (i). Thus

$$|\widehat{w}_f - \widehat{w}'_f| \leq \frac{\nu}{16} \leq \frac{1}{7}|\widehat{w}_f|,$$

which proves (ii). Next we prove (iii). When $|\widehat{w}_f| \geq \nu$, we have

$$|\widehat{G}_{o_{f,r}(f)}\widehat{w}_f| \geq (1-\epsilon)\nu \geq \frac{16(1-\epsilon)}{\beta k}\|\widehat{w}_{[n]\setminus\{f\}}\|_1.$$

Hence for the signal $y_r \in \mathbb{C}^n$ defined via its Fourier coefficients as

$$(\widehat{y}_r)_{f'} = \widehat{G}_{o_{i,r}(j)}\widehat{x}_{f'},$$

By Lemma 3.2.37, since $16(1-\epsilon) \geq 3$, we see that y_r with index i satisfies the condition of Lemma 3.2.36 and thus it will be recovered in at least $8d/10$ indices $r \in [d]$. The measurements are exactly $(m_H)_{h(i)}$ with $q \in Q$. The recovered estimate is at least $\nu - \nu/16 > \nu/2$ and thus the median estimate will pass the thresholding, and $i \in \operatorname{supp}(\widehat{w}')$. □

Let $H = H(x,k)$ and $I = \{f : |\widehat{x}_f| \geq \frac{1}{\rho k}\|x_{-k}\|_1\}$. By the SNR assumption of $\widehat{x}$, we have that $\|\widehat{x}_H\|_1 \leq k\|\widehat{x}\|_\infty \leq R^*\|\widehat{x}_{-k}\|_1$ and thus $\|\widehat{x}\|_1 \leq (R^*+1)\|\widehat{x}_{-k}\|_1$. Let $r^{(t)}$ be the residual vector at the beginning of the t-th step in the iteration. The threshold in the t-th step is

$$\nu^{(t)} = C\mu\gamma^{T-t},$$

where $C \geq 1, \gamma > 1$ are constants to be determined.

Lemma 3.2.43. *There exist C, β, ρ, γ such that it holds for all $0 \leq t \leq T$ that*

1. $\widehat{x}_f = r_f^{(t)}$ *for all* $f \notin I$;
2. $|r_f^{(t)}| \leq |\widehat{x}_f|$ *for all* f.
3. $\|r_f^{(t)}\|_\infty \leq \nu^{(t)}$;

Proof. We prove the three properties inductively. The base case is $t = 0$, where all properties clearly hold, noticing that $\mu\gamma^T = \|x\|_\infty$.

Next we prove the inductive step from t to $t+1$. Note that

$$\begin{aligned}\|r^{(t)}\|_1 &\le \|r_H^{(t)}\|_1 + \|r_{H^c}^{(t)}\|_1 \\ &= \|r_{H\cap I}^{(t)}\|_1 + \|r_{H\setminus I}^{(t)}\|_1 + \|x_{-k}\|_1 \\ &\le k \cdot \|r_I\|_\infty + k \cdot \frac{1}{\rho k}\|x_{-k}\|_1 + \|x_{-k}\|_1 \\ &\le k \cdot C\mu\gamma^{T-t} + \left(1 + \frac{1}{\rho}\right)\|x_{-k}\|_1 \\ &= C\gamma^{T-t}\|x_{-k}\|_1 + \left(1 + \frac{1}{\rho}\right)\|x_{-k}\|_1\end{aligned}$$

When

$$C\left(1 - \frac{16}{\rho}\right) \ge \frac{16}{\beta}\left(1 + \frac{1}{\rho}\right), \tag{3.15}$$

it holds that

$$\nu^{(t)} \ge \frac{16}{\beta k}\|r^{(t)}\|_1$$

and thus Lemma 3.2.42 applies.

From Lemma 3.2.42(i), we know that when

$$\frac{7}{16}C \ge \frac{1}{\rho}, \tag{3.16}$$

no coordinates in I^c will be modified. This proves (a).

Lemma 3.2.42(ii) implies (b).

To prove (c), let $J = \{f \in I : |r_f^{(t)}| \ge \nu^{(t+1)}\}$. By Lemma 3.2.42(iii), all coordinates in J will be recovered. Hence for $f \in J$,

$$|r_f^{(t+1)}| \le \frac{1}{7}|r_f^{(t)}| \le \frac{1}{7}\nu^{(t)} \le \nu^{(t+1)},$$

provided that

$$\frac{1}{7} \le \frac{1}{\gamma}. \tag{3.17}$$

For $f \in I \setminus J$, the definition of J implies that $|r_f^{(t+1)}| \le \nu^{(t+1)}$. This proves (c).

We can take $C = 2$, $\rho = 32$, $\beta = 32$, $\gamma = 2$, which satisfy all the constraints (3.15), (3.16) and (3.17). $\square$

Chapter 4

Sparse Recovery and the Design of Exact Algorithms

4.1 Nearly Optimal Sparse Polynomial Multiplication

4.1.1 Preliminaries

We will be concerned with polynomials with integer coefficients. This suffices for most applications, since numbers in a machine are represented using floating point arithmetic. We denote by $\mathbb{Z}_n$ the ring of residue modulo n and by $[n]$ the set $\{0, 1, \ldots n-1\}$. We define the convolution of two vectors $x, y \in \mathbb{R}^n$ as the n-dimensional vector $x * y$ such that

$$(x * y)_i = \sum_{j,j' \in [n] \times [n] : (j+j') \bmod N = i} x_j y_{j'}.$$

The convolution of two vectors is immediatelly related with polynomial multiplication: if $f(x) = \sum_{j=0}^{n} a_j x^j$ and $g(x) = \sum_{j=0}^{n} \beta_j x^j$, we have that the polynomial $(f \cdot g)(x) = \sum_{j=0}^{2n} c_j x^j$ satisfies $c = a * b$, where $c = (c_0, c_1, \ldots \ldots, c_{2n}, 0 \ldots, 0) \in \mathbb{Z}^N, a = (a_0, a_1, \ldots, a_n, 0, \ldots, 0) \in \mathbb{Z}^N, b = (b_0, b_1, \ldots, b_n, 0, \ldots, 0) \in \mathbb{Z}^N$, for $N \geq 2n$. It is known that $x * y$ can be computed from x and y in time $O(n \log n)$ via the Fast Fourier Transform. Throughout this subchapter we assume that we work on a machine where the word size is $w = \Omega(\log n)$, and elementary operations between two integers given as part of the input can be done in $O(1)$ time. For a complex number z we denote by $|z|$ its magnitude, and by $\arg(\phi)$ its phase.

4.1.2 Result and Proof

In this subsection we prove the following result.

Theorem 4.1.1. *Let $x, y \in \mathbb{Z}^n$, given as lists of their non-zero coordinates along with their values. Set $a = \|x\|_0, b = \|y\|_0, k = \|x * y\|_0 + 4$. Then, with probability $99/100$, we can compute a list which contains the non-zero coefficients and values of $x * y$ in time*

$$O(k \log^2 n \cdot \log(k \log n) + (a+b) \log k \log n \cdot \log\log n) + \widetilde{O}(\log^4 n).$$

We proceed by building the tools needed for the proof of theorem 4.1.1. In what folows ω is an nth root of unity; our algorithm should treat it as rounded in order to fit in the word size, since we are dealing with floating point numbers. We can also assume that n is a prime number. For that, if $n \geq 21$, we may sample $\widetilde{O}(\log n)$ numbers in the interval $[n, 2n]$ and run the Miller-Rabin test to check whether anyone of them is prime. A standard fact about the distribution of primes implies that after $\widetilde{O}(\log n)$ samples, we will find with constant probability such a prime, for a total of $\widetilde{O}(\log^3 n)$ time. Thus, in what follows n is a prime number. The following operator is particularly important for our algorithm.

Definition 4.1.2. *Let $x \in \mathbb{Z}^n$. Define function $h : [n] \to [m]$ by*

$$h_m(i) = i \bmod m.$$

Moreover, define $\mathcal{P}_m(x) \in \mathbb{Z}^m$ to be such that

$$(\mathcal{P}_m(x))_i = \sum_{j \in [n] : h_m(j) = i} x_j \cdot \omega^j, \forall i \in [m].$$

Lemma 4.1.3. *Given vectors $x, y, w \in \mathbb{Z}^n$ the vector $\mathcal{P}_m((x * y) - w)$ can be computed in time $O((\|x\|_0 + \|y\|_0 + \|w\|_0) \log n + m \log m)$.*

Proof. First, note that

$$\mathcal{P}_m((x*y)-w) = \mathcal{P}_m(x*y) - \mathcal{P}_m(w),$$

since $\mathcal{P}_m$ is a linear operator. We compute $\mathcal{P}_m(x), \mathcal{P}_m(y), \mathcal{P}_m(w)$ in time $O(m + (\|x\|_0 + \|y\|_0 + \|w\|_0)\log n)$ by computing ω^j for all $j \in \mathrm{supp}(x) \cup \mathrm{supp}(y) \cup \mathrm{supp}(w)$ using Taylor expansion of sine and cosine functions and keeping the first $\Theta(\log n)$ digits. We then compute, via the Fast Fourier Transform in time $O(m \log m)$, the vector $\mathcal{P}_m(x) * \mathcal{P}_m(y)$. We claim that

$$(\mathcal{P}_m(x*y))_i = (\mathcal{P}_m(x) * \mathcal{P}_m(y))_i .$$

Our claim is proved via the following chain of equalities:

$$(\mathcal{P}_m(x) * \mathcal{P}_m(y))_i = \tag{4.1}$$

$$\sum_{\ell,\ell' \in [m] : (\ell+\ell') \bmod m = i} (\mathcal{P}_m(x))_\ell \cdot (\mathcal{P}_m(y))_{\ell'} = \tag{4.2}$$

$$\sum_{\ell,\ell' \in [m] : (\ell+\ell') \bmod m = i} \left(\sum_{j \in [n] : h_m(j) = \ell} x_j \omega^j \right) \cdot \left(\sum_{j' \in [n] : h_m(j') = \ell'} y_{j'} \cdot \omega^{j'} \right) = \tag{4.3}$$

$$\sum_{\ell,\ell' \in [m] : (\ell+\ell') \bmod m = i} \left(\sum_{j,j' \in [n] : h_m(j) = \ell, h_m(j') = \ell'} x_j y_{j'} \omega^{j+j'} \right) = \tag{4.4}$$

$$\sum_{\ell,\ell' \in [m], j,j' \in [n] : h_m(j) = \ell, h_m(j') = \ell', (\ell+\ell') \bmod m = i} x_j y_{j'} \omega^{j+j'} = \tag{4.5}$$

$$\sum_{j,j' \in [n] : (h_m(j) + h_m(j')) \bmod m = i} x_j y_{j'} \omega^{j+j'} = \tag{4.6}$$

$$\sum_{j,j' \in [n] : h_m((j+j') \bmod m) = i} x_j y_{j'} \omega^{j+j'} = \tag{4.7}$$

$$\sum_{j'' \in [n], h_m(j'' \bmod m) = i} \sum_{j,j' \in [n] : j+j' = j''} x_j y_{j'} \omega^{j''} = \tag{4.8}$$

$$\sum_{j'' \in [n], h_m(j'' \bmod m) = i} \omega^{j''} \left(\sum_{j,j' \in [n] : j+j' = j''} x_j y_{j'} \right) = \tag{4.9}$$

$$\sum_{j'' \in [n], h_m(j'') = i} \omega^{j''} (x*y)_{j''} , \tag{4.10}$$

where (1) to (2) follows by defition of convolution, (2) to (3) by definition of the $\mathcal{P}_B$

operator, (3) to (4) by expanding the product in (2), (5) to (6) by the trivial fact each element in $[n]$ is mapped uniquely to some element in $[B]$ via h_m, (6) to (7) by the fact that $(h_m(a) + h_m(b)) \bmod m = (a \bmod m + b \bmod m) \bmod m = (a+b) \bmod m = (h(a+b) \bmod m))$, (7) to (8) by introducing the auxilliary variabe $j'' = j + j'$, (8) to (9) by the fact that $\omega^{j''}$ can be pulled outside of the inner sum since in that scope j'' is fixed, and (9) to (10) since $h_m(j'' \bmod m) = (j'' \bmod m) \bmod m = (j'' \bmod m) = h_m(j'')$ and the fact that the inner sum is the definition of convolution evaluated at point j''. □

In what follows C is some sufficiently large absolute constant.

Algorithm 15 $\textsc{Locate}(x, y, w, B, \delta)$

```
$L \leftarrow \emptyset$
**for** $t \in [5\lceil \log(1/\delta) \rceil]$ **do**
    Pick random prime $p$ in $[CB\log^2 n]$.
    Compute $\mathcal{P}_p((x * y) - w)$, using Lemma 4.1.3.
    **for** $b \in [p]$ **do**
        **if** $(\mathcal{P}_p((x * y) - w))_b \neq 0$ **then**
            $v \leftarrow |(\mathcal{P}_p((x * y) - w))_b|$.
            $ar \leftarrow (\mathcal{P}_p((x * y) - w))_b / |(\mathcal{P}_p((x * y) - w))_b|$.
            Compute $i$ from $ar$ using Lemma 4.1.5
            $L \leftarrow L \cup \{(i, v)\}$
        **end if**
    **end for**
**end for**
Prune $L$ to keep pairs $(i, v)$, which appear at least $(3/4) \cdot 5\log(1/\delta)$ times.
$z \leftarrow \vec{0} \in \mathbb{R}^n$
**for** $(i, v) \in L$ **do**
    $z_i \leftarrow v$
**end for**
Return $z$
```

The following Lemma is important, since we are dealing with numbers with finite precision.

Lemma 4.1.4. *Let $a, b \in [n]$ with $a \neq b$. If ω is rounded such that it fits in the word size, then $|\omega^a - \omega^b| > 1/n$.*

Proof. The quantity is minimized when $a = b + 1$. If n sufficiently large, it can then be approximated by an arc of length $2\pi/n$, and since the word size w is $\Omega(\log n)$, for sufficiently

Algorithm 16 HashAndIterate(x, y, B, δ)

```
$w^{(0)} \leftarrow 0$
for $r = 1$ to $\lceil \log B \rceil$ do
    $\delta_r \leftarrow \delta / \log B$
    $B_r \leftarrow B \cdot 2^{-r+1}$
    $z \leftarrow$ Locate$(x, y, w^{(r)}, B_r, \delta_r)$
    $w^{(r)} \leftarrow w^{(r)} + z$
end for
Return $w^{(\lceil \log B \rceil)}$
```

large constant we get the desired result. □

The follows Lemma is a crucial building block of our algorithm.

Lemma 4.1.5. *Given ω^j for $j \in [n]$, one can find j in time $O(\log n)$ (whether or not ω is rounded to fit the word size).*

Proof. From the pair (real part of ω^j,imaginary of ω^j) we can find in which of the four following sets j lies in

$$\{0, \ldots, \lceil n/4 \rceil\},$$

$$\{\lceil n/4 \rceil + 1, \ldots, \lceil n/2 \rceil\}$$

$$\{\lceil n/2 \rceil + 1, \ldots, \lceil 3n/2 \rceil\}$$

$$\{\lceil 3n/2 \rceil + 1, \ldots, n - 1\}$$

since each one corresponds to an arc of length $\pi/4$ of the complex circle. After detecting the set (equivalently the corresponding arc of the complex circle) one can perform a standard ternary search to find j. Due to Lemma 4.1.4 $O(\log n)$ iterations suffice to find j. □

The following Lemma is standard in the sparse recovery literature, but we give its proof for completeness. C is a large enough absolute constant.

Lemma 4.1.6. *Let an integer B such that $B > C \cdot \|(x * y) - w\|_0$, and let p be chosen at random from $[CB \log^2 n]$. Then, with probability $1 - p$, there exist at least $(1 - \gamma)\|(x * y) - w)\|_0$ indices*

$j \in \mathrm{supp}((x * y) - w)$ *such that*

$$\forall j' \in \mathrm{supp}((x * y) - w) \setminus \{j\} : h_p(j') \neq h_p(j),$$

where $2C^{-2}/p = \gamma$.

Proof. Let $j, j' \in \mathrm{supp}((x * y) - w)$, with $j \neq j'$. The hash function h_p is not pairwise independent, but the following property, which suffices for our purpose, holds

$$\mathbb{P}\left[h_p(j) = h_p(j')\right] \leq 1/B.$$

To see that, observe first that in order for $h_p(j) = h_p(j')$ to hold, it must be the case that p is a divisor of $j - j'$. Since $j - j' \leq n$ there there can be at most $\lceil \log n \rceil$ prime divisors of $j - j'$, otherwise $j - j'$ would be at least $2^{\lceil \log n \rceil + 1} > n$. By the prime number Theorem, there exist at least $(C/2)B \log n$ primes in $[CB \log^2 n]$, and hence a random prime will be one of the divisors of $j - j'$ with probability $2/(CB)$.

By the above discussion, the random variable X_j, defined to be the indicator variable of the event

$$\mathcal{E}_j : \exists j' \in \mathrm{supp}((x * y) - w) \setminus \{j\} : h_p(j) = h_p(j')$$

has expected value $\mathbb{E}\left[X_j\right] = \mathbb{P}\left[\mathcal{E}_j \text{ holds}\right] \leq (\|(x * y) - w)\|_0 - 1) \cdot (2/CB) \leq 2C^{-2}$, by a union-bound. Now, we have that

$$\mathbb{E}\left[\sum_{j \in \mathrm{supp}((x*y)-w)} X_j\right] \leq (2C^{-2})\|(x * y) - w\|_0.$$

By Markov's inequality, with probability $1 - p$ there exist at most $\gamma\|(x * y) - w\|_0$ indices $j \in \mathrm{supp}((x * y) - w)$ such that $X_j = 1$, if $2C^{-2}/p = \gamma$. This finishes the proof of the claim. □

The following argument is pretty standard in the sparse recovery literature, we give its proof for completeness.

Lemma 4.1.7. *Let the constants C, γ, p be as in Lemma 4.1.6 with $p \leq 2^{-12/5}$, and assume that $B > C\|(x * y) - w\|_0$. If $(x * y) - w$ is not the zero vector, then with probability $1 - \delta$ the subroutine*

LOCATE(x, y, w, B, δ) *returns a vector z such that*

$$\|z - ((x * y) - w)\|_0 \leq (5\gamma)\|(x * y) - w)\|_0.$$

Proof. Fix $t \in [5\log(1/\delta)]$, and assume that p, C, γ satisfy $pC^{-2}/p = \gamma$ We have that

$$\begin{aligned}(\mathcal{P}_p(x * y))_i - (\mathcal{P}_p(w))_i = (\mathcal{P}_p(x * y - w))_i = \\ \sum_{j \in [n]: h_p(j) = i} ((x * y) - w)_j \omega^j = \\ \sum_{j \in [n]: h_p(j) = i \text{ and } ((x*y)-w)_j \neq 0} ((x * y) - w)_j \omega^j\end{aligned}$$

The condition of Lemma 4.1.6 hold, so with probability $-p$ the its conclusion also holds. Condition on that event and consider the at least $(1 - \gamma)\|(x * y) - w\|_0$ indices in $\|(x * y) - w\|_0$, for which the conclusion of Lemma 4.1.6 holds. Fix such an index j^* and let $i^* = h_p(j^*)$. Due to the isolation property, we have that

$$(\mathcal{P}_p(x * y))_{i^*} - (\mathcal{P}_p(w))_{i^*} = ((x * y)_{j^*} - w_{j^*})\omega^{j^*}.$$

Now, due to Lemma 4.1.4 subroutine LOCATE(x, y, w, B, δ) will infer j^* correctly from $(\mathcal{P}_{\sigma,B}(x * y))_{i^*} - (\mathcal{P}_{\sigma,B} w)_{i^*}$, as well as $(x * y)_{j^*} - w_{j^*}$. We will say j^* is recognised in repetition t.

For the rest of the proof, unfix t. Since the conclusion of Lemma 4.1.6 holds with probability $1 - p$, the number of $t \in [5\log(1/\delta)]$ for which the conclusion of the Lemma holds is at least $4\log(1/\delta)$ with probability $1 - \delta$ since

$$\binom{5\log(1/\delta)}{(5/2)\log(1/\delta)} p^{(5/2)\log(1/\delta)} \leq 2^{5\log(1/\delta)} p^{(5/2)\log(1/\delta)} \leq \delta,$$

as long as $p \leq 2^{-12/5}$..

Let us call, for convenience, that above pairs good. Thus, with probability $1 - \delta$ the number of pairs (j, t) for which j is **not** recognised in repetition t is at most

$$\gamma \cdot 4\log(1/\delta) \cdot \|(x * y) - w\|_0 + \log(1/\delta)\|(x * y) - w\|_0.$$

Hence there exist at most $\beta = 4\gamma\|(x*y) - w\|_0$ indices which are recognized in less than $(3/4) \cdot 5\log(1/\delta)$ repetitions, otherwise the number of **not** good pairs (j,t) is at least

$$1 + \beta \cdot \frac{1}{4} \cdot 5\log(1/\delta)\|(x*y) - w\|_0 > \gamma \cdot 4\log(1/\delta) \cdot \|(x*y) - w\|_0 + \log(1/\delta)\|(x*y) - w\|_0$$

which does not hold for $\|(x*y) - w\|_0 > 0$. Moreover, there can be at most $\gamma\|(x*y) - w\|_0$ indices that do not belong in $\mathrm{supp}((x*y) - w)$, and which were mistakenly inserted into z. This gives in total the factor of 5γ. □

Lemma 4.1.8. *Let $\gamma < 1/10$, and let also B be an integer such that $B > C\|(x*y)\|_0$. Then the routine* HashAndIterate(x, y, B, δ) *returns an $\|x*y\|_0$-sparse vector r such that $r = x*y$, with probability $1-\delta$. Moreover, the running time is $O((B\log^2 n \log(B\log^2 n) + \|x\|_0 \log n + \|y\|_0 \log n) \cdot (\log\log B + \log(1/\delta)))$.*

Proof. It is an easy induction to show that at each step $\|(x*y) - w^{(r)}\|_0 \leq (5\gamma)^r \|x*y\|_0$, with probability $1 - \delta r/\log B$, so the total failure probability is δ. Conditioned on the previous events happending, we have $x*y - w^{(\lceil \log B \rceil)}$ is the all-zeroes vector since $\|x*y\|_0 \leq (4\gamma)^{\lceil \log B \rceil}\|(x*y) - w\|_0 < 1$. This gives that $w^{(\lceil \log B \rceil)} = x*y$.

The running time for Locate(x, y, w, B_r, δ_r), since $\|w\|_0 \leq 2B$ at all times is (ignoring constant factors for ease of exposition)

$$(B_r \log^2 n \log(B_r \log^2 n) + (\|x\|_0 + \|y\|_0 + B)\log n) \cdot \log(1/\delta_r),$$

' where the factor is obtained $B_r \log^2 n \log(B_r \log^2 n) + \|x\|_0 \log n + \|y\|_0 \log n + B\log n$ due to Lemma 4.1.3, and $B_r \log n$ due to Lemma 4.1.5.

So the total running time of HashAndIterate(x, y, B, δ) becomes, by summing over all $\lceil \log B \rceil$ rounds (ignoring constant factors for ease of exposition)

$$\Big(B\log^2 n \log(B\log^2 n) + \log n(\|x\|_0 + \|y\|_0)\Big)\log(\log B/\delta).$$

□

The following Lemma is a standard fact which follows by the fact that a degree n polynomial over $\mathbb{Z}_p$ has at most n roots. We give a sketch of the proof.

Lemma 4.1.9. *There exists a procedure* EqualityTesting(x, y, w)*, which runs in time* $O(\|x\|_0 + \|y\|_0 + \|w\|_0) \log n \log(1/\delta) + \widetilde{O}(\log^3 n \cdot \log(1/\delta))$*, and answers whether* $x * y = w$ *with probability* $1 - \delta$.

Proof. Let c' large enough. We pick a random prime in $[c' n \log n, 2c' n \log n]$, by picking a random number in that interval and running the Miler-Rabin primality test with target failure probability δ. We form polynomials f_x, f_y, f_w that have x, y, w as their coefficients respectively. We then pick $\Theta(\log(1/\delta)$ random elements in $\mathbb{Z}_p$ and check whether $(f_x(r) \cdot f_y(r)) \bmod p = f_w(r) \bmod p$ or not. We return Yes if this is the case for all chosen, and No otherwise.

□

We are now ready to prove our main theorem.

Proof. Let c be a sufficiently small constant and C a sufficiently large constant. For $r = 0, 1, 2, \ldots$, one by one we set $B_r \leftarrow C \cdot 2^r$ and $\delta_r = c \cdot r^{-2}$, run HashAndIterate$(x, y, B_r, \delta_r)$ to obtain z, and feed it to EqualityTesting$(x, y, z, \frac{1}{200 \log n})$. We stop when the latter procedure returns Yes. The total failure probability thus is at most

$$\log n \cdot \frac{1}{200 \log n} + \sum_{r \geq 1} \delta_r = \frac{1}{200} + \sum_{r \geq 1} c r^{-2} \leq \frac{99}{100}.$$

Conditioned on the aforementioned event happening, the total running time is (ignoring constants)

$$\|x * y\|_0 \log^2 n (\|x * y\|_0 \log^2 n) \log \log n + (\|x\|_0 + \|y\|_0) \log \|x * y\|_0 \log n \log \log n + \widetilde{O}(\log^4 n),$$

by a straighforward summation of the expression in Lemma 4.1.8 over all rounds $O(\log \|x * y\|_0) = O(\log n)$ rounds, as well as Lemma 4.1.9.

□

4.1.3 Discussion

The crux of our argument is that the operator $\mathcal{P}_m$ in defition 4.1.2 is convolution-preserving. Actually, any choice of the number ω, and not just some root of unity, would make the aboe operator be convolution-preserving, but our particular choice is crucial. First of all, the problem of finding j from ω^j should be easy, and moreover ω^j for any j should remain bounded, otherwise we would need to manipulate numbers with an enormous number of digits (even up to n), something that would be prohibitive for the runtime. Thus, choosing ω to be an nth root of unity seems a good choice. The iterative loop technique is standard in the sparse recovery literature, see for example [GLPS10, HIKP12a, GLPS17].

4.1.4 Multivariate Polynomials

We discuss how our algorithm extends to multivariate polynomials. It suffices to solve the d-dimensional version of the sparse convolution problem: given $x_{\vec{i}} = x_{i_1,i_2,\ldots,i_d}, y_{\vec{i}} = y_{i_1,i_2,\ldots,i_d}$ with $0 \leq i_j \leq n_j - 1$, their d-dimensional convolution is the vector $(x * y)$ the $\vec{i} = (i_1, \ldots, i_d)$ coordinate of which is

$$(x * y)_{\vec{i}} = \sum_{\vec{j},\vec{j}' \in [n_1]\times[n_2]\times\ldots\times[n_d] : \vec{j}+\vec{j}'=\vec{i}} x_{\vec{j}} \cdot y_{\vec{j}'},$$

where the addition of two vectors should be done component-wise over $\mathbb{Z}_{n_j}$ in the jth coordinate. The reasonable extension of the function h_m would be

$$h_m(i) = (i_1 + i_2 \cdot n_1 + i_3 \cdot n_1 n_2 + \ldots + i_d \cdot n_1 n_2 \ldots n_{d-1}) \bmod m.$$

Thus, reasonable extension of the operator $\mathcal{P}$ would be to define it as the $(n_1 n_2 \ldots n_d)$-dimensional vector the ith coordinate of which is

$$(\mathcal{P}_m)_i = \sum_{\vec{j} \in [n_1]\times[n_2]\times\ldots\times[n_d] : h_m(j)=i} x_{\vec{j}} \omega^{\frac{i}{n_1} + \frac{i_2}{n_1 n_2} + \ldots + \frac{i_d}{n_1 n_2 \cdots n_d}}.$$

where ω is now defined as $e^{2\pi\sqrt{-1}}$. It is not hard to see that the operator again is convolution-preserving and from $\omega^{\frac{i}{n_1} + \frac{i_2}{n_1 n_2} + \ldots + \frac{i_d}{n_1 n_2 \cdots n_d}}$ we can infer $i_1, \ldots i_n$ in $O(\log(n_1 n_2 \ldots n_d))$

time by first performing a ternary search in the complex circle i, and then learning first i_1, then i_2, so on so forth.

One would also need to use the Schwartz-Zippel lemma to perform the test we performed in Lemma 4.1.9.

4.2 Modular Subset Sum in Almost Linear Time. Deterministically

4.2.1 Preliminaries

For a positive integer m, we let $[m] = \{0, 1 \ldots, m-1\}$ and Z_m be the group of residues modulo m. For two sets $A, B \subseteq Z_m$, we define $A + B = \{x, \exists (a, b) \in A \times B : a + b = x\}$. Similarly, $A - B = \{x, \exists (a, b) \in A \times B : a - b = x\}$. We also let $A \bmod m = \{a \bmod m, a \in A\}$. We define

$$\mathcal{S}(A) = \{x \in \mathbb{Z}_m : \exists C \subseteq A, \sum_{c \in C} c = x\}$$

to be the set of all attainable subset sums of a set A. For two real-valued m-dimensional vectors v, u we define their cyclic convolution $v * y$ as the m-dimensional vector which for $i \in [m]$ satisfies

$$(v * u)_i = \sum_{j, j' \in \mathbb{Z}_m \times \mathbb{Z}_m : j + j' = i} u_j v_{j'} = \sum_{j, j' \in [m] \times [m] : (j + j') \bmod m = i} u_j v_{j'}$$

For a function $f : \mathbb{N} \to \mathbb{N}$ we define $\widetilde{O}(f) = O(f \log^c f)$ for some absolute constant f.

4.2.2 Results

Our algorithms solve the harder problem of producing all attainable subset sums. The randomized algorithm is output-sensitive, and the deterministic algorithm has a "small" additional multiplicative factor $m^{o(1)}$ in its runtime complexity, which would be reduced with any improvement in the deterministic version of sumset computation. The main result of this subchapter is the following.

Theorem 4.2.1. *(All attainable subset sums: deterministic version) Let $S \subseteq Z_m$ be a multiset with n elements. Then one can find $\mathcal{S}(S)$ in time $(|\mathcal{S}(S)| + n)m^{o(1)}$, deterministically. More precisely, the algorithm runs in time $\widetilde{O}((|\mathcal{S}(S)| + n)2^{O(\sqrt{\log|\mathcal{S}(S)| \log\log m})})$.*

The randomized version easily follows by our main approach, just by substituting algorithm 4.2.7 with 4.2.8.

Theorem 4.2.2. *(All attainable subset sums: randomized version) Let $S \subseteq Z_m$ be a multiset with n elements. Then one can find $\mathcal{S}(S)$ in expected $\widetilde{O}(|\mathcal{S}(S)| + n) \cdot \log^3 m$ time[1].*

For $n < m$ are algorithms become $O(m^{1+o(1)})$ and $\widetilde{O}(m)$ respectively, in terms of their dependence on m. We note that the randomized algorithm of Axiotis et.al. is also output-sensitive, but it is Monte-Carlo, in contrast to Theorem 4.2.2 which is Las Vegas.

Along the way, we also obtain a state of the art algorithm for deterministic sumset computation. Since sumset computation is a fundamental computational primitive, we believe that this result is of independent interest. Our algorithm makes efficient an algorithm of Chan and Lewenstein [CL15], which operated in a more restrictive scenario. The authors posed it as an open question, see Section 8, Remark 8.1 and 8.2 of [CL15].

Theorem 4.2.3. *(Deterministic Sumset Computation) Given two sets $A, B \subseteq \mathbb{Z}_m$, one can find $A + B$ in time $O(|A + B|m^{o(1)})$. In specific, the running time can be bounded by*

$$\widetilde{O}(|A + B| \cdot 2^{O(\sqrt{\log|A+B| \log\log m})}).$$

4.2.3 Technical Toolkit

4.2.4 Symmetry group and its properties

Definition 4.2.4. *(Symmetry group of a set)*

Let A be a subset of $\mathbb{Z}_m$. We define the symmetry group of A as $Sym(A) = \{h \in \mathbb{Z}_m : A + h = A\}$. The set $Sym(A)$ satisfies the group properties on Z_m with respect to addition.

[1] In fact, the $\log^3 m$ factor can be reduced to $\log^3 |\mathcal{S}(S)|$, see also the comment under Theorem 4.2.8

Since $Sym(A)$ is a group over $\mathbb{Z}_m$ with respect to addition, it holds that $Sym(A) = d \cdot \mathbb{Z}_{m/d}$, where d is the minimum non-zero element of $Sym(A)$.

It is easy to see that $Sym(A) \subseteq Sym(A+B)$, for any two sets $A, B \subseteq \mathbb{Z}_m$; this property will be of great importance to us. Moreover, for any $x \in A$, $Sym(A) \subseteq \{x\} - A$. This follows since any $h \in Sym(A)$ should map x to some $x' \in A$, which means $x = x' + h$, hence $h = x - x'$. Thus the symmetry group of a set A has size at most $|A|$. The following result is proven in the appendix.

Theorem 4.2.5. *(Computing the symmetry group) Given a set $A \subseteq \mathbb{Z}_m$, we can find $Sym(A)$ in time $O(|A|)$.*

4.2.5 Computing the Sumset of Two Sets

Definition 4.2.6 (The Sumset Problem)**.** *Given two sets $A, B \subseteq \mathbb{Z}_m$, compute $A + B$.*

The following theorem is proven in the appendix.

Theorem 4.2.7. *[Output-sensitive sumset: deterrministic version] For two sets $A, B \subseteq \mathbb{Z}_m$, the sumset problem can be solved in time*

$$O\left(|A+B| m^{o(1)}\right)$$

In the appendix we also give a self-contained proof of the following result, which can be obtained also from previous work in the literature, see [CH02]. However, we give a novel, and, we believe, simpler algorithm.

Theorem 4.2.8. *[Output-sensitive sumset: randomized version] For two sets $A, B \subseteq \mathbb{Z}_m$ the sumset problem can be solved in expected time*

$$O\left(|A+B| \log^3 m\right).$$

In fact, one should be able to turn the $\log m$ factors in the above theorem factors to $\log |A+B|$ by using Dietzfelbinger's hash function as in [BDP05], but for ease of exposition and since this is not our main result we do not elaborate on it.

It is obvious that the sumset problem is a special case of the sparse convolution problem, where one is asked to compute the convolution of two sparse vectors in output sensitive time.

4.2.6 Cauchy-Davenport and Kneser's Theorem

The following theorems lie at the core of our algorithm.

Theorem 4.2.9. *[Cauchy-Davenport Theorem](Theorem 5.4 in [TV06]) Let p be a prime, and let $A, B \subseteq \mathbb{Z}_m$. Then*

$$|A + B| \geq \min(|A| + |B| - 1, p).$$

The Cauchy-Davenport Theorem is an immediate corrolary of the renowned Uncertainly Principle in Fourier Analysis, see for example [Tao03].

The following is an easy corollary of the Cauchy-Davenport Theorem.

Corollary 4.2.10. *Let p be a prime, and let k sets $A_1, A_2, \dots, A_k \subseteq \mathbb{Z}_p$. If $\sum_{j=1}^{k} |A_j| \geq p + k - 1$, then $A_1 + A_2 + \ldots + A_k = \mathbb{Z}_p$.*

Proof. It suffices to prove that $|A_1 + A_2 + \ldots + A_k| \geq \min(\sum_{j=1}^{k} |A_j| - k + 1, p)$, from which the claim follows. We perform induction on k. For $k = 2$ the result is conclusion of the Cauchy-Davenport theorem. For $k > 3$, we have that if $A_1 + A_2 + \ldots + A_k$ is not $\mathbb{Z}_p$, then

$$\sum_{k=1}^{k} |A_j| = |A_k| + \sum_{j=1}^{k-1} |A_j| \tag{4.11}$$

$$< |A_k| + (|A_1 + A_2 + \ldots + A_{k-1}| + k - 2) \tag{4.12}$$

$$< (|A_1 + A_2 + \ldots A_k| + 1) + k - 2 = |A_1 + A_2 + \ldots + A_k| + k - 1, \tag{4.13}$$

where from (1) to (2) we used the inductive hypothesis, and from (2) to (3) the Cauchy-Davenport Theorem. We note that the inequalities used follow since $A_1 + A_2 + \ldots + A_k$ (and hence also $A_1 + A_2 + \ldots + A_{k-1}$) are not equal to $\mathbb{Z}_p$.

□

Algorithm 17

```
procedure ModularSubsetSumInPrimeUniverse($S, p$)    ▷ $S$ is an $n$-element set, $p$ is prime
    $X_{0,i} \leftarrow \{0, x_i\}$, for all $i \in [n]$
    $n_0 \leftarrow n$
    for $r = 1$ to $\log n$ do
        $n_r \leftarrow n_{r-1}/2$.
        for $i = 1$ to $n_{r-1}/2$ do
            $X_{r,i} \leftarrow X_{r-1,2i-1} + X_{r-1,2i}$    ▷ Sumset Computation via Theorem 4.2.7
            if $\sum_{j \leq i} |X_{r,j}| > p + i - 1$ then
                Return $\mathbb{Z}_p$
            end if
        end for
    end for
    Return $X_{\log n,1}$
end procedure
```

Theorem 4.2.11. *[Kneser's Theorem] (Theorem 5.5 in [TV06]) Let $A, B \subseteq \mathbb{Z}_m$. Then*

$$|A + B| \geq \min(|A| + |B| - |Sym(A + B)|, m)$$

4.2.7 Modular Subset Sum over $\mathbb{Z}_p$

In this section we give a warm-up, when $m = p$ is a prime number. We remind that we solve the stronger problem of finding all the attainable subset sums modulo m. We shall assume that n is power of 2, since we can just 0 any number of times we want, without affecting the attainable subset sums. Moreover, for ease of exposition our algorithm in this section is not output-sensitive. The algorithm appears in 17.

Lemma 4.2.12 (Bound on the Running Time and Proof of Correctness)**.** *The running time of Algorithm 17 is $O\left((m+n)m^{o(1)}\right) = O\left((p+n)p^{o(1)}\right)$. Moreover, the output is the set of all attainable subset sums.*

Proof. Let (r^*, i^*) be the values of r and i at the end of the execution of the algorithm. If $r^* = \log n$ we have $i^* = 1$. We have that for $r < r^*$

$$\sum_{i \leq n_{r-1}/2} |X_{r,i}| \leq p + \frac{n_{r-1}}{2} - 1,$$

as well as

$$\sum_{i<i^*} |X_{r^*,i}| \leq p + \frac{n_{r^*-1}}{2} - 1,$$

by the fact that (i^*, r^*) is the end of the execution of the algorithm. Because of the calls to Theorem 4.2.7 we have that the running time is bounded by (ignoring constant terms)

$$\sum_{r<r^*} \sum_{i=1}^{n_r/2} |X_{r,i}| m^{o(1)} + \sum_{i\leq i^*} |X_{r^*,i}| m^{o(1)}. \tag{4.14}$$

The term $\sum_{i\leq i^*} |X_{r^*,i}|$ can be split to two terms: the first $i-1$ summands, which can be bounded by $p + \frac{n_{r^*-1}}{2}$ as we have already seen, and the last term which can be trivially bounded by p. We now bound the first term in (4). We have that

$$\sum_{r<r^*} \sum_{i=1}^{n_r/2} |X_{r,i}| \leq$$
$$\sum_{r<r^*} \left(p - 1 + \frac{n_{r-1}}{2}\right) \leq$$
$$\log n \cdot p + n.$$

Plugging this into (4) we obtain the desired bound on the running time.

To prove correctness, observe that if the algorithm reaches Line 13, then it computed all attainable subset sums. Otherwise, if it returns in Line 9, then $\sum_{j\leq i} |X_{r,j}| > p + i - 1$, which by corollary 4.2.10 implies that $\sum_{j\leq i} X_{r,j} = \mathbb{Z}_p$, and hence $\mathcal{S}(S) = \mathbb{Z}_p$. Tis means that algorithm always produces the correct answer. $\square$

4.2.8 Modular subset sum over Z_m, for any m

In this section we proce Theorem 4.2.1. As in the previous section, we shall asssume that n is a power of 2, since we can add zeros to our set S. The algorithm for the Modular Subset Sum problems appears in Algorithm 18. Similarly to the prime version, the algorithm computes sumsets in a bottom-up fashion. If at some point $|X_{r,i}| < |X_{r-1,2i}| + |X_{r-1,2i+1}| - 1$, we know by Theorem 4.2.11 that $Sym(X_{r-1,2i} + X_{r-1,2i+1})$ is non-trivial. Since $Sym(X_{r-1,2i} +$

Algorithm 18

```
1: procedure ModularSubsetSum($S, m$)                ▷ $S$ is an $n$-element set
2:     for $B = 1, 2, \dots, 2^{\lceil \log m \rceil + 1}$ do
3:         $X_{0,i} \leftarrow \{0, x_i\}$, for all $i \in [n]$
4:         $n_0 \leftarrow n$
5:         for $r = 1$ to $\log n$ do
6:             $n_r \leftarrow \lceil n_{r-1}/2 \rceil$.
7:             for $i = 0$ to $\lceil n_{r-1}/2 \rceil$ do
8:                 $X_{r,i} \leftarrow X_{r-1,2i} + X_{r-1,2i+1}$        ▷ Sumset Computation via Theorem 4.2.7
9:                 if $|X_{r,i}| < |X_{r-1,2i}| + |X_{r-1,2i+1}| - 1$ then
10:                    Compute $Sym(X_{r,i})$                        ▷ Theorem 4.2.5
11:                    $d \leftarrow$ minimum non-zero element of $Sym(X_{r,i})$ ▷ $Sym(X_{r,i}) = d \cdot \mathbb{Z}_{m/d}$
12:                    Return ModularSubsetSum($S, d$) $+ d \cdot \{1, 2, \dots, m/d - 1\}$
13:                end if
14:                if $\sum_{j \le i} |X_{r,j}| > B + i - 1_{i \bmod 2 = 1}$ then
15:                    $n_r = i$
16:                    break
17:                end if
18:                if $n_{r-1}$ is odd then
19:                    $X_{r,n_r} = X_{r-1,n_{r-1}}$
20:                end if
21:            end for
22:        end for
23:        if $|X_{\log n,1}| < B$ then
24:            Return $X_{\log n,1}$
25:        end if
26:    end for
27: end procedure
```

$X_{r-1,2i+1}) \subseteq Sym(\mathcal{S}(S))$, we obtain that the $Sym(\mathcal{S}(S))$ is non-trivial, hence we can restrict ourselves solving the problem over $\mathbb{Z}_d$, where d is the minimum element in $Sym(X_{r-1,2i} + X_{r-1,2i+1})$.

One issue is how to obtain indices r, i such that $|X_{r,i}| < |X_{r-1,2i}| + |X_{r-1,2i+1}| - 1$ (if they exist). For that, the condition in lines 14-16 ensures that if $\sum_{j \le i} |X_{r,j}|$ is large enough (above some threshold), then this forces $|X_{r,i}| < |X_{r-1,2i}| + |X_{r-1,2i+1}| - 1$, for some r, i. The value B is used to guess the value of $\mathcal{S}(S)$, so that the thresholding is implemented.

We first prove that 18 always outputs the correct set of attainable subset sums.

Correctness of 18. We shall perform induction on the universe size m. For $m = 1$, which is the base case, the result is obvious. For larger m, we split to two cases.

Case 1: For some indices r, i (and B) $|X_{r,i}| < |X_{r-1,2i}| + |X_{r-1,2i+1}| - 1$.

This means that $Sym(X_{r,i})$ is non-trivial and hence $Sym(\mathcal{S}(S))$ is also non-trivial. Let d be the smallest non-zero element of $Sym(X_{r,i})$. Then we have that $d \in Sym(\mathcal{S}(S))$, and moreover:

1. For any $x \in \mathcal{S}(S)$ and any non-negative integer j, we obtain $x + j \cdot d \in \mathcal{S}(S)$, by observing that $\mathcal{S}(S) + j \cdot d = \mathcal{S}(S)$. By the minimality of d and since $Sym(X_{r,i})$ is a group over $\mathbb{Z}_m$, we get that d is a divisor of m. Thus, we obtain that there exists some non-negative integer j such that $x + jd \in \{0, \ldots, d-1\} \in \mathcal{S}(S)$.
2. For any $x \in \{0, \ldots, d-1\} \cap \mathcal{S}(S)$, we have that $x + j \cdot d \in \mathcal{S}(S)$, for any j.

The above two facts imply that $\mathcal{S}(S) = \textsc{ModularSubsetSum}(S, d) + d \cdot \{1, 2, \ldots, m/d - 1\}$. Because of the induction hypothesis on the correctness of ModularSubsetSum, we obtain the desired result.

Case 2: For fixed B there are no indices i, r such that $|X_{r,i}| < |X_{r-1,2i}| + |X_{r-1,2i+1}| - 1$, and the condition in line 14 never evaluates to true. Then the algorithm proceeds by finding the set of all modular subset sums in a straightforward manner.

Case 3: For fixed B there are no indices i, r such that $|X_{r,i}| < |X_{r-1,2i}| + |X_{r-1,2i+1}| - 1$, and the condition in line 14 does evaluates to true at least once. Let (r^*, i^*) be the last such pair found during the execution of the algorithm.

Since

$$\sum_{j \le i^*} |X_{r^*,j}| > B + i^* - 1_{i^* \bmod 2 = 1},$$

i f $r^* < \log n$ we obtain

$$\sum_{j \le \lceil i^*/2 \rceil} |X_{r^*+1,j}| \ge$$

$$\sum_{j \le \lfloor i^*/2 \rfloor} \left(|X_{r^*,2j}| + |X_{r^*,2j+1}| - 1\right) + 1_{i^* \bmod 2 = 1} \cdot |X_{r^*,i^*}| =$$

$$\sum_{j \le i^*} |X_{r^*,j}| - \lfloor i^*/2 \rfloor >$$

$$B + i^* - 1_{i^* \bmod 2=1} - \lfloor i^*/2 \rfloor = B + \lfloor i^*/2 \rfloor > B + \lfloor i^*/2 \rfloor - 1_{\lfloor i^*/2 \rfloor \bmod 2=1}.$$

The above implies that the condition will be evaluated true also for indices $(r^* + 1, \lceil i^*/2 \rceil)$, which contradicts the minimality of the pair (r^*, i^*), except if $r^* = \log n$. This means that $|X_{\log n,1}| \ge B$.

Combining the analyses from the above three cases we can argue the following way. First of all, note that if condition in line 9 ever evaluates to true, then correctness is proved by the inductive step. Let an invocation of the outer loop with $B > |\mathcal{S}(S)|$. Then the algorithm either finds a non-trivial symmetry group of $\mathcal{S}(S)$ (Case 1), either we are in Case 2 and is going to compute all attainable subset sums in a straightforward bottom-up fashion; we cannot be in Case 3 because this would mean that $|X_{\log n,1}| \ge B > |\mathcal{S}(S)|$, which is a contradiction since $X_{\log n,1} \subseteq \mathcal{S}(S)$. If $B \le |\mathcal{S}(S)|$ and no non-trivial symmetry group is found then two things can happen. The first is that we are in Case 2, and the algorithm is going to increment B because of the condition in line 23 and the fact that in that case $X_{\log n,1} = \mathcal{S}(S)$. The second is that we are in Case 3, from which we infer that $|X_{\log n,1}| \ge B$, and again due to liine 23 the algorithm will proceed by incrementing B.

□

We are now proceed with bounding the running time of the algorithm.

Proof. Let B^* be the smallest power of 2 that is at least $|\mathcal{S}(S)|$. Let $T(S, m)$ be the running time of our algorithm. Then, by the analysis of the correctness of the algorithm a similar analysis as in the previous section one can bound the running time by

$$(T(S,d) + |\mathcal{S}(S)|) + \sum_{B \le B^*} (B+n)m^{o(1)},$$

where the first term accounts for the recursion in line 12 and building the subset sums in the same line, while in the second term the variable B is over all powers of 2. Since $d \le m/2$, the recursion easily yields the desired result.

□

4.2.9 Output-sensitive Sumset Computation

In this section we give two algorithms for sumset computation, one deterministic and one randomized, proving Theorems 4.2.7 and 4.2.8 respectively.

Deterministic algorithm

For a set S and a number m we define $S \bmod m = \{x \bmod m, x \in S\}$. We will need the following Lemma, which appears in [CL15]. For completeness, we also give a self-contained proof.

Lemma 4.2.13. *Let two sets $A, B \subseteq \{0, \dots m-1\}$, and let T be a set containing $A + B$. There exists an algorithm* PromiseSumset*(A, B, T, m) which in time $O(|T|m^{o(1)})$ computes $A+B$ over Z_m. More specifically, the running time is $O(|T|2^{\sqrt{\log|T| \log\log m}})$.*

For numbers $p_1, p_2, \dots, p_\ell$, we will call $(p_1, p_2, \dots, p_\ell)$ a tuple of length ℓ, or just a tuple.

Definition 4.2.14. *For prime numbers $p_1, \dots, p_i$ we define the function*

$$h_{p_1,\dots,p_i}(x) = x \mod \prod_{j=1}^{i} p_j.$$

Moreover, for a set T we define

$$h_{p_1,\dots,p_i}(T) = \{y : \exists x \in T, h_{p_1,\dots,p_i}(x) = y\}$$

Lastly, let $h^{-1}_{p_1,\dots,p_i}(y, T)$ be the values $x \in T$ such that $h_{p_1,\dots,p_i}(x) = y$.

Clearly, the function h satisfies the property $(h(x) + h(y)) \bmod \Pi_{i=1}^{j} p_j = h(x+y)$, for any two natural numbers x, y.

Definition 4.2.15 (Perfect Family). *Let $\mathcal{H}$ be a collection of tuples of primes. Then $\mathcal{H}$ is called perfect for a set T if for every $x \in T$ there exists a tuple of prime numbers $(p_1, \ldots, p_i) \in \mathcal{H}$ such that*

$$h^{-1}_{p_1,\ldots,p_i}(h_{p_1,\ldots,p_i}(x), T) = \{x\}.$$

In other words, every $x \in T$ is isolated from $T \setminus \{x\}$ under $h_{p_1,\ldots p_i}$.

Lemma 4.2.16. *(Construction of perfect family [CL15]) For a set $T \subseteq \mathbb{Z}_m$, a perfect family $\mathcal{H}$ for T can be found in time*

$$O(\ell 2^\ell \cdot |T|^{1+1/\ell} \cdot \mathrm{poly}(\log m)).$$

The size of $\mathcal{H}$ is at most

$$\ell \cdot 2^\ell \log |T|.$$

Moreover, the length of every tuple in $\mathcal{H}$ is ℓ, and the largest prime in every tuple is $O(|T|^{1/\ell} \log^2 m)$.

Proof. If such a prime p_i in Line 7 always exists, then the algorithm at the end of the inner loop it detects at least $|T|/2^\ell$ numbers $x \in S$ for which are mapped to a bucket with strictly less than $|T|^{1-\ell/\ell} + 1 = 2$ elements in T. In other words, each one of these elements x satisfy $h_{p_1,\ldots,p_\ell}(T) = \{x\})$, and hence will be removed from S in Line 10. Thus, if r is the number of iterations till S is exhausted, we have that

$$|T| \left(1 - 2^{-\ell}\right)^r < 1,$$

which happens for $r = \left\lceil \frac{\log |T|}{\log(\frac{2^\ell}{2^\ell - 1})} \right\rceil$.

We now proceed by proving that in Line 7 such a prime always exists. We fix S and perform induction on i. For $i = 0$ we trivially have that $h^{-1}_{p_1,\ldots,p_i}\left(h_{p_1,\ldots,p_i}(x), T\right)$ is a set of $|T|$ elements for every $x \in S$, so condition in 7 trivially holds. Define

$$C_i(x) = h^{-1}_{p_1,\ldots,p_i}\left(h_{p_1,\ldots,p_i}(x), T\right) \setminus \{x\},$$

to be elements of T that collide with x under the first i primes. For $i \geq 1$ we already have by the inductive hypothesis that

$$|C_{i-1}(x)| < |T|^{1-(i-1)/\ell} + 1$$

for at least $|S| \cdot 2^{-i+1}$ elements $x \in S$. Call these elements good. We will now argue the existence of p_i with the desired properties via the probabilistic method. We pick p_i as a random prime from $[C \cdot |T|^{1/\ell} \cdot \log^2 m]$. Observe that that for a good x and $y \in T \setminus \{x\}$ we have that the probability $\mathbb{P}_{p_i}[p_i|(x-y)]$ is the number of prime divisors of $x - y$, divided by the number of primes in $[C|T|^{1/\ell} \log^2 m]$. A crude upper bound on the prime divisors of $x - y$ is $\log m$, otherwise $x - y > 2^{\log m} = m$. The number of primes in $[C|T|^{1/\ell} \log^2 m]$ is $c \log |T|^{1/\ell} \log m$ by the Prime Number Theorem, where c is an absolute constant larger than 6, as long as C is chosen to be at least 7. Thus, the probability that $p_i|x-y$ is at most $c^{-1}|T|^{-1/\ell}$. Thus, by the inductive hypothesis on the number of good elements of S we have that

$$\mathop{\mathbf{E}}_{p_i}[C_i(x)] < |T|^{1-(i-1)/\ell} \cdot c^{-1}|T|^{-1/\ell} = c^{-1}|T|^{1-i/\ell},$$

as we need to take a union-bound only over the elements where x collides with under $p_1, \ldots, p_{i-1}$; these elements are at most $|T|^{1-(i-1)/\ell}$ by the induction hypothesis.

Invoking Markov's inequality we get that

$$\Pr_{p_i}\left[|C_i(x)| > |T|^{1-i/\ell}/2\right] < 2/c.$$

Thus, the expected number of good x such that $|C_i(x)| < |T|^{1-i/\ell}/2$ is at least (number of good elements)$\cdot(1 - 2/c) \geq |S|2^{-i}$, and hence there exists a choice of p_i such that the condition of Line 7 holds.

Lastly, we bound the running time. The outer loop will be performed $\log |T| / \log(2^\ell/(2^\ell - 1)) = O(2^\ell \log |T|)$ times. The inner loop will be performed ℓ times. The computation in Line 7 can be easily implemented in $O(|T|^{1+1/\ell} \cdot \mathrm{poly}(\log u))$ time. The total running time

Algorithm 19

```
1: procedure PerfectFamilyConstruction($T, m, \ell$)
2:     $S \leftarrow T$
3:     $\mathcal{H} \leftarrow \emptyset$
4:     while $S \neq \emptyset$ do
5:         for $i = 1$ to $\ell$ do
6:             Pick prime $p_i \in [C \cdot |T| \cdot \log^2 m]$ such that
7:                 $|\{x \in S : |h^{-1}_{p_1,\ldots,p_i}\left(h_{p_1,\ldots,p_i}(x), T\right)| < |T|^{1-i/\ell} + 1\} \geq |S| \cdot 2^{-i}$
                   ▷ all $x$ such that the bucket of $x$ under $h_{p_1,\ldots,p_i}$ is not "too heavy"
8:         end for
9:         $\mathcal{H} \leftarrow \mathcal{H} \cup (p_1, p_2, \ldots, p_\ell)$
10:        Remove $x \in S$ such that $h_{p_1,\ldots,p_i}(T) = \{x\}$ ▷ $x$ is isolated from $T$ under $p_1, \ldots p_i$
11:    end while
12:    Return $\mathcal{H}$
13: end procedure
```

then follows. □

The proof of Lemma 4.2.13 follows.

Proof. We first find a perfect family $\mathcal{H}$ for T by calling PerfectFamilyConstruction(T, m, ℓ) (Algorithm 19) with $\ell = \Theta(\sqrt{\log n / \log\log m})$. Then, every tuple $(p_1, \ldots, p_\ell) \in \mathcal{H}$ is mapped to an integer by the formula $\prod_{j=1}^{\ell} p_j$. Using Lemma 4.2.16 the total running time for this step is

$$\ell \cdot |\mathcal{H}| = \ell^2 2^\ell \cdot \log |T|.$$

Next for every $p' = (p_1, \ldots, p_\ell) \in \mathcal{H}$ we compute $h_{p'}(A)$ and $h_{p'}(B)$. By a Fast Fourier Transform in a universe of size $(|T|^{1/\ell} \log^2 m)^\ell$ we compute $h_{p'}(A+B) = \left(h_{p'}(A) + h_{p'}(B)\right) \bmod \prod_{j=1}^{\ell} p_j$. We then iterate over all $x \in T$ and check whether $h_{p'}(x) \in h_{p'}(A+B)$. Every $x \in T$ which yields a negative answer for at least one $p' \in \mathcal{H}$, is discarded. It is clear that every $x \in A+B$ is not discarded. Since for every $x \in T$ there exists a tuple $p' \in \mathcal{H}$ such that $h^{-1}_{p'}(h'_p(x)) = \{x\}$, every $x \in T \setminus (A+B)$ will be discarded when considering the corresponding p'.

Summing up the time to construct the perfect family, the time to map tuples to integers,

Algorithm 20

```
1: procedure DETERMINISTIC SUMSET($A, B, m$)
2:     $T_0 \leftarrow \{0, 1\}$
3:     $m_0 \leftarrow 2$
4:     for $r = 1$ to $\lceil \log m \rceil$ do
5:         $m_r \leftarrow 2 \cdot m_{r-1}$
6:         $Q_r \leftarrow \emptyset$
7:         for $x \in Q_{r-1}$ do $Q_r \leftarrow T_r \cup \{x, x + m_{r-1}\}$
8:         end for
9:         $A_r \leftarrow \emptyset$
10:        for $a \in A$ do $A_r \leftarrow A_r \cup \{a \bmod m_r\}$
11:        end for
12:        $B_r \leftarrow \emptyset$
13:        for $b \in B$ do $B_r \leftarrow B_r \cup \{b \bmod m_r\}$
14:        end for
15:        $T_r \leftarrow$ PROMISESUMSET($A_r, B_r, Q_r, m_r$)
16:    end for
17:    $S \leftarrow \emptyset$
18:    for $x \in T_{2^{\lceil \log m \rceil}}$ do $S \leftarrow S \cup \{x \bmod m\}$
19:    end for
20:    Return $S$
21: end procedure
```

and the time to perform $2|\mathcal{H}|$ Fast Fourier Transforms, the total running time then becomes (ignoring logarithmic factors in T, m)

$$\ell 2^{\ell} \cdot |T|^{1+1/\ell} + \ell^2 2^{\ell} + |T| \log^{2l} m \cdot \ell^2 2^{\ell}.$$

Plugging in the value $\ell = \beta \left\lceil \sqrt{\log |T| / \log \log m} \right\rceil$, for some constant β, yields the desired result.

□

We proceed with the proof of Theorem 4.2.7.

Proof. The algorithm appears in 20. We prove inductively that $T_r = (A + B) \bmod m_r$ in line 11. The base of the induction clearly holds for $r = 0$. For $r \geq 1$, we first observe that $x = x'$ over $\mathbb{Z}_{m_{r-1}}$ implies that $x - x' = km_{r-1}$ for some integer k, which in turns implies that $x - x'$ over Z_{m_r} is either 0 (if k is even) either odd (if k is odd). By the induction hypothesis, we have that $(a+b) \bmod m_{r-1} \in T_{r-1}$, so $(a+b) \bmod m_r \in Q_r$ and $(a+b+m_{r-1}) \bmod m_r \in Q_r$. Thus $|Q_r| \leq 2|T_{r-1}| \leq 2|A+B|$, and due the call to PromiseSumset we will get the desired result in time $O(|T_{r-1}|m_r^{o(1)})$. At the last step in Line 18 we map all elements of $(A+B) \bmod 2^{\lceil \log m \rceil}$ to $(A+B) \bmod m$. Since $|(A+B) \bmod m| \geq \frac{1}{2} \cdot |(A+B) \bmod 2^{\lceil \log m \rceil}|$, we obtain the desired bound on the running time.

□

Randomized algorithm

We also give a self-contained proof of Theorem 4.2.8. Our algorithm will solve the more general problem of computing the cyclic convolution of two m-dimensional sparse vectors $u, v \in$ with positive coordinates in output-sensitive time. We will need the following simple lemma, which has also been critically exploited in previous work.

Lemma 4.2.17. *Let u, v be m-dimensional vectors with non-negative coordinates. Let w be m dimensional vector such that $a_i \geq (u * v)_i$ for all $i \in [m]$. Then,*

$$\left(\sum_{i\in[m]} u_i\right)\left(\sum_{i\in[m]} v_i\right) = \sum_{i\in[m]} w_i,$$

if and only if $w = u * v$.

Proof. Let $w_i - (v * u)_i = \epsilon_i \geq 0$. Observe that

$$\sum_{i\in[m]} w_i = \sum_{i\in[m]} \left((u*v)_i + \epsilon_i\right) =$$

$$\sum_{i\in[m]} \sum_{j\in[m]} u_j v_{i-j} + \sum_{i\in[m]} \epsilon_i =$$

$$\left(\sum_{i\in[m]} u_i\right)\left(\sum_{i\in[m]} v_i\right) + \sum_{i\in[m]} \epsilon_i,$$

The latter equality implies that all ϵ_i should be zero, since they are all non-negative, and vice versa. □

Equipped with the above Lemma, we proceed with the proof of Theorem 4.2.8. Since a lot of ideas here are common with previous work, we will give a more high-level proof.

Proof. As in the deterministic case, we shall solve the following promise problem: we are given a subset T of size $2|A+B|$ which contains $A+B$. Then, invoking Algorithm 20 we can solve the more general problem.

We randomly pick numbers $R = c \log m (c \geq 3)$ prime numbers $q_1, q_2, \ldots q_R \in [C|T|\log^2 m]$ for a sufficiently large constant C; to do that, we afford to run Eratosthene's sieve in time $O(|T|\log^2 m \cdot \log\log m)$, and then pick R of them at random. By the prime number theorem, there are $\Omega(|T|\log m)$ primes in $[C|T|\log^2 m]$. Moreover, for any distinct indices $i, i' \in T$ there are at most $\log m + 1$ distinct prime divisors of $i - i'$, otherwise $i - i'$ would be at least $2^{\log m + 1} > m$. This means that for the hash function $h_p(x) = x \bmod p$ we have that for any $x, y \in T$ we have that the probability that $h_p(x) = h_p(y)$ is $\frac{2\log m+1}{\Omega(|T|\log m)} = O(\frac{1}{|T|})$, and hence the probability that $h_p(x) = h_p(z)$ for some $z \in T \setminus \{x\}$ is les than a small constant, by a union-bound. Due to the R different primes, the probability that for at least half of them x collides with $T \setminus \{x\}$ under the corresponding hash function, can be made $1/\text{poly}(m)$ by tuning the paramteter c in the definition of R. Thus, if for every p among the R chosen, we define vectors $u^{(p)}, v^{(p)} \in \mathbb{Z}^p$ such that $u_i^{(p)} = |\{x \in A : h_p(x) = i\}|, v_j^{(p)} = |\{y \in B : h_p(y) = j\}$, and compute $u^{(p)} * v^{(p)} \in \mathbb{Z}^p$, we have the following fact. For all $x \in T$, for at least $R/2$ of the R primes it holds that $(u^{(p)} * v^{(p)})_x = \sum_{j \in [m]} 1_A(j) 1_B(x-j)$. This holds since $(h_p(x) + h_p(y)) \bmod p = h_p(x+y)$, see also [CL15]. Thus, by a standard median argument, we will find the values $\sum_{j\in[m]} 1_{h_p(A)}(j) 1_{h_p(B)}(x-j)$ for all $x \in T$, i.e. the number of ways that x can be written as a sum of a number from A and a number from B. Since the values we obtained can only be over-estimators, using Lemma 4.2.17 we can easily turn this algorithm to a Las Vegas one by repeating and checking.

□

4.2.10 Computing the symmetry group

In this section we prove Theorem 4.2.5. We look at the following two strings. The first one is z with length $2n$, having characters

$$\begin{aligned} z_1 &= a_2 - a_1, \\ z_2 &= a_3 - a_2, \\ &\dots, \\ z_{n-1} &= a_n - a_{n-1}, \\ z_n &= a_1 - a_n + m, \\ z_{n+1} &= a_2 - a_1, \\ &\dots, \\ z_{2n} &= a_n - a_{n-1}, \end{aligned}$$

and the second string is w of length n, having characters

$$\begin{aligned} w_1 &= a_2 - a_1, \\ w_2 &= a_3 - a_2, \\ &\dots, \\ w_{n-1} &= a_n - a_{n-1}, \\ w_n &= a_1 - a_n + m. \end{aligned}$$

We perform pattern matching with w as the pattern and z as the text. This can be acomplished in time $O(|A|)$ by running the Knuth-Morris-Pratt algorithm. We claim the following, which proves correctness of our algorithm, and finishes the proof of 4.2.5.

Lemma 4.2.18. *If there exists a match between z and w at position i, then $a_i - a_1 \in Sym(A)$. Moreover, if $x \in Sym(A)$, then $x = a_i - a_1$ for some i and there exists a match between z and w at* *position i.*

Proof. Assume that there exists a match at position i. Then we have that for all $1 \leq j \leq |A|$

$$a_{j+1} - a_j = a_{i+j} - a_{i+j-1} \tag{4.15}$$

where the indices are taken modulo $|A|$. This means that $a_i - a_1 \in Sym(A)$, since

$$a_i = a_1 + (a_i - a_1), \tag{4.16}$$

$$a_{i+1} = (a_2 - a_1) + a_i = a_2 + (a_i - a_1), \tag{4.17}$$

$$a_{i+2} = (a_3 - a_2) + a_{i+1} = a_3 + (a_{i+1} - a_2) = a_3 + (a_i - a_1), \tag{4.18}$$

$$\dots \tag{4.19}$$

$$a_{i+j} = a_{i+j-1} + a_{j+1} - a_j = a_{j+1} + (a_{i+j-1} - a_j) = a_{j+1} + (a_i - a_1), \tag{4.20}$$

where all indices are taken modulo $|A|$, and in every line the first equality follows from (6), and the last equality by the previous equation, except from the first two lines in which it follows trivially. The above equalities imply that $a_i - a_1 \in Sym(A)$. This proves the first part of the lemma.

For the second part, observe, as we also discussed in subsection 4.2.4, that $Sym(A) \subseteq A - \{a_1\}$. Thus, if $a_i - a_1 \in Sym(A)$, all equations (7)-(11) hold, and thus one can deduce equation (1) by substracting every two consecutive equations (cyclically), which in turn implies that there exists a match between w and z in position i.

□

4.3 Subset Sum in Almost Output Sensitive Time

4.3.1 Preliminaries

For a set S of n positive intgers and an integer t we denote by $\mathcal{S}(S,t)$ to be the set of all attainable subset sums of S, which are also smaller than t. Formally, we have

$$\mathcal{S}(S,t) = \{u \le t : \exists T \subseteq S, \sum_{x \in T} x = u\}.$$

We will also denote by $\sigma_S = \sum_{x \in S} x$, and $\max(S) = \max_{x \in S} x, \min(S) = \min_{x \in S} x$. For a set $B \subseteq \mathcal{S}(S,t)$ we define the function I_B which maps every $x \in B$ to the lexicographically smallest $T \subseteq S$ such that $\sigma_T = x$.

We also let $[n] = \{1, 2, \dots n\}$. For a function $f : \mathbb{N} \to \mathbb{N}$ we define $\widetilde{O}(f) = O(f \log^c f)$ for some sufficiently large constant c. Moreover, for a hash function h from a universe U to a universe V, we define $h(T) = \{v \in V : \exists t \in T, h(t) = v\}$. We will say that a set $T \subseteq U$ is perfectly hashed under h if $|h(T)| = |T|$. Any $h^{-1}(v) = \{u \in U : h(u) = v\}$ will be referred to as a "bucket" of h.

4.3.2 Our Contribution

We remind the classical SubsetSum problem: given a set S of positive integers, and a target t, find whether there exists a subset of S that sums up to t. The classical textbook algorithm of Bellman via dynamic programming runs in time $O(nt)$, while the algorithm of Bringmann [Bri17] solves the problem in time $\widetilde{O}(t+n)$. However, a closer look at both algorithms reveals an interesting property: the algorithm of Bellman can be implemented to run in time $O(n|\mathcal{S}(S,t)|)$, while the algorithm of [Bri17] runs in time $\widetilde{\Omega}(n+t)$. This means that it can still be the case that the folklore dynamic programming algorithm is much better than [Bri17] in many cases of interest, in particular when $|\mathcal{S}(S,t)| = o(t/n)$. Thus, it might not be clear which of the two algorithms is better, and the following natural question arises.

Question 4.3.1. *Does there exist an algorithm which runs in time $O(|\mathcal{S}(S,t)| \cdot \mathrm{poly}(\log(nt))|)$?*

Such an algorithm would be strictly better than the textbook solution, and would be at least as good as Bringmann's algorithm. Our work makes considerable progress towards this question. Namely, we prove the following theorem.

Theorem 4.3.2. *There exists a randomized algorithm which, given a set S and a target t, computes*

$\mathcal{S}(S,t)$ in time (ignoring logarithmic factors in t and n)

$$\alpha^{-2} \left|\mathcal{S}\left(S, t + t/\alpha\right)\right|),$$

with probability 99/100. Setting $\alpha = \operatorname{poly}(\log t)$ in specific, one obtains running time (ignoring logarithmic factors in n and t)

$$\left|\mathcal{S}\left(S, t + \frac{t}{\operatorname{poly}(\log(nt))}\right)\right|.$$

Remark 4.3.3. *In the above theorem it is possible to obtain (ignoring logarithmic factors in n and t) $\alpha^{-1} \cdot |\mathcal{S}(S, t + t/a)|$, but we do not elaborate on it.*

4.3.3 Output-sensitive sumset computation in the non-modular case

This algorithm follows immediatelly by the modular version of sumset computation.

Lemma 4.3.4. *Let $A, B \subseteq [u]$ be two sets of integers, both containing zero. The one can find $A + B$ in expected time $\widetilde{O}(|A+B|) \cdot \log^3 u$.*

Lemma 4.3.5 (Capped Version of Output-Sensitive Sumset Computation)**.** *Let A, B be two sets containing integers less than u, both containing zero. Then, for any positive integer α, one can find $(A+B) \cap [u]$ in expected*

$$\widetilde{O}\left(\alpha^{-2} \left|(A+B) \cap [(1+\alpha)u]\right|\right) \cdot \log^3 u$$

time.

Proof. We partition $[u]$ to α intervals of length $\lfloor u/\alpha \rfloor$, except possibly the last one which has length $u - (\alpha - 1)\lfloor u/\alpha \rfloor$. Let these intervals be $J_1, \ldots, J_\alpha$, and denote $A_i = A \cap J_i$, $B_i = B \cap J_i$. For every $0 \leq i, j \leq \alpha$ we check whether $\min_{x \in A_i} x + \min_{y \in B_j} y \leq u$, and if this is the case we compute $A_i + B_j$ using Lemma 4.3.4. The union of all $A_i + B_j$ clearly is a superset of $(A+B) \cap [u]$, and thus we can project their union down to $[u]$ to obtain $(A+B) \cap [u]$. Snce every $A_i + B_j \subseteq (A+B) \cap [(1+\alpha)u]$ and $|(A+B) \cap [u]| \geq \max(|A|, |B|)$ by the fact that both sets contain zero, we obtain the desired bound on the running time.

□

Remark 4.3.6. *Using machinery developed in the context of the sparse Fourier transform, one can give a Monte Carlo algorithm with running time proportional to (ignoring polylogarithmic factors)*

$$\alpha^{-1} \left|(A+B) \cap [(1+\alpha)u]\right|.$$

4.3.4 A simple algorithm for all-attainable Subset Sums (case $t = \sigma_S$)

The following theorem is the result of this section.

Theorem 4.3.7. *Let S be a multiset of n positive integers. Then $\mathcal{S}(S, \sigma_S)$ can be found in expected*

$$\widetilde{O}(|\mathcal{S}(S, \sigma_S)|) \cdot \log^3 \sigma_S$$

time.

Our algorithm is based on a simple divide and conquer-approach. We indicate that a careful way to divide the multiset S to three parts reduces the sum of every part to half of that of the initial set. On the conquer step we apply our output-sensitive algorithm. The above two properties allow us to carefully bound the running time by $\widetilde{O}(|\mathcal{S}(S, \sigma_t)|)$. For the following two claims we let $S = \{s_1, \dots, s_n\}$ be in sorted order, S_1 be the odd-indexed elements of $S \setminus \{s_n\}$, and S_2 be the even-indexed element of $S \setminus \{s_n\}$.

Claim 4.3.8. *If n is odd then*

$$\sigma_{S_1} \le \sigma_{S_2} \le \frac{\sigma_S}{2},$$

else

$$\sigma_{S_2} \le \sigma_{S_1} \le \frac{\sigma_S}{2}.$$

Proof. If n is odd then

$$\sum_{1 \le i \le (n-1)/2} s_{2i-1} \le \sum_{1 \le i \le (n-1)/2} s_{2i} \le \sum_{1 \le i \le (n-1)/2} s_{2i+1},$$

by the monotonicity of the s_i. The above inequality is translated to $\sigma_{S_1} \leq \sigma_{S_2} \leq \sigma_S - \sigma_{S_2}$, which gives the deisred result. A totally analogous argument holds for the case that n is even.

□

Lemma 4.3.9 (Number of subset sums decrease by 1/2). *It holds that*

$$|\mathcal{S}(S_1, \sigma_{S_1})|, \mathcal{S}(S_2, \sigma_{S_2})| \leq \frac{1}{2}(|\mathcal{S}(S, \sigma_S)| + 1).$$

Proof. We prove the case that n is odd, since the case that n is even is completely analogous. For any $x \in \mathcal{S}(S_1, \sigma_{S_1}) \setminus \{0\}$ take $I \subseteq S_1$ such that $\sigma_I = x$. Then map I to $I \cup S_2$. We thus have $z = \sigma_{I \cup S_2} = x + \sigma_{S_2} > \sigma_{S_1}$ by Claim 4.3.8, and hence $z \notin \mathcal{S}(S_1, \sigma_{S_1})$. The aforementioned mapping never maps two different x to the same number z, so we obtain the inequality

$$|\mathcal{S}(S_1, \sigma_{S_1})| - 1 \leq |\mathcal{S}(S, \sigma_S) \setminus \mathcal{S}(S_1, \sigma_{S_1})|,$$

from which we conclude, since $\mathcal{S}(S_1, \sigma_{S_1}) \subseteq \mathcal{S}(S, \sigma_S)$, that $|\mathcal{S}(S_1, \sigma_{S_1})| \leq \frac{1}{2}(|\mathcal{S}(S, \sigma_S)| + 1)$.

Now, for S_2 we do the following. As before, $x \in \mathcal{S}(S_2, \sigma_{S_2}) \setminus \{0\}$ let $I \subseteq S_2$ such that $\sigma_I = x$. Now map I to $I \cup S_1 \cup \{s_n\}$. We have that $z = \sigma_{I \cup S_1 \cup \{s_n\}} = \sigma_I + \sigma_{S_1 \cup \{s_n\}} = \sigma_I + (\sigma_S - \sigma_{S_2}) \geq \sigma_I + \sigma_{S_2}$, by Claim 4.3.8. Now, from the above it follows that $z > \sigma_{S_2}$, and hence $z \notin \mathcal{S}(S_2, \sigma_{S_2})$. This implies that

$$|\mathcal{S}(S_2, \sigma_{S_2})| - 1 \leq |\mathcal{S}(S, \sigma_S) \setminus \mathcal{S}(S_1, \sigma_{S_2})|,$$

from which the claim follows.

□

Proof. We split S to $S_1 \cup S_2 \cup \{s_n\}$, as in Lemmas 4.3.8 and 4.3.9. We recursively compute $\mathcal{S}(S_1, \sigma_{S_1})$ and $\mathcal{S}(S_2, \sigma_{S_2})$ and then, using two calls to the output-sensitive sumset computation, we compute

Algorithm 21

```
procedure SubsetSumforLargeElements($S, t, t', \delta$)        ▷ $t \le t', \forall x \in S, x \ge t/\log^c n$
    $R \leftarrow \Theta(\log(t/\delta))$
    $O \leftarrow \emptyset$
    for $r \in [R]$ do
        Pick 2-wise independent hash function $h_r : S \leftarrow [2\log^{2c} t']$
        $S_r \leftarrow \emptyset$
        for $j = 1$ to $2\log^{2c} t'$ do
            $Z_{r,j} = h_r^{-1}(j) \cup \{0\}$        ▷ Elements that are hashed to bucket $j$ under $h_r$
            $S_r \leftarrow (S_r + Z_{r,j}) \cap [t]$        ▷ Capped Sumset Computation via Lemma 4.3.5
        end for
        $O \leftarrow O \cup S_r$
    end for
    Return $O$
end procedure
```

$$\mathcal{S}(S_1, \sigma_{S_1}) + \mathcal{S}(S_2, \sigma_{S_2}) + \{s_n\}.$$

The expected running time of the conquer step is $\widetilde{O}(|\mathcal{S}(S, \sigma_s)|)$, since every element that lies in the sumset of $\mathcal{S}(S_1, \sigma_{S_1}) + \mathcal{S}(S_2, \sigma_{S_2}) + \{s_n\}$ is contained in $\mathcal{S}(S, \sigma_s)$. The expected running time T of the whole algorithm, denoting $o = |\mathcal{S}(S, \sigma_S)|$, obeys the recursive relation

$$T(o) \le 2 \cdot T\left(\frac{1}{2}(o+1)\right) + \widetilde{O}(o) \cdot \log^3 \sigma_s.$$

The above yields $T(o) = \widetilde{O}(o) \cdot \log^3 \sigma_s$, as desired. □

4.3.5 Algorithm for general t

Handling Large Elements

The instance consisting solely of large items is much easier to solve, since only a polylogarithmic number of elements can participate in a subset sum which is at most t. We use a color-coding argument, similar to the one in [Bri17]. In what follows, t' is an upper bound on t. We will see why we need this slight variant of the problem in section 4.4.

Lemma 4.3.10 (Guarantee for large elements). *Given positive integers $t \le t'$ and a set S of* *non-negative integers elements, such that every $x \in S$ satisfies $x \geq t/\log^c t'$, we can find $\mathcal{S}(S,t)$ using Algorithm 21 in expected time*

$$\widetilde{O}(\alpha^{-2}|\mathcal{S}(S,t(1+\alpha))|)\cdot\log^3 t\log(t'/\delta)$$

with probability $1-\delta$.

Proof. We pick $R = \Theta(\log(t/\delta))$ pairwise independent hash functions $h_r : S \to [2\log^{2c} t']$, for $r \in [R]$. For every $r \in R$ we denote $Z_{r,j} = h_r^{-1}(j) \cup \{0\}$ and compute the sumset

$$S_r = \left(Z_{r,1} + Z_{r,2} + \ldots Z_{r,2\log^{2c} t'}\right) \cap [t]$$

as in Algorithm 21. We first prove correctness. First of all, observe that $S_r \subseteq \mathcal{S}(S,t)$ for any $r \in [R]$, and hence $\cup_{r\in[R]} S_r \subseteq \mathcal{S}(S,t)$. So it suffices to prove that $\mathcal{S}(S,t) \subseteq \cup_{r\in[R]} S_r$.

For a fixed $x \in \mathcal{S}(S,t)$, the assumption on the integers in S yields that $|I(x)| \leq \log^c t'$. We now observe that elements in $I_{\mathcal{S}(S,t)}(x)$ will be perfectly hashed under h_r with $1/2$ probability. This yields that $x \in S_r$, since the value x can be attained by picking every element $c \in I_{\mathcal{S}(S,t)}(x)$ from one of the buckets it has been hashed to under h_r, and 0 from every other (empty) bucket. A standard repetition argument gives that $I_{\mathcal{S}(S,t)}(x)$ will be perfectly hashed under at least one h_r with probability $1-1/(t/\delta)$ due to the choice of R. Now, by a union-bound we have that for every $x \in \mathcal{S}(S,t)$, the set $I_{\mathcal{S}(S,t)}(x)$ is perfectly hashed under some h_r with probability $1-|\mathcal{S}(S,t)|/(t/\delta) \geq 1-\delta$. This means that $x \in \cup_{r\in[R]} S_r$, and hence $\cup_{r\in[R]} S_r \subseteq \mathcal{S}(S,t)$. This finishes the proof of correctness.

For the running time, we invoke Algorithm from Lemma 4.3.5 giving total expected running time (ignoring constants and any logarithmic factors)

$$\sum_{r\in[R]}\sum_{2\leq\ell\leq\log^2 cn}\alpha^{-2}|\mathcal{S}(\cup_{j\leq\ell}Z_{r,j},(1+\alpha)t)| \leq$$
$$\alpha^{-2}(R-1)(\log^{2c} t'-1)|\mathcal{S}(S,(1+\alpha)t)|,$$

from which the main result follows. □

Algorithm 22

```
1: procedure OutputSensitiveSubsetSum($S, t$)                                ▷ $\forall x \in S, x \le t$
2:     $n \leftarrow |S|$
3:     $O \leftarrow S \cap [0, t/n]$
4:     for $i = 1$ to $\lceil \log n \rceil$ do
5:         $L_i \leftarrow \emptyset$
6:         for $\ell = 1$ to $5 \log t$ do                                   ▷ Boosting success probability
7:             $L_i \leftarrow L_i \cup$ OutputSensitiveSubsetSumPerLayer $\left(S \cap \left[\frac{t}{2^i} + 1, \frac{2t}{2^i}\right], t, t\right)$
8:         end for
9:     end for
10:    for $i = 1$ to $\lceil \log n \rceil$ do                               ▷ Combine all layers
11:        $O \leftarrow (O + L_i) \cap [t]$                  ▷ Capped Sumset Computation via Lemma 4.3.5
12:    end for
13:    Return $O$
14: end procedure
```

Algorithm 23

```
1: procedure OutputSensitiveSubsetSumPerLayer($S, t, t'$)      ▷ $t \le t'$ and $\forall x \in S, x \le t$
2:     if $\min(S) \ge \frac{t}{\log^c t'}$ then
3:         Return SubsetSumforLargeElements $\left(S, t, t', (t')^{-5}\right)$
4:     end if
5:     if $\sigma_S < 2t$ then
6:         $x \leftarrow$ maximum element of $S$                          ▷ assume that $S$ is sorted
7:         $(S_1, S_2) \leftarrow$ (odd elements of $S \setminus \{x\}$, even elements of $S \setminus \{x\}$)   ▷ $\sigma_{S_1}, \sigma_{S_2} \le t$
8:         $O_1 \leftarrow \mathcal{S}(S_1, \sigma_{S_1})$                                ▷ Theorem 4.3.7
9:         $O_2 \leftarrow \mathcal{S}(S_2, \sigma_{S_2})$                                ▷ Theorem 4.3.7
10:        Return $((O_1 + O_2) \cap [u] + \{x\}) \cap [u]$
11:    end if
12:    Randomly partition $S$ to $(S_1, S_2)$ with $|S_1 - S_2| \le 1$. ▷ Split $S$ as evenly as possible.
13:    $\epsilon \leftarrow \frac{1}{\log t'}$
14:    $O_1 \leftarrow$ OutputSensitiveSubsetSumPerLayer $\left(S_1, (1+\epsilon)\frac{t}{2}, t'\right)$
15:    $O_2 \leftarrow$ OutputSensitiveSubsetSumPerLayer $\left(S_2, (1+\epsilon)\frac{t}{2}, t'\right)$
16:    Return $(O_1 + O_2) \cap [t]$                   ▷ Capped Sumset Computation via Lemma 4.3.5
17: end procedure
```

4.4 General Algorithm

Our approach randomly partitions the input to two equal-sized sets S_1, S_2. Our main algorithm is Algorithm 22. Similar to [Bri17] the algorithm splits the set to layers, and solves each layer separately, by calling Algorithm 23.

Bounding the Running Time

The following lemma is the crux of our argument on bounding the running time. In a nutshell, this lemma says that if all elements are small compared to the target, then any way of splitting S to two parts X and Y forces the number of subset sums of X and Y to be sufficiently less than the number of subset sums of S.

Lemma 4.4.1 (Recursive Splitting Condition)**.** *Let S be a set of numbers, partitioned into X and Y. Let t be such that $\sigma_S \geq 2t$. Set $\mu = \frac{\max(S)}{t}$ and assume μ to be at most a sufficiently small constant. Then for any ϵ we have*

$$\left|\mathcal{S}\left(X, (1+\epsilon)\frac{t}{2}\right)\right| + \left|\mathcal{S}\left(Y, (1+\epsilon)\frac{t}{2}\right)\right| - 1 \leq \frac{|\mathcal{S}(Z,t)|}{1-4\epsilon-8\mu}$$

Proof. We denote

$$C = \mathcal{S}(S,t),$$
$$A = \mathcal{S}\left(X, (1+\epsilon)\frac{t}{2}\right),$$
$$B = \mathcal{S}\left(Y, (1+\epsilon)\frac{t}{2}\right).$$

We also let $A' = A \cap \left[(1-\epsilon-2\mu)\frac{t}{2}, (1+\epsilon)\frac{t}{2}\right]$. The statement is trivial if $\max(A) \leq (1-\epsilon)\frac{t}{2}$ (or $\max(B) \leq (1-\epsilon)\frac{t}{2}$), since then mapping every $x \in B \setminus \{0\}$ to $\sigma_{I_B(x)\cup X} = x + \sigma_X$ (equivalently, map $I_B(x)$ to $I_B(x) \cup X$) we obtain $|B| - 1$ distinct subset sums above $\max(A)$, yielding $|C| \geq |A| + |B| - 1$. We will need the following two claims:

Claim 4.4.2. $|C| \geq |A| + |B| - |A'| - 1$

Proof. Note that

$$\left|C \cap \left[0, (1-\epsilon-2\mu)\frac{t}{2}\right]\right| \geq \left|A \cap \left[0, (1-\epsilon-2\mu)\frac{t}{2}\right]\right| = |A| - |A'|$$

Moreover, since all items are bounded by μt, we can choose $P \subseteq X$ such that

$$\sigma_P \in \left[(1-\epsilon-2\mu)\frac{t}{2}, (1-\epsilon)\frac{t}{2}\right].$$

To see that, let any ordering of elements of X, initialize P to the empty set, and start adding elements to it one by one; clearly at some point σ_P will fall inside the aforementioned interval. Now, for every $x \in B$, we map x to $\sigma_{I_B(x) \cup P} = \sigma_x + \sigma_P$ (equivalently, map $I_B(x)$ to $I_B(x) \cup P$, to obtain $|B| - 1$ different sums, all in the interval $\left[(1-\epsilon-2\mu)\frac{t}{2}+1, t\right]$, and thus disjoint from the numbers in A counted above. Thus, we obtain at least $(|A| - |A'|) + (|B| - 1)$ numbers in C.

□

Claim 4.4.3. $|C| \geq |A| + \frac{|A'|}{2\epsilon+4\mu-2}$

Proof. In the interval $\left[0, (1+\epsilon)\frac{t}{2}\right]$ we simply count the numbers in A. Above that interval, we argue as follows. Since all items are at most μt, using a totally analogous argument as in the previous claim, we can find prefixes $P_i \subseteq Y$ such that

$$\sigma_{P_i} \in \left[i(\epsilon+2\mu)t, i(\epsilon+2\mu)t + \mu t)\right],$$

for all i satisfying

$$i(\epsilon+2\mu)t + \mu t \leq \max(B).$$

Let us call such i good. Since $\max(B) \geq (1-\epsilon)\frac{t}{2}$ by an earlier argument, all i smaller than

$$\begin{aligned}\frac{(1-\epsilon)\frac{t}{2}}{(\epsilon+2\mu)t} &= \frac{1-\epsilon}{2\epsilon+4\mu} \\ &\geq \frac{1}{2\epsilon+4\mu} - 2.\end{aligned}$$

are good.

For any $x \in A'$ map $I_{A'}(x)$ to $I_{A'}(x) \cup P_i$, to obtain $|A'|$ different numbers in the interval

$$\left[i(\epsilon+2\mu)t + (1-\epsilon-2\mu)\frac{t}{2}, (i+1)(\epsilon+2\mu)t + (1-\epsilon-2\mu)\frac{t}{2}\right].$$

These intervals are pairwise disjoint as well as disjoint from the initial interval $\left[0, (1+\epsilon)\frac{t}{2}\right]$.

In order for all generated sums to be at most t, we also need $(i+1)(\epsilon+2\mu)t + (1-\epsilon-$

$2\mu)\frac{t}{2} \leq t$, which boils down to $i \leq (1+\epsilon+2\mu)/(2\epsilon+4\mu) - 1$. Since i is an integer, we have at least $1/(2\epsilon+4\mu) - 1$ valid i's. Hence, we obtain

$$|C| \geq |A| + |A'| \left(\frac{1}{2\epsilon + 4\mu} - 1 \right).$$

□

We now combine the two claims, by considering two cases:

- Case 1: $|A'| \leq (2\epsilon + 4\mu)|B|$ Then

$$|C| \geq |A| + |B| - |A'| - 1 \geq |A| + |B|(1 - 2\epsilon - 2\mu) \geq (|A| + |B|)(1 - 2\epsilon - 2\mu) - 1.$$

- Case 2: $|A'| \geq (2\epsilon + 4\mu) \cdot |B|$ Then

$$\begin{aligned} |C| \geq |A| + \frac{|A'|}{2\epsilon + 4\mu - 2} &\geq \\ |A| + |B|(1 - 4\epsilon - 8\mu) &\geq \\ (|A| + |B|)(1 - 4\epsilon - 8\mu). \end{aligned}$$

□

Lemma 4.4.4 (Bound on the Running Time)**.** *The procedure* OutputSensitiveSubsetSum*(S, t) has expected running time (ignoring any logarithmic factor) at most*

$$\alpha^{-2} \left| \mathcal{S}\left(S, (1+\alpha)t\right) \right|.$$

Proof. It clearly suffices to prove that OutputSensitiveSubsetSumPerLayer(S, t, n) runs in expected $(|\alpha^{-2}\mathcal{S}(S, (1+\alpha)t)|) \cdot \text{poly}(\log(nt))$ time.

If every element of S is at least $t/\log^c t$, we immediatelly get the desired result by Lemma 4.3.10. Otherwise, the algorithm will split the set to two sets S_1, S_2 and recurse on each one. If we are initally in layer i then every element is at least $t/2^i$, the recursion tree will have depth roughly $\Theta(i - \log\log t)$, and number of nodes $\Theta(2^i/\log^c t)$. In the leaves of the recursion OutputSensitiveSubsetSumForLargeElements will be called. If the set

S satisfies $\sigma < 2t$, then the result will follow by theorem 4.3.7. Otherwise, by invoking Lemma 4.4.1 (in every call $\mu = \max(S)/t$ is smaller than some absolute constant) and since $\epsilon = 1/\log n$, by an inductive argument we can bound the total running time per level by $|\mathcal{S}(S,t)| \cdot \mathrm{poly}(\log n)$. The claim then follows.

□

Proof of Correctness

We now prove that OutputSensitiveSubsetSum(S,t) returns $\mathcal{S}(S,t)$ with probability 99/100. Combining this result with Lemma 4.4.4, we obtain Theorem 4.3.2. Similarly to [Bri17], we perform a coloring coding argument to argue correctness.

Fix $x \in \mathcal{S}(S,t)$. In the following we will write $I(x)$ meaning $I_{\mathcal{S}(x)}$. We will argue that the algorithm will include it in the final output with probability $1 - 99/(100t)$; then the result will follow by a union bound. Moreover, it suffices to prove that for all $i = 1, 2, \ldots, \lceil \log n \rceil$, with probability $1 - (\log n + 1) \cdot 99/(100t)$ the element

$$\sigma_{I(x) \cap [t2^{-1}+1, t2^{-i}]}$$

will belong to the output of

$$\textsc{OutputSensitiveSubsetSumPerLayer}(x, S \cap [t2^{-i+1} + 1, t2^{-i}], t),$$

as well as a similar fact for $\sigma_{I(x) \cap [0,t/n]}$. If this is the case, by a union-bound we will get that

$$x = \sigma_{I(x)} = \sigma_{I(x) \cap [0,t/n]} + \sum_{i=1}^{\lceil \log n \rceil} \sigma_{S \cap [t2^{-i+1}+1, t2^{-i}]}$$

belongs to the final list of elements. For that, since the failure probability of the leaves of the recursion tree of the call SubsetsumPerLayerPerLayer calls in OutputSensitiveSubset-Sum succeed with probability $1 - t^{-5}$, the only thing we need to prove is the following. For every set T wit containing elements less than $t/\log^c t'$, and satisfying $\sigma_T \leq t$ the following holds. If we split it randomly to T_1, T_2 (line 12) then with probability $1 - 1/\mathrm{poly}(t')$ we have that $\sigma_{T_1} \leq (1+\epsilon)\frac{t}{2}$ and $\sigma_{T_2} \leq (1+\epsilon)\frac{t}{2}$. To do that we will invoke the following version of the Chernoff-Hoeffding bound: given random variables $Z_1, \dots, Z_N$ identically distributed in $[0, M]$ with $\mathbf{E}\, Z_i = \mu$ for all i, for $\zeta \le 1/2$ we have that

$$\Pr\left[\left|\sum_{i=1}^{N} Z_i - N\mu\right| > \zeta\right] \le e^{-CN\mu\zeta^2/M},$$

where C is an absolute constant. In our case, if $\sigma_T \le \frac{t}{2}$ there is nothing to prove, otherwise by Chernoff-Hoeffding with $M = t/\log^c t'$ and $\zeta = 1/\log t'$ we obtain that $\sigma_{T_1} \in [(1-\zeta)\sigma_T, (1+\zeta)\sigma_T]$ with probability $1 - 1/\operatorname{poly}(t')$, if $c \ge 3$. This failure probability allows for a union-bound over every splitting in OutputSensitiveSubsetSumPerLayer.

Chapter 5

Conclusion and Open Problems

In this thesis we presented several efficient solutions to fundamental problems in sparse/recovery compressed sensing, all of them being nearly optimal, as well as (nearly or almost) optimal algorithms for classical problems such as polynomial multiplication, Subset Sum and Modular Subset Sum. Many new facinating questions remain open or arise from our work.

5.1 Non-Linear Compressed Sensing

OPEN: Devise an algorithm for compressed sensing from intensity-only measurements which uses $O(k \log(n/k))$ measurements and runs in time $O(k \log^c n)$.

OPEN: Devise an algorithm for $\delta - \ell_2/\ell_2$ One-Bit Compressed Sensing which runs in time $\widetilde{O}(k \log^c n)$ and uses $O(k \log n + \delta^{-2} k)$ measurements.

5.2 Sparse Fourier Transform

OPEN: Does there exist a sublinear-time dimension-free Sparse Fourier Transform algorithm, achieving nearly optimal sample complexity?

OPEN: Devise an ℓ_∞/ℓ_2 Sparse Fourier Transform algorithm with $O(k \log n)$ samples in any dimension.

OPEN: Does there exist a strongly explicit construction of an ℓ_∞/ℓ_1 scheme with $O(k^2 \log n)$ measurements?

OPEN: Does there exist a for-all ℓ_∞/ℓ_1 scheme for the Sparse Fourier Transform which uses $O(k^2 \log n)$ samples and runs in sublinear time?

OPEN (Possibly very hard, if not impossible): Does there exist a for-all ℓ_2/ℓ_1 scheme for the Sparse Fourier Transform running time $o(k^2) \cdot \log^c n$?

OPEN: Does there exist a polynomial time algorithm for finding a subset of $O(k^2(\log_k n)^2)$ rows of the DFT matrix, such that the resulting matrix is $1/k$-incoherent?

OPEN: Does there exist a strongly explicit construction of an $1/k$-incoherent matrix with $O(k^2 \log n)$ rows? The same when the rows of the matrix should be a subset of the rows of the DFT matrix.

OPEN (Possibly very hard): Does there exist a $1/k$-incoherent with $O(k^2 \log_k n)$ rows?

5.3 Polynomial Multiplication and Sumset Computation

OPEN: Devise a $O(k \log n)$-time algorithm[1] for sparse polynomial multiplication, where k is the size of the input plus the size of the output? Such an algorithm would outperform FFT for any $k = o(n)$.

OPEN: Does there exist a deterministic algorithm for finding the sumset of two sets in output-sensitive time?

[1] $\log \log n$ factors are prohibited!

5.4 Subset Sum

OPEN: Can you find a deterministic algorithm for Modular Subset Sum in nearly linear time **without** using a deterministic algorithm for output-sensitive sumset computation? Even for the case where m is a prime number such a result would be very interesting.

OPEN: Devise an output-sensitive algorithm for Subset Sum, running time $O(|\mathcal{S}(S,t)| \cdot \text{poly}(\log t))$?

OPEN: Does there exist a deterministic algorithm for Subset Sum faster than $O\left(\min\{\sqrt{n}t, t^{5/4}\}\right)$ which is achieved by [KX17]? Note that our techniques imply a deterministic algorithm running in time

$$\min\{\sqrt{n}|\mathcal{S}(S,2t)|^{1+o(1)}, |\mathcal{S}(S,2t)|^{5/4(1+o(1))}\}.$$

OPEN: Without a dependence on t, which is the best algorithm for Subset Sum in terms of $|S|$?

References

[ABHS19] Amir Abboud, Karl Bringmann, Danny Hermelin, and Dvir Shabtay. Seth-based lower bounds for subset sum and bicriteria path. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 41–57. SIAM, 2019.

[ABJ$^+$19] Kyriakos Axiotis, Arturs Backurs, Ce Jin, Christos Tzamos, and Hongxun Wu. Fast modular subset sum using linear sketching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 58–69. SIAM, 2019.

[ABK17] Jayadev Acharya, Arnab Bhattacharyya, and Pritish Kamath. Improved bounds for universal one-bit compressive sensing. *arXiv preprint arXiv:1705.00763*, 2017.

[AGS03] Adi Akavia, Shafi Goldwasser, and Shmuel Safra. Proving hard-core predicates using list decoding. In *FOCS*, volume 44, pages 146–159, 2003.

[AKKM13] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jussi Määttä. Space–time tradeoffs for Subset Sum: An improved worst case algorithm. In *Proc. of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 45–56, 2013.

[AKKN15] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Subset Sum in the absence of concentration. In *Proc. of the 32nd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 48–61, 2015.

[AKKN16] Per Austrin, Petteri Kaski, Mikko Koivisto, and Jesper Nederlof. Dense Subset Sum may be the hardest. In *Proc. of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 13:1–13:14, 2016.

[AKM$^+$18] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. A universal sampling method for reconstructing signals with simple fourier transforms. *arXiv preprint arXiv:1812.08723*, 2018.

[Alo87] Noga Alon. Subset sums. *Journal of Number Theory*, 27(2):196–205, 1987.

[Alo09] Noga Alon. Perturbed identity matrices have high rank: Proof and applications. *Combinatorics, Probability and Computing*, 18(1-2):3–15, 2009.

[AM11] Arash Amini and Farokh Marvasti. Deterministic construction of binary, bipolar, and ternary compressed sensing matrices. *IEEE Transactions on Information Theory*, 57(4):2360–2370, 2011.

[AR15] Andrew Arnold and Daniel S Roche. Output-sensitive algorithms for sumset and sparse polynomial multiplication. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation*, pages 29–36. ACM, 2015.

[ASSN08] Abiodun M Aibinu, Momoh JE Salami, Amir A Shafie, and Athaur Rahman Najeeb. MRI reconstruction using discrete Fourier transform: a tutorial. *World Academy of Science, Engineering and Technology*, 42:179, 2008.

[AZGR16] Zeyuan Allen-Zhu, Rati Gelashvili, and Ilya Razenshteyn. Restricted isometry property for general p-norms. *IEEE Transactions on Information Theory*, 62(10):5839–5854, 2016.

[Baw75] Vijay S Bawa. Optimal rules for ordering uncertain prospects. *Journal of Financial Economics*, 2(1):95–121, 1975.

[BB08] Petros T Boufounos and Richard G Baraniuk. 1-bit compressive sensing. In *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pages 16–21. IEEE, 2008.

[BCDH10] Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

[BCG$^+$14] Petros Boufounos, Volkan Cevher, Anna C Gilbert, Yi Li, and Martin J Strauss. What's the frequency, Kenneth?: Sublinear Fourier sampling off the grid. In *Algorithmica(A preliminary version of this paper appeared in the Proceedings of RANDOM/APPROX 2012, LNCS 7408, pp.61–72)*, pages 1–28. Springer, 2014.

[BCJ11] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. Improved generic algorithms for hard knapsacks. In *Proc. of 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 364–385, 2011.

[BD08] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5-6):629–654, 2008.

[BD09a] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

[BD09b] Thomas Blumensath and Mike E Davies. A simple, efficient and near optimal algorithm for compressed sensing. *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009.

[BD10] Thomas Blumensath and Mike E Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of selected topics in signal processing*, 4(2):298–309, 2010.

[BDF$^+$11] Jean Bourgain, Stephen J Dilworth, Kevin Ford, Sergei V Konyagin, and Denka Kutzarova. Breaking the k 2 barrier for explicit rip matrices. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 637–644. ACM, 2011.

[BDP05] Ilya Baran, Erik D Demaine, and Mihai Pătraşcu. Subquadratic algorithms for 3sum. In *Workshop on Algorithms and Data Structures*, pages 409–421. Springer, 2005.

[Bel57] Richard E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

[BFN$^+$16] Rich Baraniuk, Simon Foucart, Deanna Needell, Yaniv Plan, and Mary Wootters. One-bit compressive sensing of dictionary-sparse signals. *arXiv preprint arXiv:1606.07531*, 2016.

[BFN$^+$17] Richard G Baraniuk, Simon Foucart, Deanna Needell, Yaniv Plan, and Mary Wootters. Exponential decay of reconstruction error from binary measurements of sparse signals. *IEEE Transactions on Information Theory*, 63(6):3368–3385, 2017.

[BGNV17] Nikhil Bansal, Shashwat Garg, Jesper Nederlof, and Nikhil Vyas. Faster space-efficient algorithms for Subset Sum, k-Sum and related problems. In *Proc. of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 198–209, 2017.

[BI08] Radu Berinde and Piotr Indyk. Sparse recovery using sparse random matrices. *preprint*, 2008.

[BIR08] Radu Berinde, Piotr Indyk, and Milan Ruzic. Practical near-optimal sparse recovery in the ℓ_1 norm. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 198–205. IEEE, 2008.

[BIRW16] Arturs Backurs, Piotr Indyk, Ilya Razenshteyn, and David P Woodruff. Nearly-optimal bounds for sparse recovery in generic norms, with applications to k-median sketching. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 318–337. SIAM, 2016.

[BLLM19] Jaroslaw Blasiok, Patrick Lopatto, Kyle Luh, and Jake Marcinek. Sparse reconstruction from hadamard matrices: A lower bound. In *arXiv preprint*. `https://arxiv.org/pdf/1903.12135.pdf`, 2019.

[BLM13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic theory of Independence*. Oxford University Press, 2013.

[BO88] Ernest F Brickell and Andrew M Odlyzko. Cryptanalysis: A survey of recent results. *Proceedings of the IEEE*, 76(5):578–593, 1988.

[Bou14] Jean Bourgain. An improved estimate in the restricted isometry problem. In *Geometric Aspects of Functional Analysis*, pages 65–70. Springer, 2014.

[Bri17] Karl Bringmann. A near-linear pseudopolynomial time algorithm for Subset Sum. In *Proc. of of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1073–1084, 2017.

[BZI17] Sina Bittens, Ruochuan Zhang, and Mark A Iwen. A deterministic sparse FFT for functions with structured Fourier sparsity. *Advances in Computational Mathematics, to appear.*, 2017.

[CBJC14] Sheng Cai, Mayank Bakshi, Sidharth Jaggi, and Minghua Chen. Super: Sparse signals with unknown phases efficiently recovered. *arXiv preprint arXiv:1401.4269*, 2014.

[CCF02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming*, pages 693–703. Springer, 2002.

[CDD09] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Compressed sensing and best k-term approximation. *Journal of the American mathematical society*, 22(1):211–231, 2009.

[CEPR07] Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. k-mismatch with don't cares. *Algorithms–ESA 2007*, pages 151–162, 2007.

[CGV13] Mahdi Cheraghchi, Venkatesan Guruswami, and Ameya Velingker. Restricted isometry of Fourier matrices and list decodability of random linear codes. *SIAM Journal on Computing*, 42(5):1888–1914, 2013.

[CH02] Richard Cole and Ramesh Hariharan. Verifying candidate matches in sparse and wildcard matching. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 592–601. ACM, 2002.

[CH09] Graham Cormode and Marios Hadjieleftheriou. Finding the frequent items in streams of data. *Communications of the ACM*, 52(10):97–105, 2009.

[CI17] Mahdi Cheraghchi and Piotr Indyk. Nearly optimal deterministic algorithm for sparse walsh-hadamard transform. *ACM Transactions on Algorithms (TALG)*, 13(3):34, 2017.

[CIHB09] Volkan Cevher, Piotr Indyk, Chinmay Hegde, and Richard G Baraniuk. Recovery of clustered sparse signals from compressive measurements. Technical report, RICE UNIV HOUSTON TX DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, 2009.

[Cip00] Barry A Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM news*, 33(4):1–2, 2000.

[CJK$^+$04] Graham Cormode, Theodore Johnson, Flip Korn, Shan Muthukrishnan, Oliver Spatscheck, and Divesh Srivastava. Holistic UDAFs at streaming speeds. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 35–46. ACM, 2004.

[CKPS16] Xue Chen, Daniel M Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 741–750. IEEE, 2016.

[CKSZ17] Volkan Cevher, Michael Kapralov, Jonathan Scarlett, and Amir Zandieh. An adaptive sublinear-time block sparse Fourier transform. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, https://arxiv.org/pdf/1702.01286, 2017.

[CL15] Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proc. of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 31–40, 2015.

[CLS15a] E. J. Candès, X. Li, and M. Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, April 2015.

[CLS15b] Emmanuel J. Candès, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval from coded diffraction patterns. *Applied and Computational Harmonic Analysis*, 39(2):277 – 299, 2015.

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51th Annual Symposium on the Theory of Computing (STOC)*. https://arxiv.org/pdf/1810.07896, 2019.

[CM04] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. In *Proceedings of the Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2004.

[CM06] Graham Cormode and S Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 198–201. IEEE, 2006.

[CR88] Benny Chor and Ronald R. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, 1988.

[CRT06] Emmanuel J Candès, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006.

[CSV13] Emmanuel J. Candès, Thomas Strohmer, and Vladislav Voroninski. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Mathematics*, 66(8):1241–1274, 2013.

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[CT05] Emmanuel Candes and Terence Tao. Decoding by linear programming. *arXiv preprint math/0502327*, 2005.

[CT06] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

[DBIPW10] Khanh Do Ba, Piotr Indyk, Eric Price, and David P Woodruff. Lower bounds for sparse recovery. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1190–1197. SIAM, 2010.

[DDKS12] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. In *Proc. of the 32nd Annual Conference on Advances in Cryptology (CRYPTO)*, pages 719–740, 2012.

[DDT$^+$08] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE signal processing magazine*, 25(2):83–91, 2008.

[DDTS06] David Leigh Donoho, Iddo Drori, Yaakov Tsaig, and Jean-Luc Starck. *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*. Department of Statistics, Stanford University, 2006.

[DeV07] Ronald A DeVore. Deterministic constructions of compressed sensing matrices. *Journal of complexity*, 23(4):918–925, 2007.

[DLTY06] Ke-xue Dai, Guo-hui Li, Dan Tu, and Jian Yuan. Prospects and current studies on background subtraction techniques for moving objects detection from surveillance video. *Journal of Image and Graphics*, 7(002), 2006.

[DM08] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing: Closing the gap between performance and complexity. Technical report, ILLINOIS UNIV AT URBANA-CHAMAPAIGN, 2008.

[Don06] David L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006.

[DP09] Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 2009.

[DWG$^+$13] Denisa Duma, Mary Wootters, Anna C Gilbert, Hung Q Ngo, Atri Rudra, Matthew Alpert, Timothy J Close, Gianfranco Ciardo, and Stefano Lonardi. Accurate decoding of pooled sequenced data using compressed sensing. In *International Workshop on Algorithms in Bioinformatics*, pages 70–84. Springer, 2013.

[ECG$^+$09] Yaniv Erlich, Kenneth Chang, Assaf Gordon, Roy Ronen, Oron Navon, Michelle Rooks, and Gregory J Hannon. DNA sudoku-harnessing high-throughput sequencing for multiplexed specimen analysis. *Genome research*, 19(7):1243–1253, 2009.

[EGZ61] Paul Erdos, Abraham Ginzburg, and Abraham Ziv. Theorem in the additive number theory. *Bull. Res. Council Israel F*, 10:41–43, 1961.

[End10] Joachim HG Ender. On compressive sensing applied to radar. *Signal Processing*, 90(5):1402–1414, 2010.

[ER59] Paul Erdős and Alfréd Rényi. On random graphs I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959 1959.

[ESAZ09] Yaniv Erlich, Noam Shental, Amnon Amir, and Or Zuk. Compressed sensing approach for high throughput carrier screen. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 539–544. IEEE, 2009.

[EV03] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3):270–313, 2003.

[Fou11] Simon Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

[FPRU10] Simon Foucart, Alain Pajor, Holger Rauhut, and Tino Ullrich. The gelfand widths of ℓ_p-balls for $0 \leq p \leq 1$. *Journal of Complexity*, 26(6):629–640, 2010.

[FR13] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser Basel, 2013.

[FSGM$^+$99] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D Ullman. Computing iceberg queries efficiently. In *Internaational Conference on Very Large Databases (VLDB'98), New York, August 1998*. Stanford InfoLab, 1999.

[Gan08] Sumit Ganguly. Lower bounds on frequency estimation of data streams. In *International Computer Science Symposium in Russia*, pages 204–215. Springer, 2008.

[GGI$^+$02] Anna C Gilbert, Sudipto Guha, Piotr Indyk, S Muthukrishnan, and Martin Strauss. Near-optimal sparse Fourier representations via sampling. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 152–161. ACM, 2002.

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

[GK09] Rahul Garg and Rohit Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *ICML*, volume 9, pages 337–344, 2009.

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

[GLP$^+$10] C Sinan Güntürk, Mark Lammers, Alex Powell, Rayan Saab, and Özgür Yilmaz. Sigma delta quantization for compressed sensing. In *Information Sciences and Systems (CISS), 2010 44th Annual Conference on*, pages 1–6. IEEE, 2010.

[GLPS10] Anna C Gilbert, Yi Li, Ely Porat, and Martin J Strauss. Approximate sparse recovery: optimizing time and measurements. *SIAM Journal on Computing 2012 (A preliminary version of this paper appears in STOC 2010)*, 41(2):436–453, 2010.

[GLPS17] Anna C Gilbert, Yi Li, Ely Porat, and Martin J Strauss. For-all sparse recovery in near-optimal time. *ACM Transactions on Algorithms (TALG)*, 13(3):32, 2017.

[GMS05] Anna C Gilbert, S Muthukrishnan, and Martin Strauss. Improved time bounds for near-optimal sparse Fourier representations. In *Optics & Photonics 2005*, pages 59141A–59141A. International Society for Optics and Photonics, 2005.

[GNJN13] Sivakant Gopi, Praneeth Netrapalli, Prateek Jain, and Aditya Nori. One-bit compressed sensing: Provable support and vector recovery. In *International Conference on Machine Learning*, pages 154–162, 2013.

[GNP$^+$13] Anna C Gilbert, Hung Q Ngo, Ely Porat, Atri Rudra, and Martin J Strauss. ℓ_2/ℓ_2-foreach sparse recovery with low risk. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2013.

[Goo05] Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company Publishers, 2005.

[GR16] A Whitman Groves and Daniel S Roche. Sparse polynomials in flint. *ACM Communications in Computer Algebra*, 50(3):105–108, 2016.

[GS12] Surya Ganguli and Haim Sompolinsky. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual review of neuroscience*, 35:485–508, 2012.

[GS17] Omer Gold and Micha Sharir. Improved bounds for 3SUM, k-SUM, and linear degeneracy. In *Proc. of the 25th Annual European Symposium on Algorithms (ESA)*, pages 42:1–42:13, 2017.

[GWX16] Bing Gao, Yang Wang, and Zhiqiang Xu. Stable signal recovery from phaseless measurements. *Journal of Fourier Analysis and Applications*, 22(4):787–808, Aug 2016.

[HIKP12a] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse Fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 563–578. ACM, 2012.

[HIKP12b] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse Fourier transform. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1183–1194. SIAM, 2012.

[HMT$^+$13] Hong Huang, Satyajayant Misra, Wei Tang, Hajar Barani, and Hussein Al-Azzawi. Applications of compressed sensing in communications networks. *arXiv preprint arXiv:1305.3002*, 2013.

[HR69] Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American economic review*, 59(1):25–34, 1969.

[HR16] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled Fourier matrices. In *SODA*, pages 288–297. arXiv preprint arXiv:1507.01768, 2016.

[HS74] Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21(2):277–292, 1974.

[IK14] Piotr Indyk and Michael Kapralov. Sample-optimal Fourier sampling in any constant dimension. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 514–523. IEEE, 2014.

[IKP14] Piotr Indyk, Michael Kapralov, and Eric Price. (Nearly) Sample-optimal sparse Fourier transform. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 480–499. SIAM, 2014.

[IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006.

[Ind07] Piotr Indyk. Sketching, streaming and sublinear-space algorithms. *Graduate course notes, available at*, 2007.

[IPSV16] Mark A. Iwen, Brian Preskitt, Rayan Saab, and Aditya Viswanathan. Phase retrieval from local measurements: Improved robustness via eigenvector-based angular synchronization. arXiv:1612.01182 [math.NA], 2016.

[IPW11] Piotr Indyk, Eric Price, and David P Woodruff. On the power of adaptivity in sparse recovery. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 285–294. IEEE, 2011.

[IR08] Piotr Indyk and Milan Ruzic. Near-optimal sparse recovery in the ℓ_1 norm. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 199–207. IEEE, 2008.

[IR13] Piotr Indyk and Ilya Razenshteyn. On model-based RIP-1 matrices. In *International Colloquium on Automata, Languages, and Programming*, pages 564–575. Springer, 2013.

[IVW16] Mark A. Iwen, Aditya Viswanathan, and Yang Wang. Fast phase retrieval from local correlation measurements. *SIAM Journal on Imaging Sciences*, 9(4):1655–1688, 2016.

[IVW17] Mark Iwen, Aditya Viswanathan, and Yang Wang. Robust sparse phase retrieval made easy. *Applied and Computational Harmonic Analysis*, 42(1):135 – 142, 2017.

[Iwe08] Mark A Iwen. A deterministic sub-linear time sparse Fourier algorithm via non-adaptive compressed sensing methods. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 20–29. Society for Industrial and Applied Mathematics, 2008.

[Iwe10] Mark A Iwen. Combinatorial sublinear-time Fourier algorithms. *Foundations of Computational Mathematics*, 10(3):303–338, 2010.

[Iwe13] Mark A Iwen. Improved approximation guarantees for sublinear-time Fourier algorithms. *Applied And Computational Harmonic Analysis*, 34(1):57–82, 2013.

[JDDV13] Laurent Jacques, Kévin Degraux, and Christophe De Vleeschouwer. Quantized iterative hard thresholding: Bridging 1-bit and high-resolution quantized compressed sensing. *arXiv preprint arXiv:1305.1786*, 2013.

[JLBB13] Laurent Jacques, Jason N Laska, Petros T Boufounos, and Richard G Baraniuk. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *IEEE Transactions on Information Theory*, 59(4):2082–2102, 2013.

[JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for ℓ_p samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58. ACM, 2011.

[JW18] Ce Jin and Hongxun Wu. A simple near-linear pseudopolynomial time randomized algorithm for subset sum. *arXiv preprint arXiv:1807.11597*, 2018.

[Kap16] Michael Kapralov. Sparse Fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. In *Symposium on Theory of Computing Conference, STOC'16, Cambridge, MA, USA, June 19-21, 2016*, 2016.

[Kap17] Michael Kapralov. Sample efficient estimation and recovery in sparse FFT via isolation on average. In *Foundations of Computer Science, 2017. FOCS'17. IEEE 58th Annual IEEE Symposium on*. https://arxiv.org/pdf/1708.04544, 2017.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[KBG$^+$10] Raghunandan M Kainkaryam, Angela Bruex, Anna C Gilbert, John Schiefelbein, and Peter J Woolf. PoolMC: Smart pooling of mRNA samples in microarray experiments. *BMC bioinformatics*, 11(1):299, 2010.

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[KPP04] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[KSW16] Karin Knudson, Rayan Saab, and Rachel Ward. One-bit compressive sensing with norm estimation. *IEEE Transactions on Information Theory*, 62(5):2748–2758, 2016.

[KSZC03] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247. ACM, 2003.

[KT75] Samuel Karlin and Howard M. Taylor. *A First Course in Stochastic Processes*. Elsevier, 2nd edition, 1975.

[KVZ19] Michael Kapralov, Ameya Velingker, and Amir Zandieh. Dimension-independent sparse Fourier transform. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2709–2728. SIAM, 2019.

[KX17] Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for Subset Sum. In *Proc. of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1062–1072, 2017.

[LDP07] Michael Lustig, David Donoho, and John M Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic resonance in medicine*, 58(6):1182–1195, 2007.

[LDSP08] Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. Compressed sensing MRI. *IEEE signal processing magazine*, 25(2):72–82, 2008.

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In *Proc. of the 27th 2nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 777–789, 2011.

[LN18] Yi Li and Vasileios Nakos. Sublinear- time algorithms for compressive phase retrieval. In *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*, pages 2301–2305, 2018.

[LN19] Yi Li and Vasileios Nakos. Deterministic sparse Fourier transform with an ℓ_∞ guarantee. *arXiv preprint arXiv:1903.00995*, 2019.

[LNN15] Kasper Green Larsen, Jelani Nelson, and Huy L Nguyên. Time lower bounds for nonadaptive turnstile streaming algorithms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 803–812. ACM, 2015.

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyên, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 61–70. IEEE, https://arxiv.org/pdf/1604.01357, 2016.

[LNW18] Yi Li, Vasileios Nakos, and David P. Woodruff. On low-risk heavy hitters and sparse recovery schemes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 19:1–19:13, 2018.

[LRRB05] Bin Le, Thomas W Rondeau, Jeffrey H Reed, and Charles W Bostian. Analog-to-digital converters. *IEEE Signal Processing Magazine*, 22(6):69–77, 2005.

[LV13] Xiaodong Li and Vladislav Voroninski. Sparse signal recovery from quadratic measurements via convex programming. *SIAM Journal on Mathematical Analysis*, 45(5):3019–3033, 2013.

[LWC13] David Lawlor, Yang Wang, and Andrew Christlieb. Adaptive sub-linear time Fourier algorithms. *Advances in Adaptive Data Analysis*, 5(01):1350003, 2013.

[LWWW16] Andrea Lincoln, Virginia Vassilevska Williams, Joshua R. Wang, and R. Ryan Williams. Deterministic time-space trade-offs for k-SUM. In *Proc. of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 58:1–58:14, 2016.

[LXZL18] Zhilin Li, Wenbo Xu, Xiaobo Zhang, and Jiaru Lin. A survey on one-bit compressed sensing: theory and applications. *Frontiers of Computer Science*, pages 1–14, 2018.

[Man92] Yishay Mansour. Randomized interpolation and approximation of sparse polynomials stpreliminary version. In *International Colloquium on Automata, Languages, and Programming*, pages 261–272. Springer, 1992.

[Maz01] Marc Moreno Maza. Sparse polynomial arithmetic with the bpas library. *Computer Algebra in Scientific Computing: CASC 2001*, 11077:32, 2001.

[MH78] Ralph Merkle and Martin Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, 1978.

[Mix] Dustin Mixon. Deterministic rip martrices. https://math-research.fandom.com/wiki/Deterministic_RIP_Matrices.

[MP07] Michael Monagan and Roman Pearce. Polynomial division using dynamic arrays, heaps, and packed exponent vectors. In *International Workshop on Computer Algebra in Scientific Computing*, pages 295–315. Springer, 2007.

[MP09] Michael Monagan and Roman Pearce. Parallel sparse polynomial multiplication using heaps. In *Proceedings of the 2009 international symposium on Symbolic and algebraic computation*, pages 263–270. ACM, 2009.

[MP14] Michael Monagan and Roman Pearce. Poly: A new polynomial data structure for maple 17. In *Computer Mathematics*, pages 325–348. Springer, 2014.

[MP15] Michael Monagan and Roman Pearce. The design of maple's sum-of-products and poly data structures for representing mathematical objects. *ACM Communications in Computer Algebra*, 48(3/4):166–186, 2015.

[MT90] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[Mut05] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[MZIC17] Sami Merhi, Ruochuan Zhang, Mark A Iwen, and Andrew Christlieb. A new class of fully discrete sparse Fourier transforms: Faster stable implementations with guarantees. *Journal of Fourier Analysis and Applications*, pages 1–34, 2017.

[Nac10] Mergen Nachin. Lower bounds on the column sparsity of sparse recovery matrices. *UAP: MIT Undergraduate Thesis*, 2010.

[Nak17a] Vasileios Nakos. Almost optimal phaseless compressed sensing with sublinear decoding time. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 1142–1146, 2017.

[Nak17b] Vasileios Nakos. On fast decoding of high-dimensional signals from one-bit measurements. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[Nak19] Vasileios Nakos. One-bit expandersketch for one-bit compressed sensing. *ISIT 2019*, 2019.

[Ned17] Jesper Nederlof. A short note on Merlin-Arthur protocols for subset sum. *Information Processing Letters*, 118:15–16, 2017.

[NN13] Jelani Nelson and Huy L Nguyên. Sparsity lower bounds for dimensionality reducing maps. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 101–110. ACM, 2013.

[NNW14] Jelani Nelson, Huy L Nguyên, and David P Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Linear Algebra and its Applications*, 441:152–167, 2014.

[NSV08] Hoi H Nguyen, Endre Szemerédi, and Van H Vu. Subset sums modulo a prime. *Acta Arith*, 131(4):303–316, 2008.

[NSWZ18] Vasileios Nakos, Xiaofei Shi, David P. Woodruff, and Hongyang Zhang. Improved algorithms for adaptive compressed sensing. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 90:1–90:14, 2018.

[NT09a] Deanna Needell and Joel A Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009.

[NT09b] Deanna Needell and Joel A Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009.

[NV09] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of computational mathematics*, 9(3):317–334, 2009.

[NV10] D Needel and R Vershynin. Signal recovery from inaccurate and incomplete measurements via regularized orthogonal matching pursuit. *IEEE Journal of Selected Topics in Signal Processing*, pages 310–316, 2010.

[O'B11] Kevin O'Bryant. Sets of integers that do not contain long arithmetic progressions. *Electronic Journal of Combinatorics*, 18(1):P59, 2011.

[Odl90] Andrew M. Odlyzko. The rise and fall of knapsack cryptosystems. *Cryptology and Computational Number Theory*, 42:75–88, 1990.

[Ols68] John E Olson. An addition theorem modulo p. *Journal of Combinatorial Theory*, 5(1):45–52, 1968.

[Ols75] John E Olson. Sums of sets of group elements. *Acta Arith*, 28(2):147–156, 1975.

[OYDS11] Henrik Ohlsson, Allen Y Yang, Roy Dong, and S Shankar Sastry. Compressive phase retrieval from squared output measurements via semidefinite programming. *arXiv preprint arXiv:1111.6323*, pages 1–27, 2011.

[Phi] Phillips. Compressed sense. https://www.philips.com/healthcare/resources/landing/compressed-sense.

[Pis03] David Pisinger. Dynamic programming on the word RAM. *Algorithmica*, 35(2):128–145, 2003.

[PR08] Ely Porat and Amir Rothschild. Explicit non-adaptive combinatorial group testing schemes. In *International Colloquium on Automata, Languages, and Programming*, pages 748–759. Springer, 2008.

[PR14] Sameer Pawar and Kannan Ramchandran. A robust R-FFAST framework for computing a k-sparse n-length DFT in O(k log n) sample complexity using sparse-graph codes. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 1852–1856. IEEE, 2014.

[Pri11] Eric Price. Efficient sketches for the set query problem. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 41–56. Society for Industrial and Applied Mathematics, 2011.

[Pro95] R. Prony. Essai éxperimental et analytique: sur les lois de la dilatabilité de fluides élastique et sur celles de la force expansive de la vapeur de l'alkool, à différentes températures. *Journal de l'École Polytechnique Floréal et Plairial*, pages 24–76, 1795.

[PS12] Ely Porat and Martin J Strauss. Sublinear time, measurement-optimal, sparse recovery for all. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1215–1227. Society for Industrial and Applied Mathematics, 2012.

[PS15] Eric Price and Zhao Song. A robust sparse Fourier transform in the continuous setting. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 583–600. IEEE, 2015.

[PSL$^+$12] Mark S Pearce, Jane A Salotti, Mark P Little, Kieran McHugh, Choonsik Lee, Kwang Pyo Kim, Nicola L Howe, Cecile M Ronckers, Preetha Rajaraman, Alan W Craft, et al. Radiation exposure from ct scans in childhood and subsequent risk of leukaemia and brain tumours: a retrospective cohort study. *The Lancet*, 380(9840):499–505, 2012.

[PV13a] Yaniv Plan and Roman Vershynin. One-bit compressed sensing by linear programming. *Communications on Pure and Applied Mathematics*, 66(8):1275–1297, 2013.

[PV13b] Yaniv Plan and Roman Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *IEEE Transactions on Information Theory*, 59(1):482–494, 2013.

[PV14] Yaniv Plan and Roman Vershynin. Dimension reduction by random hyperplane tessellations. *Discrete & Computational Geometry*, 51(2):438–461, 2014.

[PW11] Eric Price and David P Woodruff. (1+ϵ)-approximate sparse recovery. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 295–304. IEEE, 2011.

[PYLR17] Ramtin Pedarsani, Dong Yin, Kangwook Lee, and Kannan Ramchandran. Phasecode: Fast and efficient compressive phase retrieval based on sparse-graph codes. *IEEE Transactions on Information Theory*, 63(6):3663–3691, 2017.

[QBI$^+$13] Saad Qaisar, Rana Muhammad Bilal, Wafa Iqbal, Muqaddas Naureen, and Sungyoung Lee. Compressive sensing: From theory to applications, a survey. *Journal of Communications and networks*, 15(5):443–456, 2013.

[Rao19] Sharavas Rao. Improved lower bounds for the restricted isometry property of subsampled fourier matrices. In *arXiv preprint*. https://arxiv.org/pdf/1903.12146.pdf, 2019.

[Rey89] George O Reynolds. *The New Physical Optics Notebook: Tutorials in Fourier Optics.* ERIC, 1989.

[RJ93] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[Roc08] Daniel S Roche. Adaptive polynomial multiplication. *Proc. Milestones in Computer Algebra (MICA'08)*, pages 65–72, 2008.

[Roc11] Daniel S Roche. Chunky and equal-spaced polynomial multiplication. *Journal of Symbolic Computation*, 46(7):791–806, 2011.

[Roc18] Daniel S Roche. What can (and can't) we do with sparse polynomials? *arXiv preprint arXiv:1807.08289*, 2018.

[RV08] Mark Rudelson and Roman Vershynin. On sparse reconstruction from Fourier and Gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008.

[SAZ09] Noam Shental, Amnon Amir, and Or Zuk. Rare-allele detection using compressed se(que)nsing. *arXiv preprint arXiv:0909.0400*, 2009.

[Sch76] W. M. Schmidt. *Equations over Finite Fields: An Elementary Approach*. Lecture Notes in Mathematics. Springer-Verlag Berlin Heidelberg, 1976.

[SEC$^+$15] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev. Phase retrieval with application to optical imaging: A contemporary overview. *IEEE Signal Processing Magazine*, 32(3):87–109, 2015.

[Sha84] Adi Shamir. A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Transactions on Information Theory*, 30(5):699–704, 1984.

[Sie] Siemens. Compressed sensing beyond speed. https://www.healthcare.siemens.com/magnetic-resonance-imaging/clinical-specialities/compressed-sensing.

[Spi96] Daniel A Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

[SV16] Damian Straszak and Nisheeth K Vishnoi. Irls and slime mold: Equivalence and convergence. *arXiv preprint arXiv:1601.02712*, 2016.

[Sze70] Endre Szemerédi. On a conjecture of erdös and heilbronn. *Acta Arithmetica*, 17(3):227–229, 1970.

[Tao03] Terence Tao. An uncertainty principle for cyclic groups of prime order. *arXiv preprint math/0308286*, 2003.

[TG07] Joel Tropp and Anna C Gilbert. Signal recovery from partial information via orthogonal matching pursuit. *IEEE Trans. Inform. Theory*, 53(12):4655–4666, 2007.

[TLW$^+$06] Dharmpal Takhar, Jason N Laska, Michael B Wakin, Marco F Duarte, Dror Baron, Shriram Sarvotham, Kevin F Kelly, and Richard G Baraniuk. A new compressive imaging camera architecture using optical-domain compression. In *Computational Imaging IV*, volume 6065, page 606509. International Society for Optics and Photonics, 2006.

[TMB$^+$17] Charalampos E. Tsourakakis, Michael Mitzenmacher, Jarosław Błasiok, Ben Lawson, Preetum Nakkiran, and Vasileios Nakos. Predicting positive and negative links with noisy queries: Theory & practice. Allerton 2018, 2017.

[Tro15] Joel A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends®in Machine Learning*, 8(1-2):1–230, 2015.

[TV06] Terence Tao and Van H Vu. *Additive combinatorics*, volume 105. Cambridge University Press, 2006.

[TZ12] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal on Computing*, 41(2):293–331, 2012.

[VDHL12] Joris Van Der Hoeven and Grégoire Lecerf. On the complexity of multivariate blockwise polynomial multiplication. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, pages 211–218. ACM, 2012.

[VDHL13] Joris Van Der Hoeven and Grégoire Lecerf. On the bit-complexity of sparse polynomial and series multiplication. *Journal of Symbolic Computation*, 50:227–254, 2013.

[Voe11] David G Voelz. *Computational Fourier Optics: A MATLAB Tutorial (SPIE Tutorial Texts Vol. TT89)*. SPIE press, 2011.

[Vu08] Van Vu. A structural approach to subset-sum problems. In *Building Bridges*, pages 525–545. Springer, 2008.

[Wal99] Robert H Walden. Analog-to-digital converter survey and analysis. *IEEE Journal on selected areas in communications*, 17(4):539–550, 1999.

[Wil05] Herbert S Wilf. *Generating functionology*. AK Peters/CRC Press, 2005.

[Yan98] Thomas Yan. The geobucket data structure for polynomials. *Journal of Symbolic Computation*, 25(3):285–293, 1998.

[YLPR15] Dong Yin, Kangwook Lee, Ramtin Pedarsani, and Kannan Ramchandran. Fast and robust compressive phase retrieval with sparse-graph codes. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2583–2587. IEEE, 2015.

[ZBSG05] Yunhui Zheng, David J Brady, Michaell E Sullivan, and Bob D Guenther. Fiber-optic localization by geometric space coding with a two-dimensional gray code. *Applied optics*, 44(20):4306–4314, 2005.

[Zol86] Vladimir M Zolotarev. *One-dimensional stable distributions*, volume 65. American Mathematical Soc., 1986.

[ZPB06] Yunhui Zheng, Nikos P Pitsianis, and David J Brady. Nonadaptive group testing based fiber sensor deployment for multiperson tracking. *IEEE Sensors Journal*, 6(2):490–494, 2006.