Sparse Fourier Transform in Any Constant Dimension with Nearly-Optimal Sample Complexity in Sublinear Time*

Michael Kapralov
EPFL
Lausanne, Switzerland
michael.kapralov@epfl.ch

ABSTRACT

We consider the problem of computing a k-sparse approximation to the Fourier transform of a length N signal. Our main result is a randomized algorithm for computing such an approximation (i.e. achieving ℓ_2/ℓ_2 sparse recovery guarantees using Fourier measurements) using $O_d(k \log N \log\log N)$ samples of the signal in time domain and $O_d(k \log^{d+3} N)$ runtime, where $d \geq 1$ is the dimensionality of the Fourier transform. The sample complexity matches the $\Omega(k \log(N/k))$ lower bound for non-adaptive algorithms due to Do Ba et al (SODA'10) for any $k \leq N^{1-\delta}$ for a constant $\delta > 0$ up to an $O(\log\log N)$ factor. Prior to our work a result with comparable sample complexity $k \log N \log^{O(1)} \log N$ and sublinear runtime was known for the Fourier transform on the line, but for any dimension $d \geq 2$ previously known techniques either suffered from a poly$(\log N)$ factor loss in sample complexity or required $\Omega(N)$ runtime.

Categories and Subject Descriptors

F.0 [**Theory of Computation**]: General

Keywords

sublinear algorithms, Sparse Fourier Transform

1. INTRODUCTION

The Discrete Fourier Transform (DFT) is a fundamental mathematical concept that allows to represent a discrete signal of length N as a linear combination of N pure harmonics, or frequencies. The development of a fast algorithm for Discrete Fourier Transform, known as FFT (Fast Fourier Transform) in 1965 revolutionized digital signal processing, earning FFT a place in the top 10 most important algorithms of the twentieth century [Cip00]. Fast Fourier Transform (FFT) computes the DFT of a length N signal in time $O(N \log N)$, and finding a faster algorithm for DFT is a major open problem in theoretical computer science. While FFT applies to general signals, many of the applications of FFT (e.g. image and video compression schemes such as JPEG and MPEG) rely on the fact that the Fourier spectrum of signals that arise in practice can often be approximated very well by only a few of the top Fourier coefficients, i.e. practical signals are often (approximately) *sparse* in the Fourier basis.

Besides applications in signal processing, the Fourier sparsity property of real world signal plays and important role in medical imaging, where the cost of *measuring a signal*, i.e. *sample complexity*, is often a major bottleneck. For example, an MRI machine effectively measures the Fourier transform of a signal x representing the object being scanned, and the reconstruction problem is exactly the problem of inverting the Fourier transform $\widehat{x}$ of x approximately given a set of measurements. Minimizing the sample complexity of acquiring a signal using Fourier measurements thus translates directly to reduction in the time the patient spends in the MRI machine [LDSP08] while a scan is being taken. In applications to Computed Tomography (CT) reduction in measurement cost leads to reduction in the radiation dose that a patient receives [Sid11]. Because of this strong practical motivation, the problem of computing a good approximation to the FFT of a Fourier sparse signal fast and using few measurements in time domain has been the subject of much attention several communities. In the area of *compressive sensing* [Don06, CT06], where one studies the task of recovering (approximately) sparse signals from linear measurements, Fourier measurements have been one of the key settings of interest. In particular, the seminal work of [CT06, RV08] has shown that length N signals with at most k nonzero Fourier coefficients can be recovered using only $k \log^{O(1)} N$ samples in time domain. The recovery algorithms are based on linear programming and run in time polynomial in N. A different line of research on the *Sparse Fourier Transform* (Sparse FFT), initiated in the fields of computational complexity and learning theory, has been focused on developing algorithms whose sample complexity *and* running time scale with the sparsity as opposed to the length of the input signal. Many such algorithms have been proposed in the literature, including [GL89, KM91, Man92, GGI+02, AGS03, GMS05, Iwe10, Aka10, HIKP12b, HIKP12a, BCG+12, HAKI12, PR13, HKPV13, IKP14]. These works show that, for a wide range of signals, both the time complexity and the number of signal samples taken can be significantly sub-linear in N, often of the form $k \log^{O(1)} N$.

*A full version of this paper is available as an arxiv report at http://arxiv.org/abs/1604.00845

STOC'16, June 19–21, 2016, Cambridge, MA, USA
© 2016 ACM. 978-1-4503-4132-5/16/06...$15.00
http://dx.doi.org/10.1145/2897518.2897650

In this paper we consider the problem of computing a sparse approximation to a signal $x \in \mathbb{C}^N$ given access to its Fourier transform $\widehat{x} \in \mathbb{C}^N$.[1] The best known results obtained in both compressive sensing literature and sparse FFT literature on this problem are summarized in Fig. 1. We focus on algorithms that work for worst-case signals and recover k-sparse approximations satisfying the so-called ℓ_2/ℓ_2 approximation guarantee. In this case, the goal of an algorithm is as follows: given m samples of the Fourier transform $\widehat{x}$ of a signal x, and the sparsity parameter k, output x' satisfying

$$\|x - x'\|_2 \leq C \min_{k\text{-sparse } y} \|x - y\|_2, \quad (1)$$

The algorithms are randomized[2] and succeed with at least constant probability.

Higher dimensional Fourier transform. While significant attention in the sublinear Sparse FFT literature has been devoted to the basic case of Fourier transform on the line (i.e. one-dimensional signals), the sparsest signals often occur in applications involving higher-dimensional DFTs. Although a reduction from DFT on a two-dimensional grid *with relatively prime side lengths* $p \times q$ to a one-dimensional DFT of length pq is possible [GMS05, Iwe12]), the reduction does not apply to the most common case when the side lengths of the grid are equal to the same powers of two. It turns out that most sublinear Sparse FFT techniques developed for the one-dimensional DFT do not extend well to the higher dimensional setting, suffering from *at least a polylogaritmic loss in sample complexity.* Specifically, the only prior sublinear time algorithm that applies to general $m \times m$ grids is due to to [GMS05], has $O(k \log^c N)$ sample and time complexity for a rather large value of c. If N is a power of 2, a two-dimensional adaptation of the [HIKP12a] algorithm (outlined in [GHI+13]) has roughly $O(k \log^3 N)$ time and sample complexity, and an adaptation of [IKP14] has $O(k \log^2 N (\log \log N)^{O(1)})$ sample complexity. In general dimension $d \geq 1$ both of these algorithms have sample complexity $\Omega(k \log^d N)$.

Thus, none of the results obtained so far was able to guarantee sparse recovery from high dimensional (any $d \geq 2$) Fourier measurements without suffering at least a polylogarithmic loss in sample complexity, while at the same time achieving sublinear runtime.

Our results. In this paper we give an algorithm that achieves the ℓ_2/ℓ_2 sparse recovery guarantees (1) with d-dimensional Fourier measurements using $O_d(k \log N \log \log N)$ samples of the signal and and runtime $O_d(k \log^{d+3} N)$. This is the first sublinear time algorithm that comes within a poly$(\log \log N)$ factor of the sample complexity lower bound of $\Omega(k \log(N/k))$ due to [DIPW10] for any dimension higher than one.

Sparse Fourier Transform overview. The overall outline of our algorithm follows the framework of [GMS05], [HIKP12a, IKP14], which adapt the methods of [CCFC02, GLPS10] from arbitrary linear measurements to Fourier measurements. The idea is to take, multiple times, a set of $B = O(k)$ linear measurements of the form

$$\tilde{u}_j = \sum_{i:h(i)=j} s_i x_i$$

for random hash functions $h : [N] \to [B]$ and random sign changes s_i with $|s_i| = 1$. This corresponds to *hashing* to B *buckets.* With such ideal linear measurements, $O(\log(N/k))$ hashes suffice for sparse recovery, giving an $O(k \log(N/k))$ sample complexity.

The sparse Fourier transform algorithms approximate $\tilde{u}$ using linear combinations of Fourier samples. Specifically, the coefficients of x are first permuted via a random affine permutation of the input space. Then the coefficients are partitioned into buckets. This step uses the"filtering" process that approximately partitions the range of x into intervals (or, in higher dimension, squares, or ℓ_∞ balls) with N/B coefficients each, where each interval corresponds to one bucket. Overall, this ensures that most of the large coefficients are "isolated", i.e., are hashed to unique buckets, as well as that the contributions from the "tail" of the signal x to those buckets is not much greater than the average (the tail of the signal defined as $\mathrm{Err}_k(x) = \min_{k-\text{sparse } y} ||x - y||_2$). This enables the algorithm to identify the locations of the dominant coefficients and estimate their values, producing a sparse estimate χ of x. To improve this estimate, we repeat the process on $x - \chi$ by subtracting the influence of χ during hashing, thereby *refining* the approximation of x constructed. After a few iterations of this refinement process the algorithm obtains a good sparse approximation χ of x.

A major hurdle in implementing this strategy is that any filter that has been constructed in the literature so far is imprecise in that coefficients contribute ("leak"') to buckets other than the one they are technically mapped into. This contribution, however, is limited and can be controlled by the quality of the filter. The details of filter choice have played a crucial role in recent developments in Sparse FFT algorithms. For example, the best known runtime for one-dimensional Sparse FFT, due to [HIKP12b], was obtained by constructing filters that (almost) precisely mimic the ideal hash process, allowing for a very fast implementation of the process in dimension one. The price to pay for the precision of the filter, however, is that each hashing becomes a $\log^d N$ factor more costly in terms of sample complexity and runtime than in the idealized case. At the other extreme, the algorithm of [GMS05] uses much less precise filters, which only lead to a C^d loss of sample complexity in higher dimensions d, for a constant $C > 0$. Unfortunately, because of the imprecision of the filters the iterative improvement process becomes quite noisy, requiring $\Omega(\log N)$ iterations of the refinement process above. As [GMS05] use fresh randomness for each such iteration, this results in an $\Omega(\log N)$ factor loss in sample complexity. The result of [IKP14] uses a hybrid strategy, effectively interpolating between [HIKP12b] and [GMS05]. This gives the near optimal $O(k \log N \log^{O(1)} \log N)$ sample complexity in dimension one (i.e. Fourier transform on the line), but still suffers from a $\log^{d-1} N$ loss in dimension d.

Techniques of [IK14]. The first algorithm to achieve optimal sample complexity was recently introduced in [IK14]. The algorithms uses an approach inspired by [GMS05] (and

[1]Note that the problem of reconstructing a signal from Fourier measurements is equivalent to the problem of computing the Fourier transform of a signal x whose spectrum is approximately sparse, as the DFT and its inverse are only different by a conjugation.

[2]Some of the algorithms [CT06, RV08, CGV12] can in fact be made deterministic, but at the cost of satisfying a somewhat weaker ℓ_2/ℓ_1 guarantee.

Reference	Time	Samples	C	Dimension $d > 1$?
[CT06, RV08, CGV12, Bou14, HR16]	$N \times m$ linear program	$O(k \log^2(k) \log(N))$	$O(1)$	yes
[CP10]	$N \times m$ linear program	$O(k \log N)$	$(\log N)^{O(1)}$	yes
[HIKP12a]	$O(k \log(N) \log(N/k))$	$O(k \log(N) \log(N/k))$	any	no
[IKP14]	$k \log^2(N) \log^{O(1)} \log N$	$k \log(N) \log^{O(1)} \log N$	any	no
[IK14]	$N \log^{O(1)} N$	$O(k \log N)$	any	yes
[DIPW10]		$\Omega(k \log(N/k))$	O(1)	lower bound

Figure 1: Bounds for the algorithms that recover k-sparse Fourier approximations. All algorithms produce an output satisfying Equation 1 with probability of success that is at least constant. The forth column specifies constraints on approximation factor C. For example, $C = O(1)$ means that the algorithm can only handle constant C as opposed to any $C > 1$. The last column specifies whether the sample complexity bounds are unchanged, up to factors that depend on dimension d only, for higher dimensional DFT.

hence uses 'crude' filters that do not lose much in sample complexity), but introduces a key innovation enabling optimal sample complexity: the algorithm does *not* use fresh hash functions in every repetition of the refinement process. Instead, $O(\log N)$ hash functions are chosen at the beginning of the process, such that each large coefficient is isolated by most of those functions with high probability. The same hash functions are then used throughout the execution of the algorithm. As every hash function required a separate set of samples to construct the buckets, reusing the hash functions makes sample complexity *independent of the number of iterations*, leading to the optimal bound.

While a natural idea, reusing hash functions creates a major difficulty: if the algorithm identified a non-existent large coefficient (i.e. a false positive) by mistake and added it to χ, this coefficient would be present in the difference vector $x - \chi$ (i.e. residual signal) and would need to be corrected later. As the spurious coefficient depends on the measurements, the 'isolation' properties required for recovery need not hold for it as its position is determined by the hash functions themselves, and the algorithm might not be able to correct the mistake. This hurdle was overcome in [IK14] by ensuring that no large coefficients are created spuriously throughout the execution process. This is a nontrivial property to achieve, as the hashing process is quite noisy due to use of the 'crude' filters to reduce the number of samples (because the filters are quite simple, the bucketing process suffers from substantial leakage). The solution was to recover the large coefficients in decreasing order of their magnitude. Specifically, in each step, the algorithm recovered coefficients with magnitude that exceeded a specific threshold (that decreases at an exponential rate). With this approach the ℓ_∞ norm of the residual signal decreases by a constant factor in every round, resulting in the even stronger ℓ_∞/ℓ_2 sparse recovery guarantees in the end. The price to pay for this strong guarantee was the need for a very strong primitive for locating dominant elements in the residual signal: a primitive was needed that would make mistakes with at most inverse polynomial probability. This was achieved by essentially brute-force decoding over all potential elements in $[N]$: the algorithm loops over all elements $i \in [N]$ and for each i tests, using the $O(\log N)$ measurements taken, whether i is a dominant element in the residual signal. This resulted in $\Omega(N)$ runtime.

Our techniques. In this paper we show how to make the aforementioned algorithm run in sub-linear time, at the price of a slightly increased sampling complexity of $O_d(k \log N \log \log N)$. To achieve a sub-linear runtime, we need to replace the loop over all N coefficients by a location primitive (similar to that in prior works) that identifies the position of any large coefficient that is isolated in a bucket in $\log^{O(1)} N$ time per bucket, i.e. without resorting to brute force enumeration over the domain of size N. Unfortunately, the identification step alone increases the sampling complexity by $O(\log N)$ per hash function, so unlike [IK14], here we cannot repeat this process using $O(\log N)$ hash functions to ensure that each large coefficient is isolated by one of those functions. Instead, we can only afford $O(\log \log N)$ hash functions overall, which means that $1/\log^{O(1)} N$ fraction of large coefficients will not be isolated in most hashings. This immediately precludes the possibility of using the initial samples to achieve ℓ_∞ norm reduction as in [IK14]. Another problem, however, is that the weaker location primitive that we use may generate *spurious coefficients* at every step of the recovery process. These spurious coefficients, together with the $1/\log^{O(1)} N$ fraction of non-isolated elements, contaminate the recovery process and essentially render the original samples useless after a small number of refinement steps. To overcome these hurdles, instead of the ℓ_∞ reduction process of [IK14] we use a weaker invariant on the reduction of mass in the 'heavy' elements of the signal throughout our iterative process. Specifically, instead of reduction of ℓ_∞ norm of the residual as in [IK14] we give a procedure for reducing the ℓ_1 norm of the 'head' of the signal. To overcome the contamination coming from non-isolated as well as spuriously created coefficients, we achieve ℓ_1 norm reduction by alternating two procedures. The first procedure uses the $O(\log \log N)$ hash functions to reduce the ℓ_1 norm of 'well-hashed' elements in the signal, and the second uses a simple sparse recovery primitive to reduce the ℓ_∞ norm of offending coefficients when the first procedure gets stuck. This can be viewed as a signal-to-noise ratio (SNR) reduction step similar in spirit the one achieved in [IKP14]. The SNR reduction phase is insufficient for achieving the ℓ_2/ℓ_2 sparse recovery guarantee, and hence we need to run a cleanup phase at the end, when the signal to noise ratio is constant. It has been observed before (in [IKP14]) that if the signal to noise ratio is constant, then recovery can be done using standard techniques with optimal sample complexity. The crucial difference between [IKP14] and our setting is, however, that we only have

bounds on ℓ_1-SNR as opposed to ℓ_2-SNR In [IKP14]. It turns out, however, that this is not a problem – we give a stronger analysis of the corresponding primitive from [IKP14], showing that ℓ_1-SNR bound is sufficient.

Related work on continuous Sparse FFT. Recently [BCG+12] and [PS15] gave algorithms for the related problem of computing Sparse FFT in the continuous setting. These results are not directly comparable to ours, and suffer from a polylogarithmic inefficiency in sample complexity bounds.

Organization. The rest of the paper is organized as follows. Definitions, notation and some basic claims are presented in section 2. We present the algorithm and give an overview of the analysis in section 3. In section 4 we set up notation for analysis of our signal location primitive on reused measurements, and in section 5 prove the bounds on ℓ_1 norm of elements that are not discovered by an invocation of LocateSignal – these bounds form the basis of our main result. Then section 6 uses these results to prove efficiency of the ℓ_1-SNR reduction loop in our Sparse FFT algorithm and proves correctness of the algorithm itself.

2. PRELIMINARIES

For a positive even integer a we will use the notation $[a] = \{-\frac{a}{2}, -\frac{a}{2}+1, \ldots, -1, 0, 1, \ldots, \frac{a}{2}-1\}$. We will consider signals of length $N = n^d$, where n is a power of 2 and $d \geq 1$ is the dimension. We use the notation $\omega = e^{2\pi i/n}$ for the root of unity of order n. The d-dimensional forward and inverse Fourier transforms are given by

$$\hat{x}_j = \frac{1}{\sqrt{N}} \sum_{i \in [n]^d} \omega^{-i^T j} x_i \quad \text{and} \quad x_j = \frac{1}{\sqrt{N}} \sum_{i \in [n]^d} \omega^{i^T j} \hat{x}_i \quad (2)$$

respectively, where $j \in [n]^d$. We will denote the forward Fourier transform by $\mathcal{F}$. Note that we use the orthonormal version of the Fourier transform. Thus, we have $||\hat{x}||_2 = ||x||_2$ for all $x \in \mathbb{C}^N$ (Parseval's identity). Given access to samples of $\widehat{x}$, we recover a signal z such that

$$||x - z||_2 \leq (1+\epsilon) \min_{k-\text{ sparse } y} ||x - y||_2$$

We will use pseudorandom spectrum permutations, which we now define. We write $\mathcal{M}_{d\times d}$ for the set of $d \times d$ matrices over $\mathbb{Z}_n$ with odd determinant. For $\Sigma \in \mathcal{M}_{d\times d}, q \in [n]^d$ and $i \in [n]^d$ let $\pi_{\Sigma,q}(i) = \Sigma(i-q) \mod n$. Since $\Sigma \in \mathcal{M}_{d\times d}$, this is a permutation. Our algorithm will use π to hash heavy hitters into B buckets, where we will choose $B \approx k$, and will always have $B = b^d$, where b is a power of 2. We will often omit the subscript Σ, q and simply write $\pi(i)$ when Σ, q is fixed or clear from context. For $i \in [n]^d$ we let $h_\pi(i) := \text{round}(\frac{b}{n}\pi(i))$, and omit the subscript π when the permutation π is fixed or clear from context. Note that h hashes $[n]^d$ to $[b]^d$. For $i, j \in [n]^d$ we let $o_i(j) = \pi(j) - (n/b)h(i)$ be the "offset" of $j \in [n]^d$ relative to $i \in [n]^d$ (note that this definition is different from the one in [IK14]).

Definition 2.1. *Suppose that Σ^{-1} exists* mod n. *For a, $q \in [n]^d$ define permutation $P_{\Sigma,a,q}$ by $(P_{\Sigma,a,q}\hat{x})_i = \hat{x}_{\Sigma^T(i-a)} \cdot \omega^{i^T \Sigma q}$.*

Lemma 2.2. $\mathcal{F}^{-1}(P_{\Sigma,a,q}\hat{x})_{\pi_{\Sigma,q}(i)} = x_i \omega^{a^T \Sigma i}$

The proof is given in [IK14] and we do not repeat it here. Define

$$\text{Err}_k(x) = \min_{k-\text{sparse } y} ||x - y||_2 \quad \text{and} \quad \mu^2 = \text{Err}_k^2(x)/k. \quad (3)$$

In this paper, we assume knowledge of μ (a constant factor upper bound on μ suffices). We assume that elements of x are polynomially bounded and have polynomial precision. In particular, we assume that the signal to noise ratio is bounded by a polynomial, namely that $R^* \geq ||x||_\infty/\mu = N^{O(1)}$. For a signal $x \in \mathbb{C}^{[n]^d}$ and integer $k \geq 1$ we let $x_{[k]}$ denote the top k elements of x. We use the notation $\mathbb{B}_r^\infty(x)$ to denote the ℓ_∞ ball of radius r around x: $\mathbb{B}_r^\infty(x) = \{y \in [n]^d : ||x-y||_\infty \leq r\}$, where $||x-y||_\infty = \max_{s\in d} ||x_s - y_s||_\circ$, and $||x_s - y_s||_\circ$ is the circular distance on $\mathbb{Z}_n$. We will also use the notation $f \lesssim g$ to denote $f = O(g)$. For a real number a we write $|a|_+$ to denote the positive part of a, i.e. $|a|_+ = a$ if $a \geq 0$ and $|a|_+ = 0$ otherwise.

We will use the filter $G, \hat{G}$ constructed in [IK14]. The filter is defined by a parameter $F \geq 1$ that governs its decay properties. The filter satisfies $\text{supp}\,\hat{G} \subseteq [-F \cdot b, F \cdot b]^d$ and

Lemma 2.3 (Lemma 3.1 in [IK14]). *One has* **(1)** $G_j \in [\frac{1}{(2\pi)^{F\cdot d}}, 1]$ *for all* $j \in [n]^d$ *such that* $||j||_\infty \leq \frac{n}{2b}$ *and* **(2)** $|G_j| \leq \left(\frac{2}{1+(b/n)||j||_\infty}\right)^F$ *for all* $j \in [n]^d$ *as long as* $b \geq 3$ *and* **(3)** $G_j \in [0,1]$ *for all* j *as long as* F *is even.*

We note that property **(3)** was not stated in Lemma 3.1 of [IK14], but follows directly from their construction. The properties above imply that most of the mass of the filter is concentrated in a square of side $O(n/b)$, approximating the "ideal" filter (whose value would be equal to 1 for entries within the square and equal to 0 outside of it). Note that for each $i \in [n]^d$ one has $|G_{o_i(i)}| \geq \frac{1}{(2\pi)^{d\cdot F}}$. We refer to the parameter F as the *sharpness* of the filter. Our hash functions are not pairwise independent, but possess a property that still makes hashing using our filters efficient:

Lemma 2.4 (Lemma 3.2 in [IK14]). *Let $i, j \in [n]^d$. Let Σ be uniformly random with odd determinant. Then for all $t \geq 0$ one has* $\Pr[||\Sigma(i-j)||_\infty \leq t] \leq 2(2t/n)^d$.

Pseudorandom spectrum permutations combined with a filter G give us the ability to 'hash' the elements of the input signal into a number of buckets (denoted by B). We formalize this using the notion of a *hashing*. A hashing is a tuple consisting of a pseudorandom spectrum permutation π, target number of buckets B and a sharpness parameter F of our filter, denoted by $H = (\pi, B, F)$. A hashing H is a function that maps a signal x to B signals, each corresponding to a hash bucket, allowing us to solve the k-sparse recovery problem on input x by reducing it to 1-sparse recovery problems on the bucketed signals. Formally,

Definition 2.5 (Hashing $H = (\pi, B, F)$). *For a permutation $\pi = (\Sigma, q)$, parameters $b > 1$, $B = b^d$ and F, a* hashing $H := (\pi, B, F)$ *is a function mapping a signal $x \in \mathbb{C}^{[n]^d}$ to B signals $H(x) = (u_s)_{s\in[b]^d}$, where $u_s \in \mathbb{C}^{[n]^d}$ for each $s \in [b]^d$, such that for each $i \in [n]^d$*

$$u_{s,i} = \sum_{j \in [n]^d} G_{\pi(j) - (n/b)\cdot s} x_j \omega^{i^T \Sigma j},$$

where G is the filter with B buckets and sharpness F constructed in Lemma 2.3.

For a hashing $H = (\pi, B, F), \pi = (\Sigma, q)$ we sometimes write $P_{H,a}, a \in [n]^d$ to denote $P_{\Sigma,a,q}$.

DEFINITION 2.6 (MEASUREMENT $m = m(x, H, a)$). *For a signal $x \in \mathbb{C}^{[n]^d}$, a hashing $H = (\pi, B, F)$ and a parameter $a \in [n]^d$, a* measurement $m = m(x, H, a) \in \mathbb{C}^{[b]^d}$ *is the B-dimensional complex valued vector of evaluations of a hashing $H(x)$ at $a \in \mathbb{C}^{[n]^d}$. The vector is indexed by $s \in [b]^d$, and we have $m_s = \sum_{j \in [n]^d} G_{\pi(j)-(n/b)\cdot s} x_j \omega^{a^T \Sigma j}$, where G is the filter with B buckets and sharpness F constructed in Lemma 2.3.*

DEFINITION 2.7. *For any $x \in \mathbb{C}^{[n]^d}$ and any hashing $H = (\pi, B, G)$ define the vector $\mu^2_{H,\cdot}(x) \in \mathbb{R}^{[n]^d}$ by letting for every $i \in [n]^d$ $\mu^2_{H,i}(x) := |G^{-2}_{o_i(i)}| \sum_{j \in [n]^d \setminus \{i\}} |x_j|^2 |G_{o_i(j)}|^2$.*

We access the signal x in Fourier domain via the function HASHTOBINS$(\hat{x}, \chi, (H, a))$, which evaluates the hashing H of residual signal $x - \chi$ at point $a \in [n]^d$, i.e. computes the measurement $m(x, H, a)$ (the computation is done with polynomial precision). One can view this function as "hashing" the residual $x - \chi$ into B bins by convolving it with the filter G constructed above and sampling at a set of equispaced points. This function is rather standard in Sparse FFT literature. A call to HASHTOBINS$(\hat{x}, \chi, (H, a))$ computes a vector $y' = \hat{G} \cdot P_{\Sigma,a,q}(\hat{x} - \hat{\chi})$ with polynomial precision using semi-equispaced FFT to subtract $\widehat{\chi}$ from $\widehat{x}$ in frequency domain (see the full version [Kap16] for more details), then computes $u_j = \sqrt{N} \mathcal{F}^{-1}(y')_{(n/b)\cdot j}$ for $j \in [b]^d$ and returns the $B = b^d$-dimensional vector u. We will use the following properties of HASHTOBINS:

LEMMA 2.8. *There exists a constant $C > 0$ such that for any dimension $d \geq 1$, any integer $B \geq 1$, any $x, \chi \in \mathbb{C}^{[n]^d}, x' := x - \chi$, if $\Sigma \in \mathcal{M}_{d\times d}, q \in [n]^d$ are selected uniformly at random, the following conditions hold.*

Let $\pi = (\Sigma, q)$, $H = (\pi, B, G)$, where G is the filter with B buckets and sharpness F constructed in Lemma 2.3, and let $u =$ HASHTOBINS$(\hat{x}, \chi, (H, a))$. If $F \geq 2d$, then for any $i \in [n]^d$

(1) *For any H one has $\max_{a \in [n]^d} |G^{-1}_{o_i(i)} \omega^{-a^T \Sigma i} u_{h(i)} - x'_i| \leq G^{-1}_{o_i(i)} \cdot \sum_{j \in [n]^d \setminus \{i\}} G_{o_i(j)} |x'_j|$. Furthermore, $\mathbf{E}_H[G^{-1}_{o_i(i)} \cdot \sum_{j \in [n]^d \setminus \{i\}} G_{o_i(j)} |x'_j|] \leq (2\pi)^{d\cdot F} \cdot C^d ||x'||_1 / B + N^{-\Omega(c)}$;*

(2) *$\mathbf{E}_H[\mu^2_{H,i}(x')] \leq (2\pi)^{2d\cdot F} \cdot C^d \|x'\|_2^2 / B$.*

Furthermore,

(3) *for any hashing H, if a is chosen uniformly at random from $[n]^d$, one has*

$$\mathbf{E}_a[|G^{-1}_{o_i(i)} \omega^{-a^T \Sigma i} u_{h(i)} - x'_i|^2] \leq \mu^2_{H,i}(x') + N^{-\Omega(c)}.$$

Here $c > 0$ is an absolute constant that can be chosen arbitrarily large at the expense of increasing runtime by a $c^{O(d)}$ factor.

The proof of Lemma 2.8 is quite standard and is deferred to the full version of the paper.

We will need several definitions and lemmas from [IK14], which we state here. We sometimes need slight modifications of the corresponding statements from [IK14], in which case we provide proofs in the full version of the paper [Kap16]. Throughout this paper the main object of our analysis is a properly defined set $S \subseteq [n]^d$ that contains the 'large' coefficients of the input vector x. Below we state our definitions and auxiliary lemmas without specifying the identity of this set, and then use specific instantiations of S in the analysis of outer primitives such as REDUCEL1NORM, REDUCEINFNORM and RECOVERATCONSTANTSNR (see Algorithm 2). Specific instantiations of S will always have $|S| \approx k$.

We will need the definition of an element $i \in [n]^d$ being isolated under a hashing $H = (\pi, B, F)$ with respect to a set S. Intuitively, an element $i \in S$ is isolated under hashing H with respect to set S if not too many other elements of S are hashed too close to i. Formally, we have

DEFINITION 2.9 (ISOLATED ELEMENT). *Let $H = (\pi, B, F)$, where $\pi = (\Sigma, q)$, $\Sigma \in \mathcal{M}_{d\times d}, q \in [n]^d$. We say that an element $i \in [n]^d$ is* isolated *under hashing H* at scale t *if*

$$|\pi(S \setminus \{i\}) \cap \mathbb{B}^\infty_{(n/b)\cdot h(i)}((n/b) \cdot 2^t)| \leq (2\pi)^{-d\cdot F} \cdot \alpha^{d/2} 2^{(t+1)d} \cdot 2^t.$$

We say that i is simply isolated *under hashing H if it is isolated under H at all scales $t \geq 0$.*

Any element $i \in [n]^d$ is likely to be isolated under a random hashing H:

LEMMA 2.10. *For any integer $k \geq 1$ and any $S \subseteq [n]^d$, $|S| \leq 2k$, if $B \geq (2\pi)^{4d\cdot F} \cdot k/\alpha^d$ for $\alpha \in (0, 1)$ smaller than an absolute constant, $F \geq 2d$, and a hashing $H = (\pi, B, F)$ is chosen randomly (i.e. $\Sigma \in \mathcal{M}_{d\times d}, q \in [n]^d$ are chosen uniformly at random, and $\pi = (\Sigma, q)$), then each $i \in [n]^d$ is* isolated *under permutation π with probability at least $1 - \frac{1}{2}\sqrt{\alpha}$.*

The proof of the lemma is very similar to Lemma 5.4 in [IK14] (the only difference is that the ℓ_∞ ball is centered at the point that i hashes to in Lemma 2.10, whereas it was centered at $\pi(i)$ in Lemma 5.4 of [IK14]) and is given in the full version [Kap16] for completeness.

As every element $i \in S$ is likely to be isolated under one random hashing, it is very likely to be isolated under a large fraction of hashings $H_1, \ldots, H_{r_{max}}$ (by Chernoff bounds):

LEMMA 2.11. *For any integer $k \geq 1$, and any $S \subseteq [n]^d$, $|S| \leq 2k$, if $B \geq (2\pi)^{4d\cdot F} \cdot k/\alpha^d$ for $\alpha \in (0, 1)$ smaller than an absolute constant, $F \geq 2d$, $H_r = (\pi_r, B, F)$, $r = 1, \ldots, r_{max}$ a sequence of random hashings, then every $i \in [n]^d$ is isolated with respect to S under at least $(1 - \sqrt{\alpha}) r_{max}$ hashings $H_r, r = 1, \ldots, r_{max}$ with probability at least $1 - 2^{-\Omega(\sqrt{\alpha} r_{max})}$.*

Our sparse FFT algorithm uses a simple decoding procedure for identifying elements of the signal x that fall into each 'hash bucket'. In order to minimize sample complexity this procedure identifies an element $i \in [n]^d$ that falls into a hash bucket in $O(d \log n / \log\log n)$ iterations, where we first loop over all d coordinates, and in each such loop perform $O(\log N / \log\log N)$ iterations, with each iteration recovering a block of $O(\log\log N)$ bits of i (see section 4). The following notation helps simplify presentation of our location primitive.

Operations on vectors in $[n]^d$. For a pair of vectors $(\alpha_1, \beta_1), (\alpha_2, \beta_2) \in [n]^d \times [n]^d$ we let $(\alpha_1, \beta_1) \star (\alpha_2, \beta_2)$ denote the vector $\gamma \in [n]^d$ such that $\gamma_i = (\alpha_1)_i \cdot (\alpha_2)_i + (\beta_1)_i \cdot$

$(\beta_2)_i$ for all $i \in [d]$. Note that for any $a, b, c \in [n]^d \times [n]^d$ one has $a \star b + a \star c = a \star (b + c)$, where addition for elements of $[n]^d \times [n]^d$ is componentwise. We write $\mathbf{1} \in [n]^d$ for the all ones vector in dimension d, and $\mathbf{0} \in [n]^d$ for the zero vector. For a set $\mathcal{A} \subseteq [n]^d \times [n]^d$ and a vector $(\alpha, \beta) \in [n]^d \times [n]^d$ we denote $\mathcal{A} \star (\alpha, \beta) := \{a \star (\alpha, \beta) : a \in \mathcal{A}\}$.

Correctness of our location algorithm relies on a simple pseudorandomness property of the set of locations that we access the signal at. Specifically, we introduce

DEFINITION 2.12 (BALANCED SET OF POINTS). *For an integer $\Delta \geq 2$ we say that a (multi)set $\mathcal{Z} \subseteq [n]^d$ is* Δ-balanced *in coordinate $s \in [1:d]$ if for every $r = 1, \ldots, \Delta - 1$ at least 49/100 fraction of elements in the set $\{\omega_\Delta^{r \cdot z_s}\}_{z \in \mathcal{Z}}$ belong to the left halfplane $\{u \in \mathbb{C} : Re(u) \leq 0\}$ in the complex plane, where $\omega_\Delta = e^{2\pi i/\Delta}$ is the Δ-th root of unity.*

Note that if Δ divides n, then for any fixed value of $r \neq 0$ the point $\omega_\Delta^{r \cdot z_s}$ is uniformly distributed over the Δ'-th roots of unity for $\Delta' < \Delta$ when z_s is uniformly random in $[n]$. Thus for $r \neq 0$ we expect at least half the points to lie in the halfplane $\{u \in \mathbb{C} : \mathrm{Re}(u) \leq 0\}$. A set $\mathcal{Z}$ is balanced if it does not deviate from expected behavior too much. The following claim is immediate via standard concentration bounds:

CLAIM 2.13. *There exists a constant $C > 0$ such that for any Δ a power of two, $\Delta = \log^{O(1)} n$, and n a power of 2 the following holds if $\Delta < n$. If elements of a (multi)set $\mathcal{A} \subseteq [n]^d \times [n]^d$ of size $C \log \log N$ are chosen uniformly at random with replacement from $[n]^d \times [n]^d$, then with probability at least $1 - 1/\log^4 N$ one has that for every $s \in [1:d]$ the set $\mathcal{A} \star (\mathbf{0}, \mathbf{e}_s)$ is Δ-balanced in coordinate s.*

Since we only use one value of Δ in the paper (see line 8 in Algorithm 1), we will usually say that a set is simply 'balanced' to denote the Δ-balanced property for this value of Δ.

3. THE ALGORITHM AND OVERVIEW OF ITS ANALYSIS

In this section we first describe the main components of our algorithm and give an outline of the analysis. The details of the analysis are presented in subsequent sections, with some proofs deferred to the full version of the paper due to space constraints. Our Sparse FFT algorithm (specified formally as Algorithm 2) proceeds as follows.

Measuring $\widehat{x}$. The algorithm starts by taking measurements of the signal in lines 11-19. Note that the algorithm selects $O(\log \log N)$ hashings $H_r = (\pi_r, B, F), r = 1, \ldots, O(\log \log N)$, where π_r are selected uniformly at random, and for each r selects a set $\mathcal{A}_r \subseteq [n]^d \times [n]^d$ of size $O(\log \log N)$ that determines locations to access in frequency domain. The signal $\widehat{x}$ is accessed via HASHTOBINS (see Lemma 2.8 above). The function HASHTOBINS accesses filtered versions of $\widehat{x}$ shifted by elements of a randomly selected set (the number of shifts is $O(\log N / \log \log N)$). These shifts are useful for locating 'heavy' elements from the output of HASHTOBINS. Note that since each hashing takes $O(B) = O(k)$ samples, the total sample complexity of the measurement step is $O(k \log N \log \log N)$. This is the dominant contribution to sample complexity, but it is not the only one. The other contribution of $O(k \log N \log \log N)$ comes from invocations of ESTIMATEVALUES from our ℓ_1-SNR reduction loop (see below). The loop goes over $O(\log R^*) = O(\log N)$ iterations,

Algorithm 1 Location primitive: given a set of measurements taken with a single hashing, returns a list of elements in $[n]^d$, at most one per each hash bucket

1: **procedure** LOCATESIGNAL($\chi, H, \{m(\widehat{x}, H, a \star (\mathbf{1}, \mathbf{w})\}_{a \in \mathcal{A}, \mathbf{w} \in \mathcal{W}}$)
2: Let $x' := x - \chi$. Compute $\{m(\widehat{x'}, H, a \star (\mathbf{1}, \mathbf{w})\}_{a \in \mathcal{A}, \mathbf{w} \in \mathcal{W}}$ using semi-equispaced FFT.
3: $L \leftarrow \emptyset$
4: **for** $j \in [b]^d$ **do** ▷ Loop over all hash buckets
5: $\mathbf{f} \leftarrow \mathbf{0}^d$
6: **for** $s = 1$ to d **do**
7: ▷ Recovering each of d coordinates separately
8: $\Delta \leftarrow 2^{\lfloor \frac{1}{2} \log_2 \log_2 n \rfloor}$
9: **for** $g = 1$ to $\log_\Delta n$ **do**
10: $\mathbf{w} \leftarrow n\Delta^{-g} \cdot \mathbf{e}_s$ ▷ Note that $\mathbf{w} \in \mathcal{W}$
11: **If** $\exists$ a unique $r \in [0 : \Delta - 1]$ s. t.
12: $\left| \omega_\Delta^{-r \cdot \beta_s} \omega^{-(n \cdot \Delta^{-g} \mathbf{f}_s) \cdot \beta_s} \frac{m_j(\widehat{x'}, H, a \star (\mathbf{1}, \mathbf{w}))}{m_j(\widehat{x'}, H, a \star (\mathbf{1}, \mathbf{0}))} - 1 \right|$
is less than 1/3 for at least 3/5 fraction of $a = (\alpha, \beta) \in \mathcal{A}$
13: **then** $\mathbf{f} \leftarrow \mathbf{f} + \Delta^{g-1} \cdot r \cdot \mathbf{e}_s$
14: **end for**
15: **end for**
16: $L \leftarrow L \cup \{\Sigma^{-1}\mathbf{f}\}$ ▷ Add element to output list
17: **end for**
18: **return** L
19: **end procedure**

and in each iteration ESTIMATEVALUES uses $O(\log \log N)$ fresh hash functions to keep the number of false positives and estimation error small. The location algorithm is Algorithm 1 (LOCATESIGNAL; to simplify presentation we assume that $\log_\Delta n$ is an integer). Our main tool for analyzing performance of LOCATESIGNAL is Theorem 3.1, stated below. Theorem 3.1 applies to the following setting. Fix a set $S \subseteq [n]^d$ and a set of hashings $H_1, \ldots, H_{r_{max}}$ that encode signal measurement patterns, and let $S^* \subseteq S$ denote the (typically very small) set of elements of S that are not isolated with respect to most of these hashings. Theorem 3.1 shows that for any signal x and partially recovered signal χ, if L denotes the output list of an invocation of LOCATESIGNAL on the pair (x, χ) with measurements given by $H_1, \ldots, H_{r_{max}}$ and a set of random shifts, then the ℓ_1 norm of elements of the residual $(x - \chi)_S$ that are not discovered by LOCATESIGNAL can be bounded by a function of the amount of ℓ_1 mass of the residual that fell outside of the 'good' set $S \setminus S^*$, plus the 'noise level' $\mu \geq ||x_{[n]^d \setminus S}||_\infty$ times k. If we think of applying Theorem 3.1 iteratively, we intuitively get that the fixed set of measurements given by hashings $H_1, \ldots, H_r$ allows us to always reduce the ℓ_1 norm of the residual $x' = x - \chi$ on the 'good' set $S \setminus S^*$ to about the amount of mass that is located outside of this good set(this is exactly how we use LOCATESIGNAL in our signal to noise ratio reduction loop below). In section 5 we prove

THEOREM 3.1. *For any constant $C' > 0$ there exist absolute constants $C_1, C_2, C_3 > 0$ such that for any $x, \chi \in \mathbb{C}^N$, $x' = x - \chi$, any integer $k \geq 1$ and any $S \subseteq [n]^d$ such that $||x_{[n]^d \setminus S}||_\infty \leq C'\mu$, where $\mu = ||x_{[n]^d \setminus [k]}||_2/\sqrt{k}$, the following conditions hold if $||x'||_\infty/\mu = N^{O(1)}$.*

Let $\pi_r = (\Sigma_r, q_r), r = 1, \ldots, r_{max}$ denote permutations, and let $H_r = (\pi_r, B, F)$, $F \geq 2d$, $F = \Theta(d)$, where $B \geq$

$(2\pi)^{4d\cdot F}k/\alpha^d$ *for* $\alpha \in (0,1)$ *smaller than a constant. Let* $S^* \subseteq S$ *denote the set of elements that are not isolated with respect to at least a* $\sqrt{\alpha}$ *fraction of hashings* $\{H_r\}$*. Then if additionally for every* $s \in [1:d]$ *the sets* $\mathcal{A}_r \star (\mathbf{0}, \mathbf{e}_s)$ *are balanced in coordinate* s *(as per Definition 2.12) for all* $r = 1, \ldots, r_{max}$*, and* $r_{max}, c_{max} \geq (C_1/\sqrt{\alpha}) \log\log N$*, then the list* $L := \bigcup_{r=1}^{r_{max}} L_r$*, where* $L_r \leftarrow$ LocateSignal $(\chi, k, \{m(\widehat{x}, H_r, a \star (\mathbf{1}, \mathbf{w}))\}_{r=1, a\in\mathcal{A}_r, \mathbf{w}\in\mathcal{W}}^{r_{max}})$ *satisfies* $||x'_{S\setminus S^*\setminus L}||_1 \leq (C_2\alpha)^{d/2}||x'_S||_1 + C_3^{d^2}(||\chi_{[n]^d\setminus S}||_1 + ||x'_{S^*}||_1) + 4\mu|S|$.

We note that the guarantees provided by Theorem 3.1 are deterministic, which lets us apply the theorem on reused measurements, i.e. when the signal χ may be correlated with the measurements themselves.

Reducing signal to noise ratio. Once the samples have been taken, the algorithm proceeds to the signal to noise (SNR) reduction loop (lines 20-27). The objective of this loop is to reduce the mass of the top (about k) elements in the residual signal to roughly the noise level $\mu \cdot k$ (once this is done, we run a 'cleanup' primitive, referred to as RecoverAtConstantSNR, to complete the recovery process – see below). Specifically, we define the set S of 'head elements' in the original signal x as

$$S = \{i \in [n]^d : |x_i| > \mu\}, \tag{4}$$

where $\mu^2 \geq \mathrm{Err}_k^2(x)/k$ upper bounds the average tail noise level. Note that we have $|S| \leq 2k$. Indeed, if $|S| > 2k$, more than k elements of S belong to the tail, amounting to more than $\mu^2 \cdot k \geq \mathrm{Err}_k^2(x)$ tail mass. Ideally, we would like this loop to construct and approximation $\chi^{(T)}$ to x *supported only on* S such that $||(x - \chi^{(T)})_S||_1 = O(\mu k)$, i.e. the ℓ_1-SNR of the residual signal on the set S of heavy elements is reduced to a constant. As some false positives will unfortunately occur throughout the execution of our algorithm due to the weaker sublinear time location and estimation primitives that we use, our SNR reduction loop is to construct an approximation $\chi^{(T)}$ to x with the somewhat weaker properties that

$$||(x - \chi^{(T)})_S||_1 + ||\chi^{(T)}||_{[n]^d\setminus S} = O(\mu k) \quad \text{and} \quad ||\chi^{(T)}||_0 \ll k. \tag{5}$$

Thus, we reduce the ℓ_1-SNR on the set S of 'head' elements to a constant, and at the same time not introduce too many spurious coefficients (i.e. false positives) outside S, and these coefficients do not contribute much ℓ_1 mass. The SNR reduction loop itself consists of repeated alternating invocations of two primitives, namely ReduceL1Norm and ReduceInfNorm. Of these two the former can be viewed as performing most of the reduction, and ReduceInfNorm is naturally viewed as performing a 'cleanup' phase to fix inefficiencies of ReduceL1Norm that are due to the small number of hash functions (only $O(\log\log N)$ as opposed to $O(\log N)$ in [IK14]) that we are allowed to use, as well as some mistakes that our sublinear runtime location and estimation primitives used in ReduceL1Norm might make.

ReduceL1Norm is presented as Algorithm 3 below. The algorithm performs $O(\log\log N)$ rounds of the following process: first, run LocateSignal on the current residual signal, then estimate values of the elements that belong to the list L output by LocateSignal, and **only keep those that are above a certain threshold** (see threshold $\frac{1}{10000}2^{-t}\nu + 4\mu$ in the call the EstimateValues in line 9 of Algorithm 3). This thresholding operation is crucial, and allows us to control the number of false positives. In fact, this is very similar to the approach of [IK14] of recovering elements starting from the largest. The only difference is that **(a)** our 'reliability threshold' is dictated by the ℓ_1 norm of the residual rather than the ℓ_∞ norm, as in [IK14], and **(b)** some false positives can still occur due to our weaker estimation primitives. Our main tool for formally stating the effect of ReduceL1Norm is Lemma 3.2 below.

Lemma 3.2. *For any* $x \in \mathbb{C}^N$*, any integer* $k \geq 1$*,* $B \geq (2\pi)^{4d\cdot F} \cdot k/\alpha^d$ *for* $\alpha \in (0,1]$ *smaller than an absolute constant and* $F \geq 2d, F = \Theta(d)$ *the following conditions hold for the set* $S := \{i \in [n]^d : |x_i| > \mu\}$*, where* $\mu^2 \geq ||x_{[n]^d\setminus[k]}||_2^2/k$*. Suppose that* $||x||_\infty/\mu = N^{O(1)}$*.*

For any sequence of hashings $H_r = (\pi_r, B, F)$*,* $r = 1, \ldots, r_{max}$*, if* $S^* \subseteq S$ *denotes the set of elements of* S *that are not isolated with respect to at least a* $\sqrt{\alpha}$ *fraction of the hashings* $H_r, r = 1, \ldots, r_{max}$*, then for any* $\chi \in \mathbb{C}^{[n]^d}$*,* $x' := x - \chi$*, if* $\nu \geq (\log^4 N)\mu$ *is a parameter such that*

A $||(x-\chi)_S||_1 \leq (\nu + 20\mu)k$*;*

B $||\chi_{[n]^d\setminus S}||_0 \leq \frac{1}{\log^{19} N}k$*;*

C $||(x-\chi)_{S^*}||_1 + ||\chi_{[n]^d\setminus S}||_1 \leq \frac{\nu}{\log^4 N}k$*,*

the following conditions hold.

If parameters r_{max}, c_{max} *are no at least* $(C_1/\sqrt{\alpha})\log\log N$*, where* C_1 *is the constant from Theorem 3.1 and measurements* m *are taken as in Algorithm 2, then the output* χ' *of the call* ReduceL1Norm $\left(\chi, k, \{m(\widehat{x}, H_r, a \star (\mathbf{1}, \mathbf{w}))\}_{r=1, a\in\mathcal{A}_r, \mathbf{w}\in\mathcal{W}}^{r_{max}}, 4\mu(\log^4 n)^{T-t}, \mu\right)$ *satisfies*

1. $||(x'-\chi')_S||_1 \leq \frac{1}{\log^4 N}\nu k + 20\mu k$ *(ℓ_1 norm of head elements is reduced by $\approx \log^4 N$ factor)*
2. $||(\chi+\chi')_{[n]^d\setminus S}||_0 \leq ||\chi_{[n]^d\setminus S}||_0 + \frac{1}{\log^{20} N}k$ *(few spurious coefficients are introduced)*
3. $||(x'-\chi')_{S^*}||_1 + ||(\chi+\chi')_{[n]^d\setminus S}||_1 \leq ||x'_{S^*}||_1 + ||\chi_{[n]^d\setminus S}||_1 + \frac{1}{\log^{20} N}\nu k$ *(ℓ_1 norm of spurious coefficients does not grow fast)*

with probability at least $1 - 1/\log^2 N$ *over the randomness used to take measurements* m *and by calls to* EstimateValues*. The number of samples used by calls to* EstimateValues *is bounded by* $2^{O(d^2)}k(\log\log N)^2$*, and the runtime is bounded by* $2^{O(d^2)}k\log^{d+2} N$*.*

The proof is deferred to the full version of the paper [Kap16]. As mentioned above, repeated calls to ReduceL1Norm do not allow us to reduce the ℓ_1 norm of the residual all the way down to noise level due to spurious elements: the process may get stuck when the spurious elements dominate the signal, as our measurements are not guaranteed to have isolation properties with respect to them. To remedy this, we periodically run a 'cleanup' primitive that reduces ℓ_∞ norm of the residual. The primitive is not very sample efficient, but that is not important since we run it using only $\approx k/\log^4 N$ buckets. Such a primitive is easy to obtain using standard techniques, and we provide one in the full version of the paper. Our primitive satisfies

LEMMA 3.3. *For any $x, \chi \in \mathbb{C}^n$, $x' = x - \chi$, any integer $k \geq 1$, if parameters ν and μ satisfy $\nu \geq ||x'_{[k]}||_1/k$, $\mu^2 \geq ||x'_{[n]^d \setminus [k]}||_2^2/k$, then the following conditions hold. If $S \subseteq [n]^d$ is the set of top k elements of x' in terms of absolute value, and $||x'_{[n]^d \setminus S}||_\infty \leq \nu$, then the output $\chi \in \mathbb{C}^{[n]^d}$ of a call to* REDUCEINFNORM$(\widehat{x}, \chi, k, \nu, \mu)$ *with probability at least $1 - N^{-10}$ over the randomness used in the call satisfies $||x' - \chi||_\infty \leq 8(\nu + \mu) + N^{-\Omega(c)}$, where the $N^{-\Omega(c)}$ term corresponds to polynomially small error from approximate computation of the semi-equispaced Fourier transform. Furthermore, we have $\chi_{[n]^d \setminus S} \equiv 0$. The number of samples used is bounded by $2^{O(d^2)} k \log^3 N$. The runtime is bounded by $2^{O(d^2)} k \log^{d+3} N$.*

Note that the lemma assumes bounds on the ℓ_1 norm of the 'head' elements and the ℓ_2 norm of the 'tail', and yields conclusions on the ℓ_∞ norm of the residual. Lemma 3.2 and Lemma 3.3 allow us to show that the SNR reduction loop in Algorithm 2 indeed achieves its cause, namely (5). Formally, we prove in section 6

THEOREM 3.4. *For any $x \in \mathbb{C}^N$, any integer $k \geq 1$, if $\mu^2 \geq \mathrm{Err}_k^2(x)/k$ and $R^* \geq ||x||_\infty/\mu = N^{O(1)}$, the following conditions hold for the set $S := \{i \in [n]^d : |x_i| > \mu\} \subseteq [n]^d$.*

The SNR reduction loop of Algorithm 2 (lines 20-27) returns $\chi^{(T)}$ such that

$$||(x - \chi^{(T)})_S||_1 \lesssim \mu k \quad \textit{(ℓ_1-SNR is constant)}$$

$$||\chi^{(T)}_{[n]^d \setminus S}||_1 \lesssim \mu k \quad \textit{(spurious elements have small ℓ_1 norm)}$$

$$||\chi^{(T)}_{[n]^d \setminus S}||_0 \lesssim \frac{1}{\log^{19} N} k \textit{ (few spurious elements)}$$

with probability at least $1 - 1/\log N$ over the internal randomness used by Algorithm 2. The sample complexity is $2^{O(d^2)} k \log N (\log \log N)$, and the runtime is $2^{O(d^2)} k \log^{d+3} N$.

Recovery at constant ℓ_1-SNR. Once (5) has been achieved, we run the RECOVERATCONSTANTSNR primitive on the residual signal. This algorithm only uses a single hashing into $\approx k/\epsilon$ buckets followed by an estimation step. Adding the correction χ' that it outputs to the output $\chi^{(T)}$ of the SNR reduction loop gives the final output of the algorithm. The RECOVERATCONSTANTSNR primitive is very similar to the corresponding primitive from [IKP14], but requires a stronger analysis due to our weaker ℓ_1-SNR assumption. In the full version of the paper [Kap16] we prove

LEMMA 3.5. *For any $\epsilon \in (0,1)$, $\hat{x}, \chi \in \mathbb{C}^N$, $x' = x - \chi$ and any integer $k \geq 1$ if $||x'_{[2k]}||_1 \leq O(||x_{[n]^d \setminus [k]}||_2 \sqrt{k})$ and $||x'_{[n]^d \setminus [2k]}||_2^2 \leq ||x_{[n]^d \setminus [k]}||_2^2$, the following conditions hold. If $||x||_\infty/\mu = N^{O(1)}$, then the output χ' of* RECOVERATCONSTANTSNR$(\hat{x}, \chi, 2k, \epsilon)$ *satisfies*

$$||x' - \chi'||_2^2 \leq (1 + O(\epsilon))||x_{[n]^d \setminus [k]}||_2^2$$

with at least 99/100 probability over its internal randomness. The sample complexity is $2^{O(d^2)} \frac{1}{\epsilon} k \log N$, and the runtime complexity is at most $2^{O(d^2)} \frac{1}{\epsilon} k \log^{d+1} N$.

We give the intuition behind the proof here, as the argument is somewhat more delicate than the analysis of RECOVERATCONSTSNR in [IKP14], due to the ℓ_1-SNR, rather than ℓ_2-SNR assumption. Specifically, if instead of $||(x - \chi)_{[2k]}||_1 \leq O(\mu k)$ we had $||(x - \chi)_{[2k]}||_2^2 \leq O(\mu^2 k)$, then it would be essentially sufficient to note that after a single hashing into about $k/(\epsilon\alpha)$ buckets for a constant $\alpha \in (0,1)$, every element $i \in [2k]$ is recovered with probability at least $1 - O(\epsilon\alpha)$, say, as it is enough to (on average) recover all but about an ϵ fraction of coefficients. This would not be sufficient here since we only have a bound on the ℓ_1 norm of the residual, and hence some elements can contribute much more ℓ_2 norm than others. However, we are able to show that the probability that an element of the residual signal x'_i is not recovered is bounded by $O(\frac{\alpha\epsilon\mu^2}{|x'_i|^2} + \frac{\alpha\epsilon\mu}{|x'_i|})$, where the first term corresponds to contribution of tail noise and the second corresponds to the head elements. This bound implies that the total expected ℓ_2^2 mass in the elements that are not recovered is upper bounded by $\sum_{i \in [2k]} |x'_i|^2 \cdot O(\frac{\alpha\epsilon\mu^2}{|x'_i|^2} + \frac{\alpha\epsilon\mu}{|x'_i|}) \leq O(\epsilon\mu^2 k + \epsilon\mu \sum_{i \in [2k]} |x'_i|) = O(\epsilon\mu^2 k)$, giving the result.

Finally, putting the results above together, in section 6 we prove

THEOREM 3.6. *For any $\epsilon \in (0,1)$, $x \in \mathbb{C}^{[n]^d}$ and any integer $k \geq 1$, if $R^* \geq ||x||_\infty/\mu = N^{O(1)}$, $\mu^2 \geq ||x_{[n]^d \setminus [k]}||_2^2/k$, $\mu = O(||x_{[n]^d \setminus [k]}||_2^2/k)$ and $\alpha \in (0,1)$ is smaller than a constant,* SPARSEFFT$(\hat{x}, k, \epsilon, R^*, \mu)$ *solves the ℓ_2/ℓ_2 sparse recovery problem using $2^{O(d^2)} \frac{1}{\epsilon} k \log N + 2^{O(d^2)} k \log N \log\log N$ samples and $2^{O(d^2)} \frac{1}{\epsilon} k \log^{d+3} N$ time with at least 98/100 success probability.*

Semi-equispaced Fourier transform. Note that the runtime of our algorithm is $k\mathrm{poly}(\log N)$ for any fixed dimension d, but depends exponentially on d. The $k \log^d N$ term in the runtime comes from the fact that we update the residual signal in frequency domain. These updates are done via semi-equispaced Fourier transform, which runs in time exponential in the dimension. We defer the details to the full version of the paper.

4. ANALYSIS OF LOCATESIGNAL: MAIN DEFINITIONS AND BASIC CLAIMS

In this section we state our main signal location primitive, LOCATESIGNAL (Algorithm 1). Given a sequence of measurements $m(\widehat{x}, H_r, a \star (\mathbf{1}, \mathbf{w}))\}_{a \in \mathcal{A}_r, \mathbf{w} \in \mathcal{W}}, r = 1, \ldots, r_{max}$ a signal $\widehat{x} \in \mathbb{C}^{[n]^d}$ and a partially recovered signal $\chi \in \mathbb{C}^{[n]^d}$, LOCATESIGNAL outputs a list of locations $L \subseteq [n]^d$ that, as we show below in Theorem 3.1 (see section 5), contains the elements of x that contribute most of its ℓ_1 mass. An important feature of LOCATESIGNAL is that it is an entirely deterministic procedure, giving recovery guarantees for any signal x and any partially recovered signal χ. As Theorem 3.1 shows, however, these guarantees are strongest when most of the mass of the residual $x - \chi$ resides on elements in $[n]^d$ that are *isolated with respect to most hashings* $H_1, \ldots, H_{r_{max}}$ used for measurements. This flexibility is crucial for our analysis, and is exactly what allows us to reuse measurements and thereby achieve near-optimal sample complexity.

In the rest of this section we derive useful characterization of elements i of the input signal $(x - \chi)_i$ that are successfully located by LOCATESIGNAL. The main result of this section is Corollary 4.2. The main technical challenge in proving the main result comes down to bounding, for a given input signal x and partially recovered signal χ, the ℓ_1 norm of the noise

contributed to the process of locating heavy hitters in a call to LocateSignal$(\widehat{x}, \chi, H, \{m(\widehat{x}, H, a\star(\mathbf{1},\mathbf{w}))\}_{a\in\mathcal{A},\mathbf{w}\in\mathcal{W}})$ by **(a)** the tail of the original signal x (tail noise e^{tail}) and **(b)** the heavy hitters and false positives (heavy hitter noise e^{head}).

Algorithm 2 SparseFFT$(\hat{x}, k, \epsilon, R^*, \mu)$

procedure SparseFFT$(\hat{x}, k, \epsilon, R^*, \mu)$
  $\chi^{(0)} \leftarrow 0$ ▷ in $\mathbb{C}^n$.
  $T \leftarrow \log_{(\log^4 N)} R^*$, $F \leftarrow 2d$
  $B \leftarrow (2\pi)^{4d\cdot F}\cdot k/\alpha^d$, $\alpha > 0$ sufficiently small constant
  $r_{max} \leftarrow (C/\sqrt{\alpha})\log\log N$, $c_{max} \leftarrow (C/\sqrt{\alpha})\log\log N$
  $\mathcal{W} \leftarrow \{\mathbf{0}_d\}$, $\Delta \leftarrow 2^{\lfloor \frac{1}{2}\log_2\log_2 n\rfloor}$
  for $g = 1$ to $\log_\Delta n$ **do**
    $\mathcal{W} \leftarrow \mathcal{W} \cup \bigcup_{s=1}^{d} n\Delta^{-g}\cdot\mathbf{e}_s$
  end for
  $G \leftarrow$ filter with B buckets and sharpness F
  for $r = 1$ to r_{max} **do**
    Choose $\Sigma_r \in \mathcal{M}_{d\times d}$, $q_r \in [n]^d$ u.a.r.
    Let $\pi_r := (\Sigma_r, q_r)$, let $H_r := (\pi_r, B, F)$
    Let $\mathcal{A}_r \leftarrow C\log\log N$ elements of $[n]^d \times [n]^d$
    sampled u.a.r. with replacement
    for $\mathbf{w} \in \mathcal{W}, a \in \mathcal{A}_r$
    $m(\widehat{x}, H_r, z) \leftarrow$ HashToBins$(\hat{x}, 0, (H_r, a\star(\mathbf{1},\mathbf{w})))$
    end for
  end for
  for $t = 0, \ldots, T-1$ **do**
    $\chi' \leftarrow$ ReduceL1Norm$\Big(\chi^{(t)}, k,$
    $\{m(\widehat{x}, H_r, a\star(\mathbf{1},\mathbf{w}))\}_{r=1,a\in\mathcal{A}_r,\mathbf{w}\in\mathcal{W}}^{r_{max}},$
    $4\mu(\log^4 n)^{T-t}, \mu\Big)$
    Let $\nu' \leftarrow (\log^4 N)(4\mu(\log^4 N)^{T-(t+1)} + 20\mu)$
    $\chi'' \leftarrow$ ReduceInfNorm$(\hat{x}, \chi^{(t)}+\chi', \frac{4k}{\log^4 N}, \nu', \nu')$
    $\chi^{(t+1)} \leftarrow \chi^{(t)} + \chi' + \chi''$
  end for
  $\chi' \leftarrow$ RecoverAtConstantSNR$(\hat{x}, \chi^{(T)}, 2k, \epsilon)$
  return $\chi^{(T)} + \chi'$
end procedure

Let $S \subseteq [n]^d$. The characterization of elements i than be recovered from a call to LocateSignal that we derive now will hold for any set S. However, useful instantiations of this characterization will involve a set S that contains the 'heavy' elements of the signal x (see, e.g. (4)). The size of S will be equal to about k. For a hashing H and set $\mathcal{A} \subseteq [n]^d \times [n]^d$ consider a call to

$$\text{LocateSignal}(\chi, H, \{m(\widehat{x}, H, a\star(\mathbf{1},\mathbf{w})\}_{a\in\mathcal{A},\mathbf{w}\in\mathcal{W}}).$$

For each $a \in \mathcal{A}$ and fixed $\mathbf{w} \in \mathcal{W}$ we let $z := a\star(\mathbf{1},\mathbf{w}) \in [n]^d$ to simplify notation. The measurement vectors $m := m(\widehat{x'}, H, z)$ computed in LocateSignal satisfy, for every $i \in [n]^d$

$$m_{h(i)} = \sum_{j\in[n]^d} G_{o_i(j)} x'_j \omega^{z^T\Sigma j} + \Delta_{h(i),z},$$

where Δ corresponds to polynomially small estimation noise due to approximate computation of the Fourier transform, and the filter $G_{o_i(j)}$ is the filter corresponding to hashing H.

Algorithm 3 ReduceL1Norm

procedure ReduceL1Norm$(\widehat{x}, \chi, k, \chi^{(t)}, k,$
$\{m(\widehat{x}, H_r, a\star(\mathbf{1},\mathbf{w}))\}_{r=1,a\in\mathcal{A}_r,\mathbf{w}\in\mathcal{W}}^{r_{max}}, \nu, \mu)$
  $\chi^{(0)} \leftarrow 0$, $B \leftarrow (2\pi)^{4d\cdot F}\cdot k/\alpha^d$
  for $t = 0$ to $\lceil\log_2(\log^4 N)\rceil - 1$ **do**
    for $r = 1$ to r_{max} **do**
      $L_r \leftarrow$ LocateSignal$\Big(\chi+\chi^{(t)}, k,$
      $\{m(\widehat{x}, H_r, a\star(\mathbf{1},\mathbf{w}))\}_{r=1,a\in\mathcal{A}_r,\mathbf{w}\in\mathcal{W}}^{r_{max}}\Big)$
    end for
    $L \leftarrow \bigcup_{r=1}^{r_{max}} L_r$
    $\chi' \leftarrow$ EstimateValues$\Big(\widehat{x}, \chi+\chi^{(t)}, L, 4k, 1,$
    $\frac{1}{1000}\nu 2^{-t} + 4\mu, C(\log\log N + d^2 + \log(B/k))\Big)$
    $\chi^{(t+1)} \leftarrow \chi^{(t)} + \chi'$
  end for
  return $\chi + \chi^{(\lceil\log_2(\log^4 N)\rceil)}$
end procedure

In particular, for each hashing H and parameter $a \in [n]^d$

$$G^{-1}_{o_i(i)} m_{h(i)}\omega^{-z^T\Sigma i} = x'_i + G^{-1}_{o_i(i)} \sum_{j\in[n]^d\setminus\{i\}} G_{o_i(j)} x'_j \omega^{z^T\Sigma(j-i)} + G^{-1}_{o_i(i)}\Delta_{h(i),z}\omega^{-z^T\Sigma i}$$

It is useful to represent the residual signal x as a sum of three terms: $x' = (x-\chi)_S - \chi_{[n]^d\setminus S} + x_{[n]^d\setminus S}$, where the first term is the residual signal coming from the 'heavy' elements in S, the second corresponds to false positives, or spurious elements discovered and erroneously subtracted by the algorithm, and the third corresponds to the tail of the signal. Similarly, we bound the noise contributed by the first two (head elements and false positives) and the third (tail noise) parts of the residual signal to the location process separately. For each $i \in S$ we write

$$\begin{aligned} G^{-1}_{o_i(i)} m_{h(i)}\omega^{-z^T\Sigma i} &= x'_i \\ &+ G^{-1}_{o_i(i)}\left[\sum_{\substack{j\in S\\ j\neq i}} G_{o_i(j)} x'_j \omega^{z^T\Sigma(j-i)} - \sum_{\substack{j\notin S\\ j\neq i}} G_{o_i(j)}\chi_j\omega^{z^T\Sigma(j-i)}\right] \\ &+ G^{-1}_{o_i(i)} \sum_{\substack{j\in[n]^d\setminus S\\ j\neq i}} G_{o_i(j)} x_j \omega^{z^T\Sigma(j-i)} \\ &+ G^{-1}_{o_i(i)}\Delta_{h(i)}\omega^{-z^T\Sigma i}. \end{aligned} \tag{6}$$

Noise from heavy hitters. The first term in (6) corresponds to noise from $(x-\chi)_{S\setminus\{i\}} - \chi_{[n]^d\setminus S\setminus\{i\}}$, i.e. noise from heavy hitters and false positives. For every $i \in S$, hashing H we let

$$e_i^{head}(H, x, \chi) := G^{-1}_{o_i(i)} \cdot \sum_{j\in[n]^d\setminus\{i\}} G_{o_i(j)}|y_j|, \tag{7}$$

where $y = (x-\chi)_S - \chi_{[n]^d\setminus S}$. We thus get that $e_i^{head}(H, x, \chi)$ upper bounds the absolute value of the first error term in (6). Note that $G \geq 0$ by Lemma 2.3 as long as F is even, which is the setting that we are in. If $e_i^{head}(H, x, \chi)$ is large, LocateSignal may not be able to locate i using measurements of the residual signal $x - \chi$ taken with hashing H. However,

the noise in other hashings may be smaller, allowing recovery. In order to reflect this fact we define, for a sequence of hashings $H_1, \ldots, H_r$ and a signal $y \in \mathbb{C}^{[n]^d}$

$$e_i^{head}(\{H_r\}, x, \chi) := \text{quant}_r^{1/5} e_i^{head}(H_r, x, \chi), \tag{8}$$

where for a list of reals $u_1, \ldots, u_s$ and a number $f \in (0,1)$ we let $\text{quant}^f(u_1, \ldots, u_s)$ denote the $\lceil f \cdot s \rceil$-th largest element of $u_1, \ldots, u_s$.

Tail noise. To capture the second term in (6) (corresponding to tail noise), we define, for any $i \in S, z \in [n]^d, \mathbf{w} \in \mathcal{W}$, permutation $\pi = (\Sigma, q)$ and hashing $H = (\pi, B, F)$

$$e_i^{tail}(H, z, x) := \left| G_{o_i(i)}^{-1} \cdot \sum_{j \in [n]^d \setminus S, j \neq i} G_{o_i(j)} x_j \omega^{z^T \Sigma (j-i)} \right|. \tag{9}$$

With this definition in place $e_i^{tail}(H, z, x)$ upper bounds the second term in (6). As our algorithm uses several values of $a \in \mathcal{A}_r \subseteq [n]^d \times [n]^d$ to perform location, a more robust version of $e_i^{tail}(H, z, x)$ will be useful. To that effect we let for any $\mathcal{Z} \subseteq [n]^d$ (we will later use $\mathcal{Z} = \mathcal{A}_r \star (\mathbf{1}, \mathbf{w})$ for various $\mathbf{w} \in \mathcal{W}$)

$$e_i^{tail}(H, \mathcal{Z}, x) := \text{quant}_{z \in \mathcal{Z}}^{1/5} e_i^{tail}(H, z, x). \tag{10}$$

Note that the algorithm first selects sets $\mathcal{A}_r \subseteq [n]^d \times [n]^d$, and then access the signal at locations $\mathcal{A}_r \star (\mathbf{1}, \mathbf{w}), \mathbf{w} \in \mathcal{W}$.

The definition of $e_i^{tail}(H, \mathcal{A} \star (\mathbf{1}, \mathbf{w}), x)$ for a fixed $\mathbf{w} \in \mathcal{W}$ allows us to capture the amount of noise that our measurements for locating a specific set of bits of Σi that use H suffer from. Since the algorithm requires all $\mathbf{w} \in \mathcal{W}$ to be not too noisy in order to succeed (see precondition 2 of Lemma 4.1), we introduce notation that captures this. Define

$$\begin{aligned} e_i^{tail}(H, \mathcal{A}, x) &:= 40\mu_{H,i}(x) \\ &+ \sum_{\mathbf{w} \in \mathcal{W}} \left| e_i^{tail}(H, \mathcal{A} \star (\mathbf{1}, \mathbf{w}), x) - 40\mu_{H,i}(x) \right|_+ \end{aligned} \tag{11}$$

where $|\eta|_+ = \max\{0, \eta\}$. The following definition is useful for bounding the norm of elements $i \in S$ that are not discovered by several calls to LocateSignal on a sequence of hashings $\{H_r\}$. For a sequence of measurement patterns $\{H_r, \mathcal{A}_r\}$ we let

$$e_i^{tail}(\{H_r, \mathcal{A}_r\}, x) := \text{quant}_r^{1/5} e_i^{tail}(H_r, \mathcal{A}_r, x). \tag{12}$$

Finally, for any $S \subseteq [n]^d$ we let $e_S^{head}(\cdot) := \sum_{i \in S} e_i^{head}(\cdot)$ and $e_S^{tail}(\cdot) := \sum_{i \in S} e_i^{tail}(\cdot)$, where $\cdot$ stands for any set of parameters as above. Equipped with the definitions above, we now prove the following lemma, which yields sufficient conditions for recovery of elements $i \in S$ in LocateSignal in terms of e^{head} and e^{tail}.

Lemma 4.1. *Let $H = (\pi, B, R)$ be a hashing, and let $\mathcal{A} \subseteq [n]^d \times [n]^d$. Then for every $S \subseteq [n]^d$ and for every $x, \chi \in \mathbb{C}^{[n]^d}$ and $x' = x - \chi$, the following conditions hold. Let L denote the output of*

$$\text{LocateSignal}(\chi, H, \{m(\widehat{x}, H, a \star (\mathbf{1}, \mathbf{w}))\}_{a \in \mathcal{A}, \mathbf{w} \in \mathcal{W}}).$$

Then for any $i \in S$ such that $|x_i'| > N^{-\Omega(c)}$, if there exists $r \in [1 : r_{max}]$ such that **(1)** *$e_i^{head}(H, x') < |x_i'|/20$,* **(2)** *$e_i^{tail}(H, \mathcal{A} \star (\mathbf{1}, \mathbf{w}), x) < |x_i'|/20$ for all $\mathbf{w} \in \mathcal{W}$, and* **(3)** *for every $s \in [1 : d]$ the set $\mathcal{A} \star (\mathbf{0}, \mathbf{e}_s)$ is balanced in coordinate s (as per Definition 2.12), then $i \in L$. The time taken by the invocation of* LocateSignal *is $O(B \cdot \log^{d+1} N)$. Here c is a (large) constant that governs precision of our semi-equispaced Fourier transform computation.*

Proof Outline: We show that each coordinate $s = 1, \ldots, d$ of Σi is successfully recovered in LocateSignal. Let $q = \Sigma i$ for convenience. Fix $s \in [1 : d]$. We show by induction on $g = 0, \ldots, \log_\Delta n - 1$ that after the g-th iteration of lines 10-13 of Algorithm 1 we have that $\mathbf{f}_s$ coincides with $\mathbf{q}_s$ on the bottom $g \cdot \log_2 \Delta$ bits, i.e. $\mathbf{f}_s - \mathbf{q}_s = 0 \mod \Delta^g$ (note that we trivially have $\mathbf{f}_s < \Delta^g$ after iteration g).

The **base** of the induction is trivial and is provided by $g = 0$. We now show the **inductive step**. Assume by the inductive hypothesis that $\mathbf{f}_s - \mathbf{q}_s = 0 \mod \Delta^{g-1}$, so that $\mathbf{q}_s = \mathbf{f}_s + \Delta^{g-1}(r_0 + \Delta r_1 + \Delta^2 r_2 + \ldots)$ for some sequence $r_0, r_1, \ldots, 0 \leq r_j < \Delta$. Thus, $(r_0, r_1, \ldots)$ is the expansion of $(\mathbf{q}_s - \mathbf{f}_s)/\Delta^{g-1}$ base Δ, and r_0 is the least significant digit. We now show that r_0 is the unique value of r that satisfies the conditions of lines 11-12 of Algorithm 1.

Let $j := h(i)$. We will show that i is recovered from bucket j. The bounds above imply that

$$\frac{m_j(\widehat{x'}, H, a \star (\mathbf{1}, \mathbf{w}))}{m_j(\widehat{x'}, H, a \star (\mathbf{1}, \mathbf{0}))} = \frac{x_i' \omega^{(a \star (\mathbf{1}, \mathbf{w}))^T \mathbf{q}} + E'}{x_i' \omega^{(a \star (\mathbf{1}, \mathbf{0}))^T \mathbf{q}} + E''} \tag{13}$$

for some E', E'' satisfying $|E'| \leq e_i^{head}(H, x, \chi) + e_i^{tail}(H, a \star (\mathbf{1}, \mathbf{w}), x) + N^{-\Omega(c)}$ and $|E''| \leq e_i^{head}(H, x, \chi) + e_i^{tail}(H, a \star (\mathbf{1}, \mathbf{0}), x) + N^{-\Omega(c)}$. For all but $1/5$ fraction of $a \in \mathcal{A}$ we have by assumption of the lemma on e^{tail} and definition of e^{tail} (see (10)) that **both** error terms are less than $|x_i'|/20$, so

$$\begin{aligned} \frac{m_j(\widehat{x'}, H, a \star (\mathbf{1}, \mathbf{w}))}{m_j(\widehat{x'}, H, a \star (\mathbf{1}, \mathbf{0}))} &= \frac{x_i' \omega^{(a \star (\mathbf{1}, \mathbf{w}))^T \mathbf{q}} + E'}{x_i' \omega^{(a \star (\mathbf{1}, \mathbf{0}))^T \mathbf{q}} + E''} \\ &= \omega^{(a \star (\mathbf{0}, \mathbf{w}))^T \mathbf{q}} \cdot \xi, \end{aligned} \tag{14}$$

where $\xi \approx 1$. Algebraic manipulation then shows that the quantity on line 12 of Algorithm 1 equals $\omega_\Delta^{(-r+r_0) \cdot (a \star (\mathbf{0}, \mathbf{e}_s))_s}$. We thus have by the assumption that the set $\mathcal{A} \star (\mathbf{0}, \mathbf{e}_s)$ is balanced in coordinate s that if $r \neq r_0$, this quantity is rather uniformly distributed over the unit circle. When $r = r_0$, this quantity is quite concentrated around 1, and hence we are able to distinguish between the two cases and learn the next $\log_2 \Delta$ bits of i, providing the inductive step. Note that the definitions of e^{head} and e^{tail} involve quantiles of the actual sequences of errors contributed by head and tail elements – this fact allows us to show that *for any χ, which may depend on the measurements*, if e^{head} and e^{tail} are small, then the bits are decoded correctly. The proof only relies on pseudorandom properties of the set of measurements, and in particular is robust enough to handle some dependence between the measurements and the signal $x' = x - \chi$ (assuming bounds on head and tail error as above). □

The following simple corollary is crucial to our proof of Theorem 3.1 in the next section.

Corollary 4.2. *For any integer $r_{max} \geq 1$, for any sequence of r_{max} hashings $H_r = (\pi_r, B, R), r \in [1 : r_{max}]$ and evaluation points $\mathcal{A}_r \subseteq [n]^d \times [n]^d$, for every $S \subseteq [n]^d$ and for every $x, \chi \in \mathbb{C}^{[n]^d}, x' := x - \chi$, the following conditions hold. If for each $r \in [1 : r_{max}]$ $L_r \subseteq [n]^d$ denotes the output of* LocateSignal$(\widehat{x}, \chi, H_r, \{m(\widehat{x}, H_r, a \star (\mathbf{1}, \mathbf{w}))\}_{a \in \mathcal{A}_r, \mathbf{w} \in \mathcal{W}})$*, $L = \bigcup_{r=1}^{r_{max}} L_r$, and the sets $\mathcal{A}_r \star (\mathbf{0}, \mathbf{e}_s)$ are balanced for all $s \in [1 : d]$ and $r \in [1 : r_{max}]$, then*

$||x'_{S\setminus L}||_1 \leq 20||e_S^{head}(\{H_r\},x,\chi)||_1 + 20||e_S^{tail}(\{H_r,\mathcal{A}_r\},x)||_1 + |S|\cdot N^{-\Omega(c)}$. *Furthermore, every element $i \in S$ such that $|x'_i| > 20(e_i^{head}(\{H_r\},x,\chi)+e_i^{tail}(\{H_r,\mathcal{A}_r\},x))+N^{-\Omega(c)}$ belongs to L.*

5. ANALYSIS OF LocateSignal: BOUNDING ℓ_1 NORM OF UNDISCOVERED ELEMENTS

The main result of this section is Theorem 3.1, which is our main tool for showing efficiency of LocateSignal. Theorem 3.1 applies to the following setting. Fix a set $S \subseteq [n]^d$ and a set of hashings $H_1,\ldots,H_{r_{max}}$, and let $S^* \subseteq S$ denote the set of elements of S that are not isolated with respect to most of these hashings $H_1,\ldots,H_{r_{max}}$. Theorem 3.1 shows that for any signal x and partially recovered signal χ, if L denotes the output list of an invocation of LocateSignal on the pair (x,χ) with hashings $H_1,\ldots,H_{r_{max}}$, then the ℓ_1 norm of elements of the residual $(x-\chi)_S$ that are not discovered by LocateSignal can be bounded by a function of the amount of ℓ_1 mass of the residual that fell outside of the 'good' set $S\setminus S^*$, plus the 'noise level' $\mu \geq ||x_{[n]^d\setminus S}||_\infty$ times k. We start with

Lemma 5.1. *Let $k \geq 1$ be an integer, $x,\chi \in \mathbb{C}^N, x' = x-\chi$, and $S \subseteq [n]^d, |S| \leq 2k$. Let $B \geq (2\pi)^{4d\cdot F}\cdot k/\alpha^d$ for some $\alpha \in (0,1)$. Let $\pi = (\Sigma,q)$ be a permutation, $H=(\pi,B,F), F \geq 2d$ be a hashing into B buckets with filter G of sharpness F. Then if $S_H^* \subseteq S$ is the set of elements $i \in S$ that are not isolated under H, one has, for e^{head} defined with respect to S, $||e^{head}_{S\setminus S_H^*}(H,x,\chi)||_1 \leq 2^{O(d)}\alpha^{d/2}||x'_{S\setminus S_H^*}||_1 + (2\pi)^{d\cdot F}\cdot 2^{O(d)}(||x'_{S^*}||_1 + ||\chi_{[n]^d\setminus S}||_1)$. Furthermore, if $\chi_{[n]^d\setminus S} = 0$, then $||e^{head}_{S\setminus S_H^*}(H,x,\chi)||_\infty \leq 2^{O(d)}\alpha^{d/2}||x'_S||_\infty$.*

Proof. By (7) for $i \in S\setminus S_H^*$

$$\begin{aligned} e_i^{head}(H,x,\chi) &= |G^{-1}_{o_i(i)}|\cdot \sum_{j\in S\setminus S_H^*\setminus\{i\}} |G_{o_i(j)}||x'_j| \\ &+ |G^{-1}_{o_i(i)}|\cdot\left[\sum_{j\in S_H^*\setminus\{i\}}|G_{o_i(j)}||x'_j| + \sum_{j\in[n]^d\setminus S\setminus\{i\}}|G_{o_i(j)}||\chi_j|\right] \\ &= |G^{-1}_{o_i(i)}|\cdot(A_1(i)+A_2(i)), \end{aligned} \tag{15}$$

where $A_1(i)$ is the contribution of isolated head elements to i and $A_2(i)$ is the contribution of non-isolated head elements and false positives. Let $A_1 := \sum_{i\in S\setminus S_H^*} A_1(i), A_2 := \sum_{i\in S\setminus S_H^*} A_2(i)$. We bound A_1 and A_2 separately.

Bounding A_1. We upper bound A_1 by

$$\begin{aligned} &\sum_{i\in S\setminus S_H^*}\sum_{j\in S\setminus S_H^*\setminus\{i\}}|G_{o_i(j)}||x'_j| \\ &= \sum_{t\geq 0}\sum_{i\in S\setminus S_H^*}\sum_{\substack{j\in S\setminus S_H^*\setminus\{i\}\text{ s.t.}\\ ||\pi(j)-\pi(i)||_\infty\in(n/b)\cdot[2^t-1,2^{t+1}-1)}}|G_{o_i(j)}||x'_j| \end{aligned} \tag{16}$$

$$\leq \sum_{t\geq 0}\sum_{i\in S\setminus S_H^*}\max_{\substack{||\pi(j)-\pi(i)||_\infty\geq\\ (n/b)\cdot(2^t-1)}} G_{o_i(j)}\cdot \sum_{\substack{j\in S\setminus S_H^*\setminus\{i\}\\ \text{s.t. }||\pi(j)-\pi(i)||_\infty\\ \leq(n/b)\cdot(2^{t+1}-1)}}|x'_j|$$

$$= \sum_{j\in S\setminus S_H^*}|x'_j|\cdot\eta_j,$$

where we let $\eta_j = \sum_{t\geq 0}\max_{\substack{||\pi(j)-\pi(i)||_\infty\geq\\ (n/b)\cdot(2^t-1)}} G_{o_i(j)}\cdot r(j,t)$, and $r(j,t)$ equals the number of points $i \in S\setminus S_H^*\setminus\{j\}$ such that $||\pi(j)-\pi(i)||_\infty \leq (n/b)\cdot(2^{t+1}-1)$. Note that in the first line of (16) we summed, over all $i \in S\setminus S_H^*$ (i.e. all isolated i), the contributions of all other $i \in S$ to the noise in their buckets. We need to bound the first line in terms of $||x'_{S\setminus S_H^*}||_1$. For that, we first classified all $j \in S\setminus S_H^*$ according to the ℓ_∞ distance from i to j (in the second line), then upper bounded the value of the filter $G_{o_i(j)}$ based on the distance $||\pi(i)-\pi(j)||_\infty$, and finally changed order of summation to ensure that the outer summation is a weighted sum of absolute values of x'_j *over all* $j \in S\setminus S_H^*$[3]. In order to upper bound A_1 it now suffices to upper bound η_j. As we now show, a strong bound follows from isolation properties of $j \in S\setminus S_H^*$.

We start by upper bounding G using Lemma 2.3, **(2)**. We first note that by triangle inequality $||o_i(j)||_\infty = ||\pi(j) - \frac{n}{b}h(i)||_\infty \geq ||\pi(j)-\pi(i)||_\infty - ||\pi(i)-\frac{n}{b}h(i)||_\infty \geq \frac{n}{b}(2^t-1) - \frac{n}{b} = \frac{n}{b}(2^{t-1}-2)$. The rhs is positive for all $t \geq 3$ and for such t satisfies $2^{t-1}-2 \geq 2^{t-2}$. We hence get for all $t \geq 3$

$$\begin{aligned} \max_{\substack{||\pi(j)-\pi(i)||_\infty\geq\\ (n/b)\cdot(2^{t-1}-1)}} G_{o_i(j)} &\leq \left(\frac{2}{1+||\pi(j)-(n/b)h(i)||_\infty}\right)^F \\ &\leq \left(\frac{2}{1+2^{t-2}}\right)^F \leq 2^{-(t-3)F}. \end{aligned} \tag{17}$$

We also have the bound $||G||_\infty \leq 1$ from Lemma 2.3, **(3)**. It remains to bound the last term on the rhs of the last line in (16). We need the fact that for a pair i,j such that $||\pi(j)-\pi(i)||_\infty \leq 2^{t+1}-1$ we have by triangle inequality $||\pi(j)-\frac{n}{b}h(i)||_\infty \leq ||\pi(j)-\pi(i)||_\infty + ||\pi(i)-(n/b)h(i)||_\infty \leq (n/b)(2^{t+1}-1)+(n/b) = (n/b)2^{t+1}$. Equipped with this bound, we now conclude that

$$\begin{aligned} &\left|\left\{i\in S\setminus S_H^*\setminus\{j\}\text{ s.t. }||\pi(j)-\pi(i)||_\infty \leq \frac{n}{b}\cdot(2^{t+1}-1)\right\}\right| \\ &= |\pi(S\setminus\{i\})\cap\mathbb{B}^\infty_{(n/b)h(i)}(\frac{n}{b}\cdot 2^{t+1})| \\ &\leq (2\pi)^{-d\cdot F}\cdot\alpha^{d/2}2^{(t+2)d+1}\cdot 2^t, \end{aligned} \tag{18}$$

where we used the assumption that $i \in S\setminus S_H^*$ are isolated with respect to S (see Definition 2.9). We thus get for any $j \in S\setminus S_H^*$

$$\begin{aligned} \eta_j &= \sum_{t\geq 0}\max_{\substack{||\pi(j)-\pi(i)||_\infty\geq\\ (n/b)\cdot(2^t-1)}} G_{o_i(j)}\cdot r(j,t) \\ &\leq \sum_{t\geq 0}((2\pi)^{-d\cdot F}\cdot\alpha^{d/2}2^{(t+2)d+1}\cdot 2^t)\min\{1,2^{-(t-3)F}\} \\ &\leq (2\pi)^{-d\cdot F}\cdot\alpha^{d/2}2^{2d+1}\sum_{t\geq 0}2^{t(d+1)}\cdot\min\{1,2^{-(t-3)F}\} \end{aligned}$$

[3]We note here that we started by summing over i first and then over j, but switched the order of summation to the opposite in the last line. This is because the quantity $G_{o_i(j)}$, which determines contribution of $j \in S$ to the estimation error of $i \in S$ *is not symmetric* in i and j.

We now note that

$$\sum_{t\geq 0} 2^{t(d+1)} \cdot \min\{1, 2^{-(t-3)F}\}$$
$$= 1 + 2^{2(d+1)} + 2^{3(d+1)} \sum_{t\geq 3} 2^{(t-3)(d+1)} \cdot 2^{-(t-3)F}$$
$$\leq 1 + 2^{2(d+1)} + 2^{3(d+1)+1} \leq 2^{4(d+1)+1},$$

since $F \geq 2d$ by assumption of the lemma, and hence for all $j \in S \setminus S^*_H$ one has $\eta_j \leq (2\pi)^{-d\cdot F} \cdot 2^{O(d)}\alpha^{d/2}$. Combining the estimates above, we now get

$$A_1 \leq \sum_{j\in S\setminus S^*_H} |x'_j| \cdot \eta_j \leq ||x'_S||_1 (2\pi)^{-d\cdot F} \cdot 2^{O(d)}\alpha^{d/2}.$$

Bounding A_2. The bound on A_2 is obtained using similar ideas to the bound on A_1: we show that $A_2 \leq 2^{O(d)}(||x'_{S^*}||_1 + ||\chi_{[n]^d\setminus S}||_1)$. The details are deferred to the full version of the paper. We note that the bound on A_2 is weaker than the bound on A_1 (e.g. the factor of $(2\pi)^{-d\cdot F} \cdot \alpha^{d/2}$ is not there). This is because $\chi_{[n]^d\setminus S}$ is an adversarially placed signal and we do not have isolation properties with respect to it. In particular, it could be located close to an element of $S \setminus S^*_H$, resulting in the weaker bound. Putting the bounds above together, we get $e^{head}_{S\setminus S^*_H}(H, x, \chi) \leq |G^{-1}_{o_i(i)}| \cdot (A_1 + A_2)$ $\leq 2^{O(d)}\alpha^{d/2}||x'_S||_1 + (2\pi)^{d\cdot F} \cdot 2^{O(d)}(||x'_{S^*}||_1 + ||\chi_{[n]^d\setminus S}||_1)$, as required. Note that we used Lemma 2.3, **(1)** to upper bound $G_{o_i(i)}$. The ℓ_∞ bound for the case when $\chi_{[n]^d\setminus S} = 0$ follows in a similar manner and is hence omitted. □

The previous lemma only bounds the error induced by head elements in a single hashing. Since our location algorithm uses multiple hashings $H_r, r = 1, \ldots, O(\log\log N)$, the following lemma provides a useful stronger bound:

LEMMA 5.2. *Let $k \geq 1$ be an integer, $x, \chi \in \mathbb{C}^N, x' = x - \chi$, and $S \subseteq [n]^d, |S| \leq 2k$. Let $B \geq (2\pi)^{4d\cdot F} \cdot k/\alpha^d$ for some $\alpha \in (0, 1)$. Let $\{\pi_r\}_{r=1}^{r_{max}}$ be a set of permutations, let $H_r = (\pi_r, B, F), F \geq 2d$ be a sequence of hashings into B buckets with filter G of sharpness F. Let S^* denote the set of elements $i \in S$ that are not isolated under at least $\sqrt{\alpha}$ fraction of H_r. Then one has for e^{head} defined with respect to S, $||e^{head}_{S\setminus S^*}(\{H_r\}, x, \chi)||_1 \leq 2^{O(d)}\alpha^{d/2}||x'_S||_1$ $+(2\pi)^{d\cdot F} \cdot 2^{O(d)}||\chi_{[n]^d\setminus S}||_1$. Furthermore, if $\chi_{[n]^d\setminus S} = 0$, then $||e^{head}_{S\setminus S^*}(\{H_r\}, x, \chi)||_\infty \leq 2^{d/2}\alpha^{d/2}||x'_S||_\infty$.*

Proof of Theorem 3.1: Theorem 3.1 now follows by combining Corollary 4.2 with Lemma 5.2 and more standard bounds on the error induced by tail elements e^{tail}. The details of the proof are deferred to the full version of the paper [Kap16]. □

6. ANALYSIS OF SNR REDUCTION LOOP AND SPARSEFFT

In this section we first give an analysis of the SNR reduction loop in SPARSEFFT (Algorithm 2), then correctness of SPARSEFFT and provide runtime bounds. We start with a proof of Theorem 3.4, which we restate here for convenience of the reader:

Theorem 3.4 *(Restated) For any $x \in \mathbb{C}^N$, any integer $k \geq 1$, if $\mu^2 = \mathrm{Err}^2_k(x)/k$ and $R^* \geq ||x||_\infty/\mu = N^{O(1)}$, the following conditions hold for the set $S := \{i \in [n]^d : |x_i| > \mu\} \subseteq [n]^d$.*

The SNR reduction loop of Algorithm 2 (lines 20-27) returns $\chi^{(T)}$ such that

$||(x - \chi^{(T)})_S||_1 \lesssim \mu k$ *(ℓ_1-SNR on head elements is constant)*

$||\chi^{(T)}_{[n]^d\setminus S}||_1 \lesssim \mu k$ *(spurious elements have small ℓ_1 norm)*

$||\chi^{(T)}_{[n]^d\setminus S}||_0 \lesssim \frac{1}{\log^{19} N}k$ *(few spurious elements)*

with probability at least $1 - 1/\log N$ over the internal randomness used by Algorithm 2. The sample complexity is $2^{O(d^2)}k\log N(\log\log N)$. The runtime is bounded by $2^{O(d^2)}k\log^{d+3} N$.

PROOF. First note that $|S^*| \leq 2^{-\Omega(\sqrt{\alpha}r_{max})}|S| \leq 2^{-\Omega(\sqrt{\alpha}r_{max})}k \leq \frac{1}{\log^{19} N}k$ with probability at least $1-2^{-\Omega(r_{max})}$ $\geq 1-1/\log^4 N$ by Lemma 2.11 and choice of $r_{max} \geq (C/\sqrt{\alpha})\log\log N$ for a sufficiently large constant $C > 0$. Also, by Claim 2.13 we have that all sets $\mathcal{A}_r \star (\mathbf{0}, \mathbf{e}_s), s \in [1 : d]$ are balanced (as per Definition 2.12 with $\Delta = 2^{\lfloor \frac{1}{2}\log_2\log_2 n\rfloor}$, as needed for Algorithm 1) with probability at least $1 - 1/\log^4 N$.

We start with correctness. We prove by induction on $t = 0, 1, \ldots, T$ that **(1)** $||(x-\chi^{(t)})_S||_1 \leq 4(\log^4 N)^{T-t}\mu k + 20\mu k$; **(2)** $||x-\chi^{(t)}||_\infty = O((\log^4 N)^{T-(t-1)}\mu)$ and **(3)** $||\chi^{(t)}_{[n]^d\setminus S}||_0 \leq \frac{t}{\log^{20} N}k$. The **base** is provided by $t = 0$, where all claims are trivially true by definition of R^*.

We now prove the **inductive step**. First note that by **(2)** of the inductive hypothesis one has $||x - \chi^{(t)}||_\infty/\mu = R^* \cdot O(\log N) = N^{O(1)}$. We will use Lemma 3.2, and hence first verify that its preconditions are satisfied with $\nu = 4(\log^4 N)^{T-t}$ μk, assuming the inductive hypothesis **(1)-(3)**. First, one has $||(x - \chi^{(t)})_S||_1 \leq 4(\log^4 N)^{T-t}\mu k$. This satisfies precondition **A** of the lemma. Then we have

$$||(x - \chi^{(t)})_{S^*}||_1 + ||\chi^{(t)}_{[n]^d\setminus S}||_1$$
$$\leq ||x - \chi^{(t)}||_\infty \cdot (||(x - \chi^{(t)})_{S^*}||_0 + ||\chi^{(t)}_{[n]^d\setminus S}||_0)$$
$$\leq O(\log^4 N) \cdot \nu \cdot \left(\frac{1}{\log^{19} N}k + \frac{t}{\log^{20} N}k\right) \leq \frac{O(1)}{\log^{14} N}\nu k \quad (19)$$

for sufficiently large N. Since the rhs is less than $\frac{1}{\log^4 N}\nu k$, precondition **C** of Lemma 3.2 is also satisfied. Precondition **B** of the lemma is satisfied by inductive hypothesis, **(3)** together with the fact that $T = o(\log R^*) = o(\log N)$.

Thus, all preconditions of Lemma 3.2 are satisfied. Then by Lemma 3.2 with $\nu = 4(\log^4 N)^{T-t}\mu$ with probability at least $1 - 1/\log^2 N$ one has **(a).** $||(x' - \chi^{(t)} - \chi')_S||_1 \leq \frac{1}{\log^4 N}\nu k + 20\mu k$; **(b).** $||(\chi^{(t)} + \chi')_{[n]^d\setminus S}||_0 - ||\chi^{(t)}_{[n]^d\setminus S}||_0 \leq \frac{1}{\log^{20} N}k$; and **(c)** $||(x'-(\chi^{(t)}+\chi'))_{S^*}||_1 + ||(\chi^{(t)}+\chi')_{[n]^d\setminus S}||_1 \leq ||(x' - \chi^{(t)})_{S^*}||_1 + ||\chi^{(t)}_{[n]^d\setminus S}||_1 + \frac{1}{\log^{20} N}\nu k$. Combining **(a)** above with (19) proves **(1)** of the inductive step:

$$||(x - \chi^{(t+1)})_S||_1 = ||(x - \chi^{(t)} - \chi')_S||_1$$
$$\leq \frac{1}{\log^4 N}\nu k + 20\mu k = \frac{1}{\log^4 N}4(\log^4 N)^{T-t}\mu k + 20\mu k$$
$$= 4(\log^4 N)^{T-(t+1)}\mu k + 20\mu k.$$

Also, combining **(b)** above with the fact that $||\chi^{(t)}_{[n]^d\setminus S}||_0 \leq$

$\frac{t}{\log^{20} N}k$ yields $||\chi^{(t+1)}_{[n]^d\setminus S}||_0 \le \frac{t+1}{\log^{20} N}k$, proving part **(3)** of the inductive step.

In order to prove the inductive step it remains to analyze the call to ReduceInfNorm, for which we use Lemma 3.3 with parameter $\tilde{k} = 4k/\log^4 N$. We first verify that preconditions of the lemma are satisfied. Let $y := x-(\chi+\chi^{(t)}+\chi')$ to simplify notation. We need to verify that

$$||y_{[\tilde{k}]}||_1/\tilde{k} \le 4(\log^4 N)^{T-(t+1)}\mu = (\log^4 N)\cdot(\frac{1}{\log^4 N}\nu + 20\mu) \tag{20}$$

and

$$||y_{[n]^d\setminus[\tilde{k}]}||_2/\sqrt{\tilde{k}} \le (\log^4 N)\cdot(\frac{1}{\log^4 N}\nu + 20\mu). \tag{21}$$

Let $Q := \mathrm{supp}(\chi+\chi^{(t)}+\chi')$ to simplify notation. The first condition is easy to verify:

$$\begin{aligned}
&||y_{\tilde{k}}||_1 \le ||y_S||_1 + ||y_{Q\setminus S}||_1 + ||x_{[n]^d\setminus S}||_\infty\cdot\tilde{k}\\
&\le ||y_S||_1 + ||(\chi+\chi^{(t)}+\chi')_{[n]^d\setminus S}||_1 + ||x_{Q\setminus S}||_\infty\cdot\tilde{k}\\
&+ ||x_{[n]^d\setminus S}||_\infty\cdot\tilde{k}\\
&\le \frac{1}{\log^4 N}\nu k + 20\mu k + \frac{1}{\log^4 N}\nu k + 2\mu\tilde{k} \le \frac{2}{\log^4 N}\nu k + 40\mu k.
\end{aligned}$$

We used triangle inequality to upper bound $||y_{Q\setminus S}||_1$ by $||(\chi^{(t)}+\chi')_{[n]^d\setminus S}||_1 + ||x_{Q\setminus S}||_\infty\cdot\tilde{k}$ to go from the first line to the second. We thus have $||y_{[\tilde{k}]}||_1/\tilde{k} \le (\frac{2}{\log^4 N}\nu k + 40\mu k)/(4k/\log^4 N) \le (\log^4 N)\cdot(\frac{1}{\log^4 N}\nu + 20\mu)$ as required. This establishes (20).

To verify the second condition, we first let $\tilde{S} := S \cup \mathrm{supp}(\chi+\chi^{(t)}+\chi')$ to simplify notation. We have

$$\begin{aligned}
||y_{[n]^d\setminus[\tilde{k}]}||_2^2 &= ||y_{\tilde{S}\setminus[\tilde{k}]}||_2^2 + ||y_{([n]^d\setminus\tilde{S})\setminus[\tilde{k}]}||_2^2\\
&\le ||y_{\tilde{S}\setminus[\tilde{k}]}||_2^2 + \mu^2 k,
\end{aligned} \tag{22}$$

where we used the fact that $y_{[n]^d\setminus\tilde{S}} = x_{[n]^d\setminus\tilde{S}}$ and hence $||y_{([n]^d\setminus\tilde{S})\setminus[\tilde{k}]}||_2^2 \le \mu^2 k$. We now note that $||y_{\tilde{S}\setminus[\tilde{k}]}||_1 \le ||y_{\tilde{S}}||_1 \le 2(\frac{1}{\log^4 N}\nu k + 20\mu k)$, and so it must be that $||y_{S\setminus[\tilde{k}]}||_\infty \le 2(\frac{1}{\log^4 N}\nu k + 20\mu k)(k/\tilde{k})$, as otherwise the top $\tilde{k}$ elements of $y_{[\tilde{k}]}$ would contribute more than $2(\frac{1}{\log^4 N}\nu k + 20\mu k)$ to $||y_{\tilde{S}}||_1$, a contradiction. With these constraints $||y_{\tilde{S}\setminus[\tilde{k}]}||_2^2$ is maximized when there are $\tilde{k}$ elements in $y_{\tilde{S}\setminus[\tilde{k}]}$, all equal to the maximum possible value, i.e. $||y_{\tilde{S}\setminus[\tilde{k}]}||_2^2 \le 4(\frac{1}{\log^4 N}\nu k + 20\mu k)^2(k/\tilde{k})^2\tilde{k}$. Plugging this into (22), we get $||y_{[n]^d\setminus[\tilde{k}]}||_2^2 \le ||y_{\tilde{S}\setminus[\tilde{k}]}||_2^2 + \mu^2 k \le 4(\frac{1}{\log^4 N}\nu k + 20\mu k)^2(k/\tilde{k})^2\tilde{k} + \mu^2 k$. This implies (after some algebraic manipulations) that

$$\begin{aligned}
||y_{[n]^d\setminus[\tilde{k}]}||_2/\sqrt{\tilde{k}} &\le 2((\frac{1}{\log^4 N}\nu k + 20\mu k) + \mu)(k/\tilde{k})\\
&\le (\log^4 N)(\frac{1}{\log^4 N}\nu k + 20\mu k),
\end{aligned}$$

establishing (21). This establishes the bounds on ℓ_1 and ℓ_2 norm of (subsets of) x' required for Lemma 3.3. To complete verification of preconditions, we note that $||y_{S\setminus[\tilde{k}]}||_\infty \le 2(\frac{1}{\log^4 N}\nu k + 20\mu k)(k/\tilde{k}) \le (\log^4 N)\cdot(\frac{1}{\log^4 N}\nu k + 20\mu k)$ and $||y_{[n]^d\setminus\tilde{S}}||_\infty = ||x_{[n]^d\setminus S}||_\infty \le \mu$. We thus have that all preconditions of Lemma 3.3 are satisfied for the set of top $\tilde{k}$ elements of y, and hence its output satisfies $||x-(\chi^{(t)}-\chi'-\chi'')||_\infty = O(\log^4 N)\cdot(\frac{1}{\log^4 N}\nu k + 20\mu k)$. Putting these bounds together establishes **(2)**, and completes the inductive step and the proof of correctness.

Finally, taking a union bound over all failure events (each call to EstimateValues succeeds with probability at least $1-\frac{1}{\log^2 N}$, and $\mathcal{A}_r \star (\mathbf{0}, \mathbf{e}_s), s \in [1:d]$ are balanced with probability $1-1/\log^2 N$) and using the fact that $\log T = o(\log N)$ and each call to LocateSignal is deterministic, we get that success probability of the SNR reduction look is lower bounded by $1-1/\log N$.

Sample complexity and runtime. The sample complexity is bounded by the the sample complexity of the calls to ReduceL1Norm and ReduceInfNorm inside the loop times $O(\log N/\log\log N)$ for the number of iterations. The former is bounded by $2^{O(d^2)}k(\log\log N)^2$ by Lemma 3.2, and the latter is bounded by $2^{O(d^2)}k/\log N$ by Lemma 3.3, amounting to at most $2^{O(d^2)}k\log N(\log\log N)$ samples overall. The runtime complexity is at most $2^{O(d^2)}k\log^{d+3} N$ overall for the calls to ReduceL1Norm and no more than $2^{O(d^2)}k\log^{d+3} N$ for the calls to ReduceInfNorm. □

We can now prove the main result of the paper:

Theorem 3.6 *(Restated) For any $\epsilon \in (0,1)$, $x \in \mathbb{C}^{[n]^d}$ and any integer $k \ge 1$, if $R^* \ge ||x||_\infty/\mu = N^{O(1)}$, $\mu^2 \ge ||x_{[n]^d\setminus[k]}||_2^2/k, \mu = O(||x_{[n]^d\setminus[k]}||_2^2/k)$ and $\alpha \in (0,1)$ is smaller than a constant,* SparseFFT$(\hat{x}, k, \epsilon, R^*, \mu)$ *solves the ℓ_2/ℓ_2 sparse recovery problem using $2^{O(d^2)}\frac{1}{\epsilon}k\log N + 2^{O(d^2)}k\log N\log\log N$ samples and $2^{O(d^2)}\frac{1}{\epsilon}k\log^{d+3} N$ time with at least 98/100 success probability.*

Proof. By Theorem 3.4 the set $S := \{i \in [n]^d : |x_i| > \mu\}$ satisfies $||(x-\chi^{(T)})_S||_1 \lesssim \mu k$, $||\chi^{(T)}_{[n]^d\setminus S}||_1 \lesssim \mu k$ and $||\chi^{(T)}_{[n]^d\setminus S}||_0 \lesssim \frac{1}{\log^{19} N}k$ with probability at least $1-1/\log N$.

We now show that the signal $x' := x-\chi^{(T)}$ satisfies preconditions of Lemma 3.5 with parameter k. Indeed, letting $Q \subseteq [n]^d$ denote the top $2k$ coefficients of x', we have

$$\begin{aligned}
||x'_Q||_1 &\le ||x'_{Q\cap S}||_1 + ||\chi^{(T)}_{(Q\setminus S)\cap\mathrm{supp}\,\chi^{(T)}}||_1 + |Q|\cdot||x_{[n]^d\setminus S}||_1\\
&\le O(\mu k)
\end{aligned}$$

Furthermore, since Q is the set of top $2k$ elements of x', we have

$$\begin{aligned}
||x'_{[n]^d\setminus Q}||_2^2 &\le ||x'_{[n]^d\setminus(S\cup\mathrm{supp}\,\chi^{(T)})}||_2^2 \le ||x_{[n]^d\setminus(S\cup\mathrm{supp}\,\chi^{(T)})}||_2^2\\
&\le ||x_{[n]^d\setminus S}||_2^2 = O(\mu^2 k)
\end{aligned}$$

as required. Thus, with at least 99/100 probability we have by Lemma 3.5 that $||x-\chi^{(T)}-\chi'||_2 \le (1+O(\epsilon))\,\mathrm{Err}_k(x)$. By a union bound over the $1/\log N$ failure probability of the SNR reduction loop we have that SparseFFT is correct with probability at least 98/100, as required.

Sample complexity and runtime. The number of samples needed to compute

$$m(\widehat{x}, H_r, a\star(\mathbf{1},\mathbf{w})) \leftarrow \textsc{HashToBins}(\hat{x}, 0, (H_r, a\star(\mathbf{1},\mathbf{w})))$$

for all $a \in \mathcal{A}_r$, $\mathbf{w} \in \mathcal{W}$ is bounded by $2^{O(d^2)}k\log N(\log\log N)$ by our choice of $B = 2^{O(d^2)}k$, $r_{max} = O(\log\log N)$, $|\mathcal{W}| = O(\log N/\log\log N)$ and $|\mathcal{A}_r| = O(\log\log N)$. This is asymptotically the same as the $2^{O(d^2)}k\log N\cdot(\log\log N)$ sample complexity of the ℓ_1 norm reduction loop by Theorem 3.4.

The sampling complexity of the call to RecoverAtConstantSNR is at most $2^{O(d^2)}\frac{1}{\epsilon}k\log N$ by Lemma 3.5, yielding the claimed bound.

The runtime of the SNR reduction loop is bounded by $2^{O(d^2)}k\log^{d+3}N$ by Theorem 3.4, and the runtime of RecoverAtConstantSNR is at most $2^{O(d^2)}\frac{1}{\epsilon}k\log^{d+2}N$ by Lemma 3.5. □

7. ACKNOWLEDGEMENTS

The author would like to thank Piotr Indyk for many useful discussions at various stages of this work.

8. REFERENCES

[AGS03] A. Akavia, S. Goldwasser, and S. Safra. Proving hard-core predicates using list decoding. *FOCS*, 44:146–159, 2003.

[Aka10] A. Akavia. Deterministic sparse Fourier approximation via fooling arithmetic progressions. *COLT*, pages 381–393, 2010.

[BCG+12] P. Boufounos, V. Cevher, A. C. Gilbert, Y. Li, and M. J. Strauss. What's the frequency, Kenneth?: Sublinear Fourier sampling off the grid. *RANDOM/APPROX*, 2012.

[Bou14] J. Bourgain. An improved estimate in the restricted isometry problem. *GAFA*, 2014.

[CCFC02] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *ICALP*, 2002.

[CGV12] M. Cheraghchi, V. Guruswami, and A. Velingker. Restricted isometry of Fourier matrices and list decodability of random linear codes. *SODA*, 2012.

[Cip00] B. A. Cipra. The Best of the 20th Century: Editors Name Top 10 Algorithms. *SIAM News*, 33, 2000.

[CP10] E. Candes and Y. Plan. A probabilistic and ripless theory of compressed sensing. *IEEE Transactions on Information Theory*, 2010.

[CT06] E. Candes and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies. *IEEE Trans. on Info.Theory*, 2006.

[DIPW10] K. Do Ba, P. Indyk, E. Price, and D. Woodruff. Lower Bounds for Sparse Recovery. *SODA*, 2010.

[Don06] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[GGI+02] A. Gilbert, S. Guha, P. Indyk, M. Muthukrishnan, and M. Strauss. Near-optimal sparse Fourier representations via sampling. *STOC*, 2002.

[GHI+13] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi. Sample-optimal average-case sparse Fourier transform in two dimensions. *Allerton*, 2013.

[GL89] O. Goldreich and L. Levin. A hard-corepredicate for allone-way functions. *STOC*, pages 25–32, 1989.

[GLPS10] A. C. Gilbert, Y. Li, E. Porat, and M. J. Strauss. Approximate sparse recovery: optimizing time and measurements. In *STOC*, pages 475–484, 2010.

[GMS05] A. Gilbert, M. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal space Fourier representations. *SPIE Conference, Wavelets*, 2005.

[HAKI12] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk. Faster GPS via the Sparse Fourier Transform. *MOBICOM*, 2012.

[HIKP12a] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Near-optimal algorithm for sparse Fourier transform. *STOC*, 2012.

[HIKP12b] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and practical algorithm for sparse Fourier transform. *SODA*, 2012.

[HKPV13] S. Heider, S. Kunis, D. Potts, and M. Veit. A sparse Prony FFT. *SAMPTA*, 2013.

[HR16] I. Haviv and O. Regev. The restricted isometry property of subsampled fourier matrices. *SODA*, 2016.

[IK14] P. Indyk and M. Kapralov. Sample-optimal Fourier sampling in any fixed dimension. *FOCS*, 2014.

[IKP14] P. Indyk, M. Kapralov, and E. Price. (Nearly) sample-optimal sparse Fourier transform. *SODA*, 2014.

[Iwe10] M. A. Iwen. Combinatorial sublinear-time Fourier algorithms. *Foundations of Computational Mathematics*, 10:303–338, 2010.

[Iwe12] M.A. Iwen. Improved approximation guarantees for sublinear-time Fourier algorithms. *Applied And Computational Harmonic Analysis*, 2012.

[Kap16] M. Kapralov. Sparse Fourier Transform in any constant dimension with nearly-optimal sample complexity in sublinear time. *http://arxiv.org/abs/1604.00845*, 2016.

[KM91] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *STOC*, 1991.

[LDSP08] M. Lustig, D.L. Donoho, J.M. Santos, and J.M. Pauly. Compressed sensing mri. *Signal Processing Magazine, IEEE*, 25(2):72–82, 2008.

[Man92] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *ICALP*, 1992.

[PR13] S. Pawar and K. Ramchandran. Computing a k-sparse n-length Discrete Fourier Transform using at most $4k$ samples and $O(klogk)$ complexity. *ISIT*, 2013.

[PS15] E. Price and Z. Song. A robust sparse Fourier transform in the continuous setting. *FOCS*, 2015.

[RV08] M. Rudelson and R. Vershynin. On sparse reconstruction from Fourier and Gaussian measurements. *CPAM*, 61(8):1025–1171, 2008.

[Sid11] Emil Sidky. What does compressive sensing mean for X-ray CT and comparisons with its MRI application. In *Conference on Mathematics of Medical Imaging*, 2011.