Sample-Optimal Fourier Sampling in Any Constant Dimension

Sample-Optimal Fourier Sampling in Any Constant Dimension – Part I

Piotr Indyk Michael Kapralov

May 14, 2014

Abstract

We give an algorithm for ℓ_2/ℓ_2 sparse recovery from Fourier measurements using $O(k \log N)$ samples, matching the lower bound of [DIPW10] for non-adaptive algorithms up to constant factors for any $k \leq N^{1-\delta}$. The algorithm runs in $\tilde{O}(N)$ time. Our algorithm extends to higher dimensions, leading to sample complexity of $O_d(k \log N)$, which is optimal up to constant factors for any $d = O(1)$. These are the first sample optimal algorithms for these problems.

A preliminary experimental evaluation indicates that our algorithm has empirical sampling complexity comparable to that of other recovery methods known in the literature, while providing strong provable guarantees on the recovery quality.

1 Introduction

The Discrete Fourier Transform (DFT) is a mathematical notion that allows to represent a sampled signal or function as a combination of discrete frequencies. It is a powerful tool used in many areas of science and engineering. Its popularity stems from the fact that signals are typically easier to process and interpret when represented in the frequency domain. As a result, DFT plays a key role in digital signal processing, image processing, communications, partial differential equation solvers, etc. Many of these applications rely on the fact that most of the Fourier coefficients of the signals are small or equal to zero, i.e., the signals are (approximately) *sparse*. For example, sparsity provides the rationale underlying compression schemes for audio, image and video signals, since keeping the top few coefficients often suffices to preserve most of the signal energy.

An attractive property of sparse signals is that they can be acquired from only a small number of samples. Reducing the sample complexity is highly desirable as it implies a reduction in signal acquisition time, measurement overhead and/or communication cost. For example, one of the main goals in medical imaging is to reduce the sample complexity in order to reduce the time the patient spends in the MRI machine [LDSP08], or the radiation dose received [Sid11]. Similarly in spectrum sensing, a lower average sampling rate enables the fabrication of efficient analog to digital converters (ADCs) that can acquire very wideband multi-GHz signals [YBL$^+$12]. As a result, designing sampling schemes and the associated sparse recovery algorithms has been a subject of extensive research in multiple areas, such as:

- *Compressive sensing:* The area of compressive sensing [Don06, CT06], developed over the last decade, studies the task of recovering (approximately) sparse signals from linear measurements. Although several classes of linear measurements were studied, acquisition of sparse signals using few Fourier measurements (or, equivalently, acquisition of Fourier-sparse signals using few signal samples) has been one of the key problems studied in this area. In particular, the seminal work of [CT06, RV08] has shown that one can recover N-dimensional signals with at most k Fourier coefficients using only $k \log^{O(1)} N$ samples. The recovery algorithms are based on linear programming and run in time polynomial in N. See [FR13] for an introduction to the area.

- *Sparse Fourier Transform:* A different line of research, with origins in computational complexity and learning theory, has been focused on developing algorithms whose sample complexity *and* running time bounds scale with the sparsity. Many such algorithms have been proposed in the literature, including [GL89, KM91, Man92, GGI$^+$02, AGS03, GMS05, Iwe10, Aka10, HIKP12b, HIKP12a, LWC12, BCG$^+$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.

The best known results obtained in both of those areas are summarized in the following table. For the sake of uniformity we focus on algorithms that work for general signals and recover k-sparse approximations satisfying the so-called ℓ_2/ℓ_2 approximation guarantee[1]. In this case, the goal of an algorithm is as follows: given m samples of the Fourier transform $\widehat{x}$ of a signal x[2], and the sparsity parameter k, output x' satisfying

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

[1]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. Also, additional results that hold for exactly sparse signals are known, see e.g., [BCG$^+$12] and references therein.

[2]Here and for the rest of this paper, we will consider the *inverse* discrete Fourier transform problem of estimating a sparse x from samples of $\widehat{x}$. This leads to a simpler notation. Note that the the forward and inverse DFTs are equivalent modulo conjugation.

The algorithms are randomized and succeed with constant probability.

Reference	Time	Samples	Approximation	Signal model
[CT06, RV08]				
[CGV12]	$N \times m$ linear program	$O(k \log^3(k) \log(N))$	$C = O(1)$	worst case
[CP10]	$N \times m$ linear program	$O(k \log N)$	$C = (\log N)^{O(1)}$	worst case
[HIKP12a]	$O(k \log(N) \log(N/k))$	$O(k \log(N) \log(N/k))$	any $C > 1$	worst case
[GHI$^+$13]	$O(k \log^2 N)$	$O(k \log N)$	$C = O(1)$	average case, $k = \Theta(\sqrt{N})$
[PR14]	$O(N \log N)$	$O(k \log N)$	$C = O(1)$	average case, $k = O(N^\alpha)$, $\alpha < 1$
[IKP14]	$O(k \log^2(N) \log^{O(1)} \log N)$	$O(k \log(N) \log^{O(1)} \log N)$	any $C > 1$	worst case
[DIPW10]		$\Omega(k \log(N/k))$	constant C	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.

As evident from the table, none of the results obtained so far was able to guarantee sparse recovery from the optimal number of samples, unless either the approximation factor was super-constant or the result held for average-case signals. In fact, it was not even known whether there is an *exponential time* algorithm that uses only $O(k \log N)$ samples in the worst case.

A second limitation, that applied to the sub-linear time algorithms in the last three rows in the table, but not to compressive sensing algorithms in the first two rows of the table, is that those algorithms were designed for *one-dimensional* signals. However, the sparsest signals often occur in applications involving higher-dimensional DFTs, since they involve much larger signal lengths N. Although one can reduce, e.g., the two-dimensional DFT over $p \times q$ grid to the one-dimensional DFT over a signal of length pq [GMS05, Iwe12]), the reduction applies only if p and q are relatively prime. This excludes the most typical case of $m \times m$ grids where m is a power of 2. The only prior algorithm that applies to general $m \times m$ grids, due 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.

Our results In this paper we give an algorithm that overcomes both of the aforementioned limitations. Specifically, we present an algorithm for the sparse Fourier transform in any fixed dimension that uses only $O(k \log N)$ samples of the signal. This is the first algorithm that matches the lower bound of [DIPW10], for k up to $N^{1-\delta}$ for any constant $\delta > 0$. The recovery algorithm runs in time $O(N \log^{O(1)} N)$.

In addition, we note that the algorithm is in fact quite simple. It is essentially a variant of an iterative thresholding scheme, where the coordinates of the signal are updated sequentially in order to minimize the difference between the current approximation and the underlying signal. In Section 7 we discuss a preliminary experimental evaluation of this algorithm, which shows promising results.

The techniques introduced in this paper have already found applications. In particular, in a followup paper [IK14], we give an algorithm that uses $O(k \log(N) \log^{O(1)} \log N)$ samples of the signal and has the running time of $O(k \log^{O(1)}(N) \log^{O(1)} \log N)$ for any constant d. This generalizes the result of [IKP14] to any constant dimension, at the expense of somewhat larger runtime.

Our techniques The overall outline of our algorithms follows the framework of [GMS05, HIKP12a, IKP14], which adapt the methods of [CCFC02, GLPS10] from arbitrary linear measurements to Fourier ones. 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 denotes *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 pseudo-randomly permuted, by re-arranging the access to $\hat{x}$ via a random affine permutation. Then the coefficients are partitioned into buckets. This steps uses the"filtering" process that approximately partitions the range of x into intervals (or, in higher dimension, squares) with N/B coefficients each, and collapses each interval into one bucket. To minimize the number of samples taken, the filtering process is approximate. In particular the coefficients contribute ("leak"') to buckets other than the one they are nominally mapped into, although that contribution is limited and controlled by the quality of the filter. The details are described in Section 3, see also [HIKP12b] for further overview.

Overall, this probabilistic process 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 is defined as $\text{Err}_k(x) = \min_{k-\text{sparse } y} ||x - y||_2$. This enables the algorithm to identify the positions of the large coefficients, as well as 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. The repetition will yield a good sparse approximation χ of x.

To achieve the optimal number of measurements, however, our algorithm departs from the above scheme in a crucial way: the algorithm does *not* use fresh hash functions in every repetition. 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 duration of the algorithm. Note that each hash function requires a separate set of samples to construct the buckets, so reusing the hash functions means that the number of samples does not grow with the number of iterations. This enables us to achieve the optimal measurement bound.

At the same time reusing the hash functions creates a major difficulty: if the algorithm identifies a non-existing large coefficient by mistake and adds it to χ, this coefficient will be present in the difference vector $x - \chi$ and will need to be corrected later. And unlike the earlier guarantee for the large coefficients of the *original* signal x, we do not have any guarantees that large *erroneous* coefficients will be isolated by the hash functions, since the positions of those coefficients are determined by those functions. Because of these difficulties, almost all prior works[3] either used a fresh set of measurements in each iteration (almost all sparse Fourier transform algorithms fall into this category) or provided stronger deterministic guarantees for the sampling pattern (such as the restricted isometry property [CT06]). However, the latter option required a larger number of measurements to ensure the desired properties. Our algorithm circumvents this difficulty by ensuring that no large coefficients are created erroneously. This is non-trivial, since the hashing process is quite noisy (e.g, the bucketing process suffers from leakage). Our solution is to recover the large coefficients in the decreasing order of their magnitude. Specifically, in each step, we recover coefficients with magnitude

[3]We are only aware of two exceptions: the algorithms of [GHI$^+$13, PR13] (which were analyzed only for the easier case where the large coefficients themselves were randomly distributed) and the analysis of iterative thresholding schemes due to [BLM12] (which relied on the fact that the measurements were induced by Gaussian or Gaussian-like matrices).

that exceeds a specific threshold (that decreases exponentially). The process is designed to ensure that (i) all coefficients above the threshold are recovered and (ii) all recovered coefficients have magnitudes close to the threshold. In this way the set of locations of large coefficients stays fixed (or monotonically decreases) over the duration of the algorithms, and we can ensure the isolation properties of those coefficients during the initial choice of the hash functions.

Overall, our algorithm has two key properties (i) it is iterative, and therefore the values of the coefficients estimated in one stage can be corrected in the second stage and (ii) does not require fresh hash functions (and therefore new measurements) in each iteration. Property (ii) implies that the number of measurements is determined by only a single (first) stage, and does not increase beyond that. Property (i) implies that the bucketing and estimation process can be achieved using rather "crude" filters[4], since the estimated values can be corrected in the future stages. As a result each of the hash function require only $O(k)$ samples; since we use $O(\log N)$ hash functions, the $O(k \log N)$ bound follows. This stands in contrast with the algorithm of [GMS05] (which used crude filters of similar complexity but required new measurements per each iteration) or [HIKP12a] (which used much stronger filters with $O(k \log N)$ sample complexity) or [IKP14] (which used filters of varying quality and sample complexity). The advantage of our approach is amplified in higher dimension, as the ratio of the number of samples required by the filter to the value k grows exponentially in the dimension. Thus, our filters still require $O(k)$ samples in any fixed dimension d, while for [HIKP12a, IKP14] this bound increases to $O(k \log^d N)$.

Organization We give definitions and basic results relevant to sparse recovery from Fourier measurements in section 2. Filters that our algorithm uses are constructed in section 3. Section 4 states the algorithm and provides intuition behind the analysis. The main lemmas of the analysis are proved in section 5, and full analysis of the algorithm is provided in section 6. Results of an experimental evaluation are presented in section 7, and ommitted proofs are given in Appendix A.

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

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

respectively, where $j \in [n]^d$. We will denote the forward Fourier transform by $\mathcal{F}$ and 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). We recover a signal z such that

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

from samples of $\widehat{x}$.

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 $\pi_{\Sigma,q}(i) = \Sigma(i-q)$

[4]In fact, our filters are only slightly more accurate than the filters introduced in [GMS05], and require the same number of samples.

$\bmod n$. 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 $B \approx k/\epsilon$. It should be noted that unlike many sublinear time algorithms for the problem, our algorithm does not use $O(k)$ buckets with centers equispaced in the time domain. Instead, we think of each point in time domain as having a bucket around it. This imporoves the dependence of the number of samples on the dimension d. We will often omit the subscript Σ, q and simply write $\pi(i)$ when Σ, q is fixed or clear from context. For $i, j \in [n]^d$ we let $o_i(j) = \pi(j) - \pi(i)$ to be the "offset" of $j \in [n]^d$ relative to $i \in [n]^d$. We will always have $B = b^d$, where b is a power of 2.

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

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 similar to the proof of Claim B.3 in [GHI$^+$13] and is given in Appendix A for completeness. Define

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

In this paper, we assume knowledge of μ (a constant factor upper bound on μ suffices). We also assume that the signal to noise ration is bounded by a polynomial, namely that $R^* := ||x||_\infty/\mu \le n^C$ for a constant $C > 0$. We use the notation $\mathbb{B}_r^\infty(x)$ to denote the ℓ_∞ ball of radius r around x:

$$\mathbb{B}_r^\infty(x) = \{y \in [n]^d : ||x - y||_\infty \le r\},$$

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 $f = O(g)$.

3 Filter construction and properties

For an integer $b > 0$ a power of 2 let

$$\hat{H}_i^1 = \begin{cases} \frac{\sqrt{n}}{b-1}, & \text{if } |i| < b/2 \\ 0 & \text{o.w.} \end{cases} \tag{4}$$

Let $\hat{H}^F$ denote the F-fold convolution of $\hat{H}^1$ with itself, so that $\operatorname{supp} \hat{H}^F \subseteq [-F\cdot b, F\cdot b]$. Here and below F is a parameter that we will choose to satisfy $F \ge 2d, F = \Theta(d)$. The Fourier transform of $\hat{H}^1$ is the Dirichlet kernel (see e.g. [SS03], page 37):

$$H_j^1 = \frac{1}{b-1}\sum_{|i|<b/2}\omega^{ij} = \frac{\sin(\pi(b-1)j/n)}{(b-1)\sin(\pi j/n)} \text{ for } j \neq 0$$
$$H_0^1 = 1.$$

Thus, $H_j^F = \left(\frac{1}{b-1}\sum_{|i|<b/2}\omega^{ij}\right)^F = \left(\frac{\sin(\pi(b-1)j/n)}{(b-1)\sin(\pi j/n)}\right)^F$ for $j \neq 0$, and $H_0^F = 1$. For $i \in [n]^d$ let

$$G_i = \prod_{s=1}^d H_{i_s}^F, \tag{5}$$

so that $\hat{G}_i = \prod_{s=1}^d \hat{H}_{i_s}^F$ and $\operatorname{supp}\hat{G} \subseteq [-F\cdot b, F\cdot b]^d$. We will use the following simple properties of G:

Lemma 3.1. *For any $F \geq 1$ one has*

1 $G_0 = 1$, *and* $G_j \in [\frac{1}{(2\pi)^{F\cdot d}}, 1]$ *for all* $j \in [n]^d$ *such that* $||j||_\infty \leq \frac{n}{2b}$;

2 $|G_j| \leq \left(\frac{2}{1+(b/n)||j||_\infty}\right)^F$ *for all* $j \in [n]^d$

as long as $b \geq 3$.

The two properties 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). The proof of the lemma is similar to the analysis of filters in [HIKP12b, IKP14] and is given in Appendix A. We will not use the lower bound on G given in the first claim of Lemma 3.1 for our $\widetilde{O}(N)$ time algorithm in this paper. We state the Lemma in full form for later use in [IK14], where we present a sublinear time algorithm.

The following property of pseudorandom permutations $\pi_{\Sigma,q}$ makes hashing using our filters effective (i.e. allows us to bound noise in each bucket, see Lemma 3.3, see below):

Lemma 3.2. *Let* $i, j \in [n]^d$. *Let* Σ *be uniformly random with odd determinant. Then for all* $t \geq 0$

$$\mathbf{Pr}[||\Sigma(i-j)||_\infty \leq t] \leq 2(2t/n)^d.$$

A somewhat incomplete proof of this lemma for the case $d = 2$ appeared as Lemma B.4 in [GHI$^+$13]. We give a full proof for arbitrary d in Appendix A.

We access the signal x via random samples of $\hat{x}$, namely by computing the signal $\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{x}) \cdot \hat{G})$. As Lemma 3.3 below shows, this effectively "hashes" x into $B = b^d$ bins by convolving it with the filter G constructed above. Since our algorithm runs in $\tilde{O}(N)$ as opposed to $\tilde{O}(k)$ time, we can afford to work with bins around any location in time domain (we will be interested in locations of heavy hitters after applying the permutation, see Lemma 3.3). This improves the dependence of our sample complexity on d. The properties of the filtering process are summarized in

Lemma 3.3. *Let* $x \in \mathbb{C}^N$. *Choose* $\Sigma \in \mathcal{M}_{d\times d}, a, q \in [n]^d$ *uniformly at random, independent of* x. *Let*

$$u = \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{x}) \cdot \hat{G}),$$

where G is the filter constructed in (5). *Let* $\pi = \pi_{\Sigma,q}$.

For $i \in [n]^d$ *let* $\mu^2_{\Sigma,q}(i) = \sum_{j\in[n]^d\setminus\{i\}} |x_j G_{o_i(j)}|^2$, *where* $o_i(j) = \pi(j) - \pi(i)$ *as before. Suppose that* $F \geq 2d$. *Then for any* $i \in [n]^d$

1. $\mathbf{E}_{\Sigma,q}[\mu^2_{\Sigma,q}(i)] \leq C^d\|x\|_2^2/B$ *for a constant* $C > 0$.
2. *for any* Σ, q *one has* $\mathbf{E}_a[|\omega^{-a^T\Sigma i}u_{\pi(i)} - x_i|^2] \lesssim \mu^2_{\Sigma,q}(i) + \delta||x||_2^2$, *where the last term corresponds to the numerical error incurred from computing FFT with* $O(\log 1/\delta)$ *bits of machine precision.*

The proof of Lemma 3.3 is given in Appendix A.

Remark 3.4. *We assume throughout the paper that arithmetic operations are performed on* $C\log N$ *bit numbers for a sufficiently large constant* $C > 0$ *such that* $\delta||x||_2^2 \leq \delta(R^*)^2 n\mu^2 \leq \mu^2/N$, *so that the effect of rounding errors on Lemma 3.3 is negligible.*

4 The algorithm

In this section we present our $\tilde{O}(N)$ time algorithm that achieves $d^{O(d)}\frac{1}{\epsilon}k\log N$ sample complexity and give the main definitions required for its analysis. Our algorithm follows the natural iterative recovery scheme. The main body of the algorithm (Algorithm 1) takes samples of the signal $\widehat{x}$ and repeatedly calls the LOCATEANDESTIMATE function (Algorithm 2), improving estimates of the values of dominant elements of x over $O(\log n)$ iterations. Crucially, samples of $\widehat{x}$ are only taken at the beginning of Algorithm 1 and passed to each invocation of LOCATEANDESTIMATE. Each invocation of LOCATEANDESTIMATE takes samples of $\widehat{x}$ as well as the current approximation χ to x as input, and outputs a constant factor approximation to dominant elements of $x - \chi$ (see section 6 for analysis of LOCATEANDESTIMATE).

Algorithm 1 Overall algorithm: perform Sparse Fourier Transform

```
 1: procedure SPARSEFFT($\hat{x}, k, \epsilon, R^*, \mu$)            ▷ $R^*$ is a bound on $||x||_\infty/(\sqrt{\epsilon}\mu)$
 2:     $\chi^{(0)} \leftarrow 0$                ▷ in $\mathbb{C}^n$.        ▷ $\mu$ is the noise level (defined in (3))
 3:     $T \leftarrow \log_2 R^*$
 4:     $B \leftarrow k/(\epsilon\alpha^d)$                     ▷ Choose $\alpha$ so that $B = b^d$ for $b$ a power of 2
 5:     $G, \widehat{G} \leftarrow$ filter as in (5)
 6:     $r_{max} \leftarrow \Theta(\log N)$
 7:     for $r = 0$ to $r_{max}$ do
 8:         Choose $\Sigma_r \in \mathcal{M}_{d\times d}, a_r, q_r \in [n]^d$ uniformly at random
 9:         For $r = 1, \ldots, r_{max}$, $u^r \leftarrow \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{x}) \cdot \hat{G})$
10:         ▷ Note that $u^r \in \mathbb{C}^{[n]^d}$ for all $r$
11:     end for
12:     for $t = 0, 1, \ldots, T-1$ do
13:         $\chi' \leftarrow$ LOCATEANDESTIMATE($\hat{x}, \chi^{(t)}, \{(\Sigma_r, a_r, b_r), u_r\}_{r=1}^{r_{max}}, r_{max}, \widehat{G}, 4\sqrt{\epsilon}\mu 2^{T-(t+1)}$)
14:         $\chi^{(t+1)} \leftarrow \chi^{(t)} + \chi'$
15:     end for
16:     return $\chi^{(T)}$
17: end procedure
```

We first give intuition behind the algorithm and the analysis. We define the set $S \subseteq [n]^d$ to contain elements $i \in [n]^d$ such that $|x_i|^2 \geq \epsilon\mu^2$ (i.e. S is the set of *head elements* of x). As we show later (see section 6) it is sufficient to locate and estimate all elements in S up to $O(\epsilon\mu^2)$ error term in order obtain ℓ_2/ℓ_2 guarantees that we need[5]. Algorithm 1 performs $O(\log N)$ rounds of location and estimation, where in each round the located elements are estimated up to a constant factor. The crucial fact that allows us to obtain an optimal sampling bound is that the algorithm uses the same samples during these $O(\log N)$ rounds. Thus, our main goal is to show that elements of S will be successfully located and estimated throughout the process, *despite the dependencies* between the sampling pattern and the residual signal $x - \chi^{(t)}$ that arise due to reuse of randomness in the main loop of Algorithm 1.

We now give an overview of the main ideas that allow us to circumvent lack of independence. Recall that our algorithm needs to estimate all *head elements*, i.e. elements $i \in S$, up to $O(\epsilon\mu^2)$ additive error. Fix an element $i \in S$ and for each permutation π consider balls $\mathbb{B}^\infty_{\pi(i)}((n/b) \cdot 2^{t+2})$ around the position that i occupies in the permuted signal. For simplicity, we assume that $d = 1$, in which case the balls

[5]In fact, one can see that our algorithm gives the stronger ℓ_∞/ℓ_2 guarantee

Algorithm 2 LOCATEANDESTIMATE($\hat{x}, \chi, \{(\Sigma_r, a_r, q_r), u_r\}_{r=1}^{r_{max}}, r_{max}, \widehat{G}, \nu$)

```
1: procedure LOCATEANDESTIMATE($\hat{x}, \chi, \{(\Sigma_r, a_r, q_r), u_r\}_{r=1}^{r_{max}}, r_{max}, \widehat{G}, \nu$)
2:     Requires that $||x - \chi||_\infty \leq 2\nu$
3:     Guarantees that $||x - \chi - \chi'||_\infty \leq \nu$
4:     $L \leftarrow \emptyset$
5:     $w \leftarrow 0$
6:     for $r = 0$ to $r_{max}$ do
7:         $v^r \leftarrow u^r - \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{\chi}) \cdot \hat{G})$    ▷ Update signal: note this does not use any new samples
8:     end for
9:     for $f \in [n]^d$ do
10:        $S \leftarrow \emptyset$
11:        for $r = 0$ to $r_{max}$ do
12:            Denote permutation $\pi_{\Sigma_r,q_r}$ by $\pi$
13:            $S \leftarrow S \cup \{v^r_{\pi(f)} \cdot \omega^{-a^T\Sigma f}\}$
14:        end for
15:        $\eta \leftarrow$ median($S$)    ▷ Take the median coordinatewise
16:        If $|\eta| \leq \nu/2$ then continue    ▷ Continue if the estimated value is too small
17:        $L \leftarrow L \cup \{f\}$
18:        $w_f \leftarrow \eta$
19:    end for
20:    return $w$
21: end procedure
```

$\mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})$ are just intervals:

$$\mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2}) = \pi(i) + [-(n/b)\cdot 2^{t+2}, +(n/b)\cdot 2^{t+2}], \tag{6}$$

where addition is modulo n. Since our filtering scheme is essentially "hashing" elements of x into $B = \Omega(|S|/\alpha)$ "buckets" for a small constant $\alpha > 0$, we expect at most $O(\alpha)2^{t+2}$ elements of S to land in a ball (6) (i.e. the expected number of elements that land in this ball is proportional to its volume).

First suppose that this expected case occurs for any permutation, and assume that all head elements (elements of S) have the same magnitude (equal to 1 to simplify notation). It is now easy to see that the number of elements of S that are mapped to (6) *for any* $t \geq 0$ does not exceed its expectation (we call element i "isolated" with respect to π *at scale* t in that case), then the contribution of S to i's estimation error is $O(\alpha)$. Indeed, recall that the contribution of an element $j \in [n]^d$ to the estimation error of i is about $(1+(b/n)|\pi(i)-\pi(j)|)^{-F}$ by Lemma 3.1, (2), where we can choose F to be any constant without affecting the asymptotic sample complexity. Thus, even if $F = 2$, corresponding to the boxcar filter, the contribution to i's estimation error is bounded by

$$\begin{aligned}&\sum_{t\geq 0, (n/b)\cdot 2^{t+2}<n/2} \left|\pi(S)\cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})\right| \cdot \max_{y\in \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})\setminus \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+1})} |G_{\pi(i)-y}| \\ &= \sum_{t\geq 0, (n/b)\cdot 2^{t+2}<n/2} O(\alpha 2^{t+2})\cdot (1+2^{t+1})^{-F} = O(\alpha).\end{aligned}$$

Thus, if not too many elements of S land in intervals around $\pi(i)$, then the error in estimating i is at most $O(\alpha)$ times the maximum head element in the current residual signal (plus noise, which can be handled separately). This means that the median in line 15 of Algorithm 2 is an additive $\pm O(\alpha)||x-\chi||_\infty$ approximation to element f. Since Algorithm 2 only updates elements that pass the magnitude test in line 16, we can conclude that whenever we update an element, we have a $(1\pm O(\alpha))$ multiplicative estimate of its value, which is sufficient to conclude that we decrease the ℓ_∞ norm of $x-\chi$ in each iteration. Finally, we crucially ensure that the signal is never updated outside of the set S. This means that the set of head elements is fixed in advance and does not depend on the execution path of the algorithm! This allows us to formulate a notion of isolation with respect to the set S of head elements fixed in advance, and hence avoid issues arising from the lack of independence of the signal $x-\chi$ and the permutaions we choose.

We formalize this notion in Definition 5.2, where we define what it means for $i \in S$ to be isolated under π. Note that the definition is essentially the same as asking that the balls in (6) do not contain more than the expected number of elements of S. However, we need to relax the condition somewhat in order to argue that it is satisfied with good enough probability *simultaneously for all* $t \geq 0$. A adverse effect of this relaxation is that our bound on the number of elements of S that are mapped to a balls around $\pi(i)$ are weaker than what one would have in expectation. This, however, is easily countered by choosing a filter with stronger, but still polynomial, decay (i.e. setting the parameter F in the definiion of our filter G in (5) sufficiently large).

As noted before, the definition of being isolated crucially only depends on the **locations** of heavy hitters as opposed to their **values**. This allows us to avoid an (intractable) union bound over all signals that appear during the execution of our algorithm. We give formal definitions of isolationin section 5, and then use them to analyze the algorithm in section 6.

5 Isolated elements and main technical lemmas

We now give the technical details for the outline above.

Definition 5.1. *For a permutation π and a set $S \subseteq [n]^d$ we denote $S^\pi := \{\pi(x) : x \in S\}$.*

Definition 5.2. *Let $\Sigma \in \mathcal{M}_{d\times d}, q \in [n]^d$, and let $\pi = \pi_{\Sigma,q}$. We say that an element i is* isolated *under permutation π* at scale t *if*

$$|(S \setminus \{i\})^\pi \cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})| \le \alpha^{d/2} 2^{(t+3)d} \cdot 2^t.$$

We say that i is simply isolated *under permutation $\pi_{\Sigma,q}$ if it is isolated under $\pi_{\Sigma,q}$ at all scales $t \ge 0$.*

Remark 5.3. *We will use the definition of isolated elements for a set S with $|S| \approx k/\epsilon$.*

The following lemma shows that every $i \in [n]^d$ is likely to be isolated under a randomly chosen permutation π:

Lemma 5.4. *Let $S \subseteq [n]^d, |S| \le 2k/\epsilon$. Let $B \ge k/(\epsilon\alpha^d)$. Let $\Sigma \in \mathcal{M}_{d\times d}, q \in [n]^d$ be chosen uniformly at random, and let $\pi = \pi_{\Sigma,q}$. Then each $i \in [n]^d$ is* isolated *under permutation π with probability at least $1 - O(\alpha^{d/2})$.*

Proof. By Lemma 3.2, for any fixed $i, j \ne i$ and any radius $r \ge 0$,

$$\mathbf{Pr}_\Sigma[\|\Sigma(i-j)\|_\infty \le r] \le 2(2r/n)^d. \tag{7}$$

Setting $r = (n/b)\cdot 2^{t+2}$, we get

$$\mathbf{E}_{\Sigma,q}[|(S \setminus \{i\})^\pi \cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})|] = \sum_{j \in S\setminus\{i\}} \mathbf{Pr}_{\Sigma,q}[\pi(j) \in \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})] \tag{8}$$

Since $\pi_{\Sigma,q}(i) = \Sigma(i-q)$ for all $i \in [n]^d$, we have

$$\begin{aligned}\mathbf{Pr}_{\Sigma,q}[\pi(j) \in \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})] &= \mathbf{Pr}_{\Sigma,q}[\|\pi(j)-\pi(i)\|_\infty \le (n/b)\cdot 2^{t+2}] \\ &= \mathbf{Pr}_{\Sigma,q}[\|\Sigma(j-i)\|_\infty \le (n/b)\cdot 2^{t+2}] \le 2(2^{t+3}/b)^d,\end{aligned}$$

where we used (7) in the last step. using this in (8), we get

$$\mathbf{E}_{\Sigma,q}[|(S \setminus \{i\})^\pi \cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})|] \le |S|\cdot(2^{t+3}/b)^d \le (|S|/B)\cdot 2^{(t+3)d} \lesssim \epsilon\alpha^d 2^{(t+3)d}.$$

Now by Markov's inequality we have that i fails to be isolated at scale t with probability at most

$$\mathbf{Pr}_{\Sigma,q}\left[|(S \setminus \{i\})^\pi \cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})| > \alpha^{d/2} 2^{(t+3)d+t}\right] \lesssim 2^{-t}\alpha^{d/2}.$$

Taking the union bound over all $t \ge 0$, we get

$$\mathbf{Pr}_{\Sigma,q}[i \text{ is not isolated}] \lesssim \sum_{t\ge 0} 2^{-t}\alpha^{d/2} \lesssim \alpha^{d/2}$$

as required.

□

The contribution of tail noise to an element $i \in [n]^d$ is captured by the following

Definition 5.5. *Let $x \in \mathbb{C}^N$. Let $S \subseteq [n]^d, |S| \leq 2k/\epsilon$. Let $B \geq k/(\epsilon\alpha^d)$. Let $u = \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{x}) \cdot \hat{G})$. We say that an element $i \in [n]^d$ is* well-hashed with respect to noise under $(\pi_{\Sigma,q}, a)$ *if*

$$|u_{\pi(i)}\omega^{-a^T\Sigma i} - x_i|^2 = O(\sqrt{\alpha})\epsilon\mu^2,$$

where we let $\pi = \pi_{\Sigma,q}$ to simplify notation.

Lemma 5.6. *Let $S \subseteq [n]^d, |S| \leq 2k/\epsilon$, be such that $||x_{[n]^d\setminus S}||_\infty \leq \mu$. Let $B \geq k/(\epsilon\alpha^d)$. Let $\Sigma_r \in \mathcal{M}_{d\times d}, q_r, a_r \in [n]^d, r = 1, \ldots, r_{max}, r_{max} \geq (C/\sqrt{\alpha})\log N$ be chosen uniformly at random, where $\alpha > 0$ is a constant and $C > 0$ is a sufficiently large constant that depends on α. Then with probability at least $1 - N^{-\Omega(C)}$*

1. *each $i \in [n]^d$ is isolated with respect to S under at least $(1 - O(\sqrt{\alpha}))r_{max}$ permutations $\pi_r, r = 1, \ldots, r_{max}$;*
2. *each $i \in [n]^d$ is well-hashed with respect to noise under at least $(1 - O(\sqrt{\alpha}))r_{max}$ pairs $(\pi_r, a_r), r = 1, \ldots, r_{max}$.*

Proof. The first claim follows by an application of Chernoff bounds and Lemma 5.4. For the second claim, let $u = \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{x}_{[n]^d\setminus S}) \cdot \hat{G})$, where $(\Sigma, a, q) = (\Sigma_r, a_r, q_r)$ for some $r = 1, \ldots, r_{max}$. Letting $\pi = \pi_{\Sigma,q}$, by Lemma 3.3, (1) and (2) we have

$$\mathbf{E}_{\Sigma,q,a}[|u_{\pi(i)}\omega^{-a^T\Sigma i} - (x_{[n]^d\setminus S})_i|^2] \leq (C')^d||x_{[n]^d\setminus S}||^2/B + ||x||^2 \cdot N^{-\Omega(c)}$$

for a constant $C' > 0$, where we asssume that arithmetic operations are performed on $c\log N$-bit numbers for some constant $c > 0$. Since we assume that $R^* \leq \text{poly}(N)$, we have

$$\mathbf{E}_{\Sigma,q,a}[|u_{\pi(i)}\omega^{-a^T\Sigma i} - (x_{[n]^d\setminus S})_i|^2] \leq (C'')^d||x_{[n]^d\setminus S}||^2/B + \mu^2 \cdot N^{-\Omega(c)}.$$

Let $S^* \subset [n]^d$ denote a set of top k coefficients of x (with ties broken arbitrarily). We have

$$||x_{[n]^d\setminus S}||^2 \leq ||x_{[n]^d\setminus(S\cup S^*)}||^2 + ||x_{S^*\setminus S}||^2 \leq ||x_{[n]^d\setminus S}||^2 \leq ||x_{[n]^d\setminus S^*}||^2 + k \cdot ||x_{[n]^d\setminus S}||_\infty^2 \leq 2k\mu^2.$$

Since $B \geq k/(\epsilon\alpha^d)$, we thus have

$$\mathbf{E}_{\Sigma,q,a}[|u_{\pi(i)}\omega^{-a^T\Sigma i} - (x_{[n]^d\setminus S})_i|^2] \leq (C'''\alpha)^d\epsilon\mu^2$$

for a constant $C''' > 0$.

By Markov's inequality

$$\mathbf{Pr}_{\Sigma,q,a}[|u_{h(i)}\omega^{-a^T\Sigma i} - (x_{[n]^d\setminus S})_i|^2 > (C'''\sqrt{\alpha})^d\epsilon\mu^2] < \alpha^{d/2}.$$

As before, an application of Chernoff bounds now shows that each $i \in [n]^d$ is well-hashed with respect to noise with probability at least $1 - N^{-10}$, and hence all $i \in [n]^d$ are well-hashed with respect to noise with probability at least $1 - N^{-\Omega(C)}$ as long as α is smaller than an absolute constant. □

We now combine Lemma 5.4 with Lemma 5.6 to derive a bound on the noise in the "bucket" of an element $i \in [n]^d$ due to both heavy hitters and tail noise. Note that crucially, the bound only depends on the ℓ_∞ norm of the head elements (i.e. the set S), and in particular, works for *any signal* that coincides with x on the complement of S. Lemma 5.7 will be the main tool in the analysis of our algorithm in the next section.

Lemma 5.7. *Let $x \in \mathbb{C}^N$. Let $S \subseteq [n]^d$, $|S| \leq 2k/\epsilon$, be such that $||x_{[n]^d \setminus S}||_\infty \leq \mu$. Let $B \geq k/(\epsilon\alpha^d)$. Let $y \in \mathbb{C}^N$ be such that $y_{[n]^d\setminus S} = x_{[n]\setminus S}$ and $||y_S||_\infty \leq 4\sqrt{\epsilon}\mu 2^w$ for some $t \geq 0$. Let $u = \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{y}) \cdot \hat{G})$. Then for each $i \in [n]^d$ that is isolated and well-hashed with respect to noise under (Σ_r, q_r, a_r) one has*

$$|u_j\omega^{-a^T\Sigma i} - y_i|^2 \lesssim \sqrt{\alpha}\epsilon((4\mu 2^w)^2 + \mu^2).$$

Proof. We have

$$\left|u_j\omega^{-a^T\Sigma i} - x_i\right|^2 \leq 2|A^H|^2 + 2|A^T|^2,$$

where $A^H = u_j^H\omega^{-a^T\Sigma i} - (y_S)_i$ for $u^H = \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{y}_S)\cdot\hat{G})$ and $A^T = u_j^T\omega^{-a^T\Sigma i} - (y_{[n]^d\setminus S})_i$ for $u^T = \sqrt{N}\mathcal{F}^{-1}((P_{\Sigma,a,q}\hat{y}_{[n]^d\setminus S})\cdot\hat{G})$

We first bound A^H. Fix $i \in [n]^d$. If i is isolated, we have

$$|(S\setminus\{i\})^\pi \cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})| \leq \alpha^{d/2}2^{(t+3)d}\cdot 2^t.$$

for all $t \geq 0$. We have

$$\begin{aligned}
|A^H| = |u_j^H\omega^{-a^T\Sigma i} - y_i| &= \left|\sum_{j\in S\setminus\{i\}} y_j G_{o_i(j)}\omega^{-a^T\Sigma j}\right| \\
&\leq \sum_{j\in S\setminus\{i\}} |y_j G_{o_i(j)}| \\
&\leq ||y_S||_\infty \cdot \sum_{t\geq 0}\left(\frac{2}{1+2^{t+2}}\right)^F |(S\setminus\{i\})^\pi \cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})| \\
&\leq ||y_S||_\infty \cdot \sum_{t\geq 0}\left(\frac{2}{1+2^{t+2}}\right)^F |(S\setminus\{i\})^\pi \cap \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t+2})| \\
&\leq ||y_S||_\infty \cdot \sum_{t\geq 0}\left(\frac{2}{1+2^{t+2}}\right)^F \alpha^{d/2}2^{(t+3)d}\cdot 2^t = ||y_S||_\infty \cdot O(\alpha^{d/2}) = O(\alpha^{d/2}\sqrt{\epsilon}\mu 2^w)
\end{aligned}$$

as long as $F \geq 2d, F = \Theta(d)$. Further, if i is well-hashed with respect to noise, we have $|A^T| \lesssim \sqrt{\alpha}\epsilon\mu^2$. Putting these estimates together yields the result. □

6 Main result

In this section we use Lemma 5.7 to prove that our algorithm satisfies the stated ℓ_2/ℓ_2 sparse recovery guarantees. The proof consists of two main steps: Lemma 6.1 proves that one iteration of the peeling process (i.e. one call to LOCATEANDESTIMATE) outputs a list containing all elements whose values are close to the current ℓ_∞ norm of the residual signal. Furthermore, approximations that LOCATEANDESTIMATE returns for elements in its output list are correct up to a multiplicative $1 \pm 1/3$ factor. Lemma 6.2 then shows that repeated invocations of LOCATEANDESTIMATE reduce the ℓ_∞ norm of the residual signal as claimed.

Lemma 6.1. *Let $x \in \mathbb{C}^N$. Let $S \subseteq [n]^d$, $|S| \leq 2k/\epsilon$, be such that $||x_{[n]^d\setminus S}||_\infty \leq \mu$. Let $B \geq k/(\epsilon\alpha^d)$. Consider the t-th iteration of the main loop in Algorithm 1. Suppose that $||x - \chi||_\infty \leq 2\nu$. Suppose that each element $i \in [n]^d$ is isolated with respect to S and well-hashed with respect to noise under at least $(1 - O(\sqrt{\alpha}))r_{max}$ values of $r = 1, \ldots, r_{max}$. Let $y = x - \chi^{(t)}$, and let χ' denote the output of* LOCATE-ANDESTIMATE. *Then one has*

1. *$|\chi'_i - y_i| < \frac{1}{3}|y_i|$ for all $i \in L$;*
2. *all i such that $|y_i| \geq \nu$ are included in L.*

as long as $\alpha > 0$ is a sufficiently small constant.

Proof. We let $y := x - \chi^{(t)}$ to simplify notation. Fix $i \in [n]^d$.

Consider r such that i is isolated under π_r and well-hashed with respect to noise under (π_r, a_r). Then we have by Lemma 5.7

$$|v_j^r \omega^{-a^T \Sigma i} - y_i|^2 \lesssim \sqrt{\alpha}(\epsilon(||y_{[S]}||_\infty)^2 + \epsilon\mu^2) \lesssim \sqrt{\alpha}\epsilon((2\nu)^2 + \mu^2) \leq (\frac{1}{16}\nu)^2 \tag{9}$$

as long as α is smaller than an absolute constant.

Since each i is well-hashed with respect to at least at $1 - O(\sqrt{\alpha})$ fraction of permutations, we get that $|y_i - \eta| \leq \frac{1}{16}\nu$. Now if $i \in L$, it must be that $|\eta| > \nu/2$, but then

$$|y_i| \geq |\eta| - \nu/16 > \nu/2 - \nu/16 > (3/4)\nu. \tag{10}$$

This also implies that $|\chi'_i - y_i| \leq \nu/16 < (4/3)|y_i|/16 < |y_i|/3$, so the first claim follows.

For the second claim, it suffices to note that if $|y_i| > \nu$, then we must have $|\eta| \geq |y_i| - \nu/16 > \nu/2$, so i passes the magnitude test and is hence included in L. □

We can now prove the main lemma required for analysis of Algorithm 1:

Lemma 6.2. *Let $x \in \mathbb{C}^N$. Let $\Sigma_r \in \mathcal{M}_{d\times d}, a_r, q_r \in [n]^d, r = 1, \ldots, r_{max} = C \log N$, where $C > 0$ is a sufficiently large constant, be chosen uniformly at random. Then with probability at least $1 - N^{-\Omega(C)}$ one has $||x - \chi^{(T-1)}||_\infty \leq 4\sqrt{\epsilon}\mu$.*

Proof. We now fix a specific choice of the set $S \subseteq [n]^d$. Let

$$S = \{i \in [n]^d : |x_i| > \sqrt{\epsilon}\mu\}. \tag{11}$$

First note that $||x_{[n]^d \setminus S}||_\infty \leq \mu$. Also, we have $|S| \leq 2k/\epsilon$. Indeed, recall that $\mu^2 = \mathrm{Err}_k^2(x)/k$. If $|S| > 2k/\epsilon$, more than k/ϵ elements of S belong to the tail, amounting to at least $\epsilon\mu^2 \cdot (k/\epsilon) > \mathrm{Err}_k^2(x)$ tail mass. Thus, since $r_{max} \geq C \log N$, and by the choice of B in Algorithm 1, we have by Lemma 5.6 that with probability at least $1 - N^{-\Omega(C)}$

1. each $i \in [n]^d$ is isolated with respect to S under at least $(1 - O(\sqrt{\alpha}))r_{max}$ permutations $\pi_r, r = 1, \ldots, r_{max}$;
2. each $i \in [n]^d$ is well-hashed with respect to noise under at least $(1 - O(\sqrt{\alpha}))r_{max}$ permutations $\pi_r, r = 1, \ldots, r_{max}$.

This ensures that the preconditions of Lemma 6.1 are satisfied. We now prove the following statement for $t \in [0 : T]$ by induction on t:

1. $\chi^{(t)}_{[n]\setminus S} \equiv 0$
2. $||(x - \chi^{(t)})_S||_\infty \leq 4\sqrt{\epsilon}\mu 2^{T-t}$.

3. $|x_i - \chi_i^{(t)}| \leq |x_i|$ for all $i \in [n]^d$.

Base:$t = 0$ True by the choice of T.

Inductive step: $t \to t+1$ Consider the list L constructed by LOCATEANDESTIMATE at iteration t. Let $S^* := \{i \in S : |(x - \chi^{(t)})_i| > 4\sqrt{\epsilon}\mu 2^{T-(t+1)}\}$. We have $S^* \subseteq L$ by Lemma 6.1, (2). Thus,

$$||(x - \chi^{(t+1)})_S||_\infty \leq \max\{||(x - \chi^{(t+1)})_{S^*}||_\infty, ||(x - \chi^{(t+1)})_{S \setminus S^*}||_\infty, ||(x - \chi^{(t+1)})_{[n]^d \setminus S}||_\infty\}.$$

By Lemma 6.1, (1) we have $|x_i - \chi_i^{(t+1)}| \leq |x_i - \chi_i^{(t)}|/3$ for all $i \in L$, so (3) follows. Furthemore, this implies that

1. $||(x - \chi^{(t+1)})_{S^*}||_\infty \leq ||x - \chi^{(t)}||_\infty/3 \leq 4\sqrt{\epsilon}\mu 2^{T-(t+1)}$ by the inductive hypothesis;
2. $||(x - \chi^{(t+1)})_{S \setminus S^*}||_\infty \leq 4\sqrt{\epsilon}\mu 2^{T-(t+1)}$ by definition of S^*;
3. $||(x - \chi^{(t+1)})_{[n]^d \setminus S}||_\infty = ||x_{[n]^d \setminus S}||_\infty \leq 4\sqrt{\epsilon}\mu 2^{T-(t+1)}$ by the inductive hypothesis together with the definition of μ and the fact that $t \leq T-1$.

This proves (2).

Finally, by Lemma 6.1, (2) only elements i such that $|(x - \chi^{(t)})_i| > \frac{3}{4} 4\sqrt{\epsilon}\mu 2^{T-t} \geq (3/2) 4\sqrt{\epsilon}\mu$ are included in L. Since $|(x - \chi^{(t)})_i| \leq |x_i|$, this means that $|x_i| > 4\mu$, i.e. $i \in S$ and $\chi_{[n] \setminus S}^{(t+1)} = 0$, as required.

□

We can now prove

Theorem 6.3. *Algorithm 1 returns a vector χ such that*

$$||x - \chi||_2 \leq (1 + O(\epsilon)) \operatorname{Err}_k(x).$$

The number of samples is bounded by $d^{O(d)} \frac{1}{\epsilon} k \log N$, and the runtime is bounded by $O(N \log^3 N)$.

Proof. By Lemma 6.2 we have by setting $t = T-1$ $||x - \chi||_\infty \leq 8\sqrt{\epsilon}\mu$, so

$$||x - \chi||_2^2 \leq ||(x - \chi)_{[k]}||_\infty^2 \cdot k + ||(x - \chi)_{[n]^d \setminus [k]}||_2^2 \leq ||(x - \chi)_{[k]}||_\infty^2 \cdot k + ||x_{[n]^d \setminus [k]}||_2^2 \leq (1 + O(\epsilon)) \operatorname{Err}_k^2(x),$$

where we used Lemma 6.2, (3) to upper bound $||(x - \chi)_{[n]^d \setminus [k]}||_2^2$ with $||x_{[n]^d \setminus [k]}||_2^2$.

We now bound sampling complexity. The support of the filter G is bounded by $O(BF^d) = d^{O(d)} \frac{1}{\epsilon} k$ by construction. We are using $r_{max} = \Theta(\log N)$, amounting to $d^{O(d)} \frac{1}{\epsilon} k \log N$ sampling complexity overall. The location and estimation loop takes $O(N \log^3 N)$ time: each time the vector v^r is calculated in LOCATEANDESTIMATE $O(N \log N)$ time is used by the FFT computation, so since we compute $O(\log N)$ vectors during each of $O(\log N)$ iterations, this results in an $O(N \log^3 N)$ contribution to runtime. □

7 Experimental evaluation

In this section we describe results of an experimental evaluation of our algorithm from section 4. In order to avoid the issue of numerical precision and make the notion of recovery probability well-defined, we focus the problem of *support recovery*, where the goal is to recover the *positions* of the non-zero coefficients. We first describe the experimental setup that we used to evaluate our algorithm, and follow with evaluation results.

7.1 Experimental setup

We present experiments for support recovery from one-dimensional Fourier measurements (i.e. $d = 1$). In this problem one is given frequency domain access to a signal $\widehat{x}$ that is *exactly* k-sparse in the time domain, and needs to recovery the support of x exactly. The support of x was always chosen to be uniformly among subsets of $[N]$ of size k. We denote the sparsity by k, and the support of x by $S \subseteq [n]^d$.

We compared our algorithm to two algorithms for sparse recovery:

- ℓ_1-minimization, a state-of-the-art technique for practical sparse recovery using Gaussian and Fourier measurements. The best known sample bounds for the sample complexity in the case of approximate sparse recovery are $O(k \log^3 k \log N)$ [CT06, RV08, CGV12]. The running time of ℓ_1 minimization is dominated by solving a linear program. We used the implementation from SPGL1 [vdBF08, vdBF07], a standard Matlab package for sparse recovery using ℓ_1-minimization. For this experiment we let x_i be chosen uniformly random on the unit circle in the complex plane when $i \in S$ and equal to 0 otherwise.

- Sequential Sparse Matching Pursuit (SSMP) [BI09]. SSMP is an iterative algorithm for sparse recovery using sparse matrices. SSMP has optimal $O(k \log(N/k))$ sample complexity bounds and $\tilde{O}(N)$ runtime. The sample complexity and runtime bounds are similar to that of our algorithm, which makes SSMP a natural point of comparison. Note, however, that the measurement matrices used by SSMP are binary and sparse, i.e., very different from the Fourier matrix. For this experiment we let x_i be uniformly random in $\{-1, +1\}$ when $i \in S$ and 0 otherwise.

Our implementation of Algorithm 1 uses the following parameters. First, the filter G was the simple boxcar filter with support $B = k + 1$. The number of measurements r_{max} was varied between 5 and 25, with the phase transition occuring around $r_{max} = 18$ for most values of k. The geometric sequence of thresholds that LOCATEANDESTIMATE is called with was chosen to be powers of 1.2 (empirically, ratios closer to 1 improve the performance of the algorithm, at the expense of increased runtime). We use $N = 2^{15}$ and $k = 10, 20, \ldots, 100$ for all experiments. We report empirical probability of recovery estimated from 50 trials.

In order to solve the support recovery problem using SPGL1 and Algorithm 1, we first let both algorithms recover an approximation x' to x, and then let

$$S := \{t \in [N] : |x'_t| \geq 1/2\}$$

denote the recovered support.

7.2 Results

Comparison to ℓ_1-minimization. A plot of recovery probability as a function of the number of (complex) measurements and sparsity for SPGL1 and Algorithm 1 is given in Fig. 2.

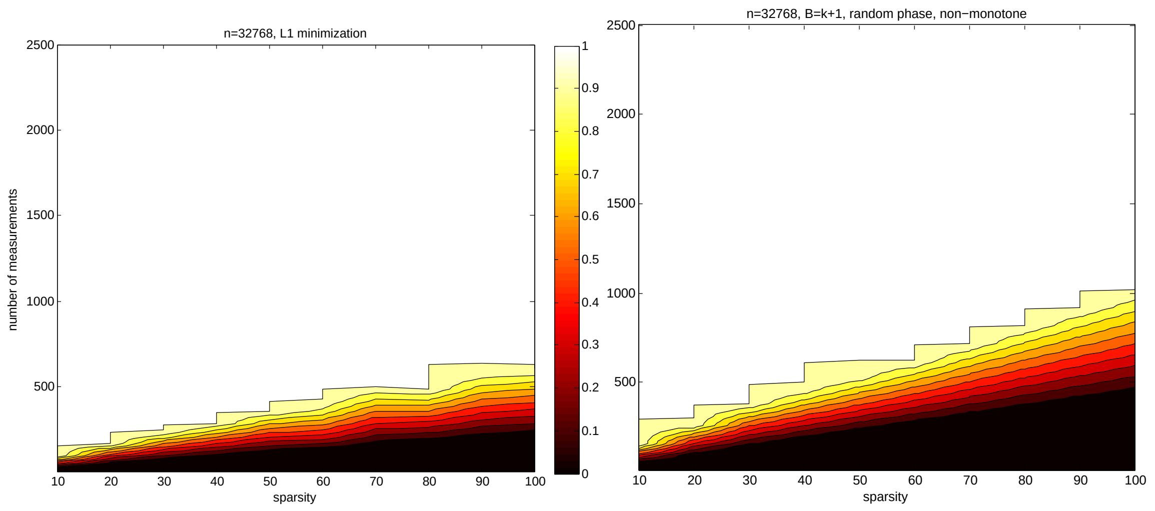

Figure 2: Success probability as a function of sparsity and number of measurements: SPGL1 (left panel) and Algorithm 1 (right panel). The number of complex measurements is reported.

The empirical sample complexity of our algorithm is within a factor of 2 of ℓ_1 minimization if success probability 0.9 is desired. The best known theoretical bounds for the general setting of approximate sparse recovery show that $O(k \log^3 k \log N)$ samples are sufficient. The runtime is bounded by the cost of solving an $N \times m$ linear program. Our algorithm provides comparable empirical performance, while providing optimal measurement bound and $\tilde{O}(N)$ runtime.

Comparison to SSMP. We now present a comparison to SSMP [BI09], which is a state-of-the art iterative algorithm for sparse recovery using sparse matrices. We compare our results to experiments in [Ber09]. Since the lengths of signal used for experiments with SSMP in [Ber09] are not powers of 2, we compare the results of [Ber09] for $N = 20000$ with our results for a larger value of N. In particular, we choose $N = 2^{15} > 20000$. Our results are presented in Fig. 3. Since experiments in [Ber09] used real measurements, we multiply the number of our (complex) measurements by 2 for this comparison (note that the right panel of Fig. 3 is the same as the right panel of Fig. 2, up to the factor of 2 in the number of measurements). We observe that our algorithm improves upon SSMP by a factor of about 1.15 when 0.9 success probability is desired.

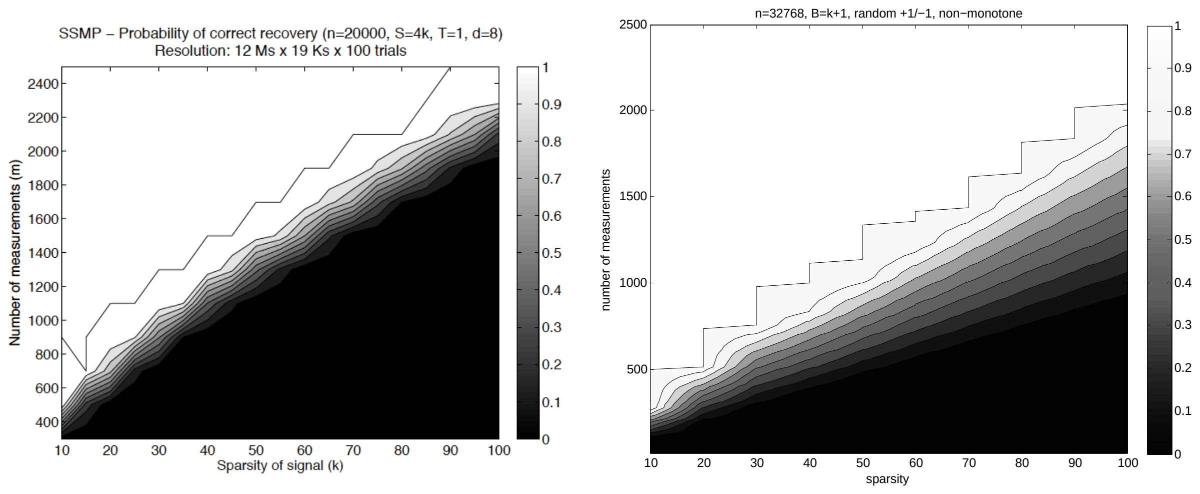

Figure 3: Success probability as a function of sparsity and number of measurements: SSMP (left panel) and Algorithm 1 (right panel). The number of real measurements is reported.

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.

[Ber09] Radu Berinde. Advances in sparse signal recovery methods. *MIT*, 2009.

[BI09] Radu Berinde and Piotr Indyk. Sequential sparse matching pursuit. *Allerton'09*, pages 36–43, 2009.

[BLM12] M. Bayati, M. Lelarge, and A. Montanari. Universality in polytope phase transitions and message passing algorithms. 2012.

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

[CGV12] Mahdi Cheraghchi, Venkatesan Guruswami, and Ameya Velingker. Restricted isometry of fourier matrices and list decodability of random linear codes. *SODA*, 2012.

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

[CT91] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley Interscience, 1991.

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

[DIPW10] Khanh Do Ba, Piotr Indyk, Eric Price, and David P. Woodruff. Lower Bounds for Sparse Recovery. *SODA*, 2010.

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

[FR13] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer, 2013.

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

[GHI$^+$13] Badih Ghazi, Haitham Hassanieh, Piotr Indyk, Dina Katabi, Eric Price, and Lixin Shi. Sample-optimal average-case sparse fourier transform in two dimensions. *arXiv preprint arXiv:1303.1209*, 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] Sabine Heider, Stefan Kunis, Daniel Potts, and Michael Veit. A sparse prony fft. *SAMPTA*, 2013.

[IK14] Piotr Indyk and Michael Kapralov. Sample-Optimal Fourier Sampling in Any Constant Dimension – Part II. *manuscript*, 2014.

[IKP14] Piotr Indyk, Michael Kapralov, and Eric 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.

[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.

[LWC12] D. Lawlor, Y. Wang, and A. Christlieb. Adaptive sub-linear time fourier algorithms. *arXiv:1207.6368*, 2012.

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

[MS78] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-holland Publishing Company, 2nd edition, 1978.

[PR13] Sameer Pawar and Kannan Ramchandran. Computing a k-sparse n-length discrete fourier transform using at most 4k samples and o (k log k) complexity. *ISIT*, 2013.

[PR14] Sameer Pawar and Kannan Ramchandran. A robust ffast framework for computing a k-sparse n-length dft in o(k log n) sample complexity using sparse-graph codes. *Manuscript*, 2014.

[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.

[SS03] Elias M. Stein and Rami Shakarchi. *Fourier Analysis:An Introduction*. Princeton University Press, 2003.

[vdBF07] E. van den Berg and M. P. Friedlander. SPGL1: A solver for large-scale sparse reconstruction, June 2007. http://www.cs.ubc.ca/labs/scl/spgl1.

[vdBF08] E. van den Berg and M. P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

[YBL$^+$12] Juhwan Yoo, S. Becker, M. Loh, M. Monge, E. Candès, and A. E-Neyestanak. A 100MHz–2GHz 12.5x subNyquist rate receiver in 90nm CMOS. In *IEEE RFIC*, 2012.

A Omitted proofs

Proof of Lemma 2.2:

$$\begin{aligned}
\mathcal{F}^{-1}(P_{\Sigma,a,q}\hat{x})_{\pi_{\Sigma,q}(i)} =& \frac{1}{\sqrt{N}}\sum_{j\in[n]^d}\omega^{j^T\Sigma(i-q)}(P_{\Sigma,a,q}\hat{x})_j\\
=& \frac{1}{\sqrt{N}}\sum_{j\in[n]^d}\omega^{j^T\Sigma(i-q)}\hat{x}_{\Sigma^T(j-a)}\omega^{j^T\Sigma q}\\
=& \frac{1}{\sqrt{N}}\sum_{j\in[n]^d}\omega^{j^T\Sigma i}\hat{x}_{\Sigma^T(j-a)}\\
=& \frac{1}{\sqrt{N}}\sum_{j\in[n]^d}\omega^{i^T\Sigma^T(j-a+a)}\hat{x}_{\Sigma^T(j-a)}\\
=& \omega^{a^T\Sigma i}\frac{1}{\sqrt{N}}\sum_{j\in[n]^d}\omega^{i^T\Sigma^T(j-a)}\hat{x}_{\Sigma^T(j-a)} = \omega^{a^T\Sigma i}x_i
\end{aligned}$$

□

Proof of Lemma 3.1: Since $|\sin(\pi x)| \le 1, |\sin(\pi x)| \le |\pi x|$ for all x and $|\sin(\pi x)| \ge 2|x|$ for $|x| \le 1/2$, we have

$$\left|\frac{\sin(\pi(b-1)j/n)}{(b-1)\sin(\pi j/n)}\right|^F \le \left(\frac{2}{(b-1)\pi(j/n)}\right)^F \tag{12}$$

for all j. We also have that the maximum absolute value is achieved at 0. Also, for any $j \in [n]$ such that $|j| \le \frac{n}{2b}$ one has

$$\left|\frac{\sin(\pi(b-1)j/n)}{(b-1)\sin(\pi j/n)}\right|^F \ge \left|\frac{(1/2)(b-1)j/n}{(b-1)\pi(j/n)}\right|^F \ge \left|\frac{(1/2)}{\pi}\right|^F = \frac{1}{(2\pi)^F}G_0.$$

which gives **(1)**.

(2) follows from (12) by writing

$$|G_j| = \prod_{s=1}^d H_{j_s}^F \le H_{||j||_\infty}^F \le \left(\frac{2}{(b-1)\pi(||j||_\infty/n)}\right)^F \le \left(\frac{2}{1+(b/n)||j||_\infty}\right)^F$$

as long as $b \ge 3$.

□

Proof of Lemma 3.2: We assume wlog that $j = 0$. Let g be the largest integer such that 2^g divides all of $i_1, \ldots, i_d$. We first assume that $g = 0$, and handle the case of general g later. For each $q \ge 0$ let

$$J_q = \{s \in [d] : i_s = 2^q e_s,\ e_s \text{ odd}\}. \tag{13}$$

Since we assume that $g = 0$, we have $J_0 \ne \emptyset$. We first prove that

$$\textbf{Pr}[||Mi||_\infty \le t] \le 2(t/n)^d$$

when M is sampled uniformly at random from a set $\mathcal{D}'$ that is a superset of the set of matrices with odd determinant, and then show that M is likely to have odd determinant when drawn from this distribution, which implies the result.

We denote the s-th column of M by M_s. With this notation we have $Mi = \sum_{s=1}^{d} M_s i_s$, where i_s is the s-th entry of i. Let

$$\mathcal{D}' = \{M \in \mathbb{Z}^{d\times d}/n : \sum_{s\in J_0} M_s \neq \mathbf{0} \bmod 2\},$$

i.e. we consider the set of matrices M whose columns with indices in J_0 do not add up to the all zeros vector modulo 2, and are otherwise unconstrained. We first note that for any $s \in [1:d]$ we can write

$$M_s = M'_s + 2M''_s,$$

where $M'_s \in \{0,1\}^s$ is uniform, and M''_s is uniform in $[n/2]^d$ (and independent of M'_s). When M is sampled from $\mathcal{D}'$, one has that $M'_s, s \in J_0$ are conditioned on not adding up to 0 modulo 2.

We first derive a more convenient expression for the distribution of M_s. In particular, note that $2M''_s$ is distributed identically to $2U$, where U is uniform in $[n]^d$ as opposed to $[n/2]^d$. Thus, from now on we assume that we have

$$M_s = M'_s + 2M''_s,$$

where $M'_s \in \{0,1\}$ is uniform and M''_s is uniform in $[n]^d$. One then has for all $s \in J_0$ (all operations are modulo n)

$$M_s i_s = (M'_s + 2M''_s)(1 + 2e_s) = M'_s + 2(e_s M'_s + M''_s(1 + 2e_s)) = M'_s + 2Q_s,$$

where Q_s is uniform in $[n]^d$ and independent of M'_s. This is because $M''_s(1 + 2e_s)$ is uniform in $[n]^d$ since M''_s is (where we use the fact that $1 + 2e_s$ is odd). We have thus shown that the distribution of $\sum_{s\in J_0} M_s i_s$ can be generated as follows: one samples bits $M'_s \in \{0,1\}^d, s \in J_0$ uniformly at random conditional on $\sum_{s\in J_0} M'_s \neq 0$, and then samples $Q_s \in [n]^d$ independently and uniformly at random, and outputs

$$\sum_{s\in J_0} M'_s + 2\sum_{s\in J_0} Q_s.$$

It remains to note that the distribution of

$$\sum_{s\in J_0} M'_s + 2\sum_{s\in J_0} Q_s + \sum_{k>0}\sum_{s\in J_k} M_s i_s$$

is the same as the distribution of

$$\sum_{s\in J_0} M'_s + 2\sum_{s\in J_0} Q_s.$$

Indeed, this is because by definition of $\mathcal{D}'$ for any $k > 0$ one has $i_s = 2^k e_s$, and $\{M_s\}_{s\notin J_0}$ are independent uniform in $[n]^d$. As a consequence,

$$\begin{aligned} 2\sum_{s\in J_0} Q_s + \sum_{k>0}\sum_{s\in J_k} M_s i_s &= 2\sum_{s\in J_0} Q_s + \sum_{k>0} 2^k \sum_{s\in J_k} M_s e_s \\ &= 2(\sum_{s\in J_0} Q_s + \sum_{k>0} 2^{k-1} \sum_{s\in J_k} M_s e_s) \end{aligned}$$

is distributed as $2U$, where U is uniform in $[n]^d$.

Thus, when M is drawn uniformly from $\mathcal{D}'$, we have that $\sum_{s=0}^{d} M_s i_s = Mi$ is uniformly random in $[n]^d \setminus 2[n]^d$, so

$$\mathbf{Pr}_{\Sigma\sim UNIF(\mathcal{D}')}[||\Sigma(i-j)||_\infty \leq t] \leq \frac{1}{1-2^{-d}}(t/n)^d \leq 2(t/n)^d.$$

So far we assumed that $g=0$. However, in general we can divide i by 2^g, concluding that $M(i/2^g)$ is uniform in $[n]^d \setminus 2[n]^d$, i.e. Mi is uniform in $2^g \cdot ([n]^d \setminus 2[n]^d)$, and the same conclusion holds.

It remains to deduce the same property when Σ is drawn uniformly at random from $\mathcal{M}_{d\times d}$, the set of matrices over $\mathbb{Z}^{d\times d}$ with odd determinant. Denote the set of such matrices by $\mathcal{D}_*$. First note that

$$\mathcal{D}_* \subset \mathcal{D}'$$

since the columns J_0 of a matrix $\Sigma \in \mathcal{D}_*$ cannot add up to 0 mod 2 since any such matrix would not be invertible over $\mathbb{Z}_n$. We now use the fact that a nonzero polynomial of degree at most d over $\mathbb{Z}_2$ is equal to 1 with probability at least 2^{-d} under a uniformly random assignment, as follows, for example, from the fact that the minimum distance of a d-th order Reed-Muller code over m binary variables is 2^{m-d}(see, e.g. [MS78]), we have

$$\mathbf{Pr}_{\Sigma\sim UNIF(\mathbb{Z}^{d\times d})}[\Sigma \in \mathcal{D}_*] \geq 2^{-d},$$

and hence

$$\mathbf{Pr}_{\Sigma\sim\mathcal{D}'}[\Sigma \in \mathcal{D}_*] \geq 2^{-d}.$$

We now get

$$\Pr_{\Sigma\sim UNIF(\mathcal{D}_*)}[||\Sigma(i-j)||_\infty \leq t] \leq \Pr_{\Sigma\sim UNIF(\mathcal{D}')}[||\Sigma(i-j)||_\infty \leq t] / \Pr_{\Sigma\sim UNIF(\mathcal{D}')}[\Sigma \in \mathcal{D}_*] \leq 2(2t/n)^d.$$

as required. $\square$

Proof of Lemma 3.3: By Lemma 3.2, for any fixed i and j and any $t \geq 0$,

$$\mathbf{Pr}_\Sigma[\|\Sigma(i-j)\|_\infty \leq t] \leq 2(2t/n)^d.$$

We have

$$u_{\pi(i)} = \sum_{j\in[n]^d} G_{o_i(j)} x_j \omega^{a^T\Sigma j} + \Delta_{\pi(i)} \tag{14}$$

for some Δ with $\|\Delta\|_\infty \leq ||x||_1 \cdot N^{-\Omega(c)} \leq ||x||_2 \cdot N^{-\Omega(c)}$ since we are assuming that arithmetic is performed using $c\log N$ bit words for a constant $c > 0$ that can be chosen sufficiently large. We define the vector $v \in \mathbb{C}^n$ by $v_{\Sigma j} = x_j G_{o_i(j)}$, so that

$$u_{\pi(i)} - \Delta_{\pi(i)} = \sum_{j\in[n]^d} \omega^{a^T j} v_j = \sqrt{N}\widehat{v}_a$$

so

$$u_{\pi(i)} - \omega^{a^T\Sigma i} x_i - \Delta_{\pi(i)} = \sqrt{N}(\widehat{v_{\{\Sigma i\}}})_a.$$

We have by (14) and the fact that $(X+Y)^2 \leq 2X^2 + 2Y^2$

$$\begin{aligned}
|u_{\pi(i)}\omega^{-a^T\Sigma i} - x_i|^2 &= |u_{\pi(i)} - \omega^{a^T\Sigma i} x_i|^2 \\
&\leq 2|u_{\pi(i)} - \omega^{a^T\Sigma i} x_i - \Delta_{\pi(i)}|^2 + 2\Delta_{\pi(i)}^2 \\
&= 2|\sum_{j\in[n]^d} G_{o_i(j)} x_j \omega^{a^T\Sigma j}|^2 + 2\Delta_{\pi(i)}^2
\end{aligned}$$

By Parseval's theorem, therefore, we have

$$\begin{aligned}\mathbf{E}_a[|u_{\pi(i)}\omega^{-a^T\Sigma i} - x_i|^2] &\le 2\mathbf{E}_a[|\sum_{j\in[n]^d} G_{o_i(j)}x_j\omega^{a^T\Sigma j}|^2] + 2\mathbf{E}_a[\Delta^2_{h(i)}] \\ &= 2(\|v_{\overline{\{\Sigma i\}}}\|_2^2 + \Delta^2_{h(i)}) \\ &\lesssim \sum_{j\in[n]^d\setminus\{i\}} |x_j G_{o_i(j)}|^2 + ||x||_2^2 \cdot N^{-\Omega(c)} \\ &\lesssim \sum_{j\in[n]^d\setminus\{i\}} |x_j G_{o_i(j)}|^2 + ||x||_2^2 \cdot N^{-\Omega(c)} \\ &\lesssim \mu^2_{\Sigma,q}(i) + ||x||_2^2 \cdot N^{-\Omega(c)}.\end{aligned} \tag{15}$$

We now prove **(2)**. We have

$$\mathbf{E}_{\Sigma,q}[\mu^2_{\Sigma,q}(i)] = \mathbf{E}_{\Sigma,q}[\sum_{j\in[n]^d\setminus\{i\}} |x_j G_{o_i(j)}|^2].$$

Recall that the filter G approximates an ideal filter, which would be 1 inside $\mathbb{B}^\infty_{\pi(i)}(n/b)$ and 0 everywhere else. We use the bound on $G_{o_i(j)} = G_{\pi(i)-\pi(j)}$ in terms of $||\pi(i)-\pi(j)||_\infty$ from Lemma 3.1, (2). In order to leverage the bound, we partition $[n]^d = \mathbb{B}^\infty_{\pi(i)}(n/2)$ as

$$\mathbb{B}^\infty_{\pi(i)}(n/2) = \mathbb{B}^\infty_{\pi(i)}(n/b) \cup \bigcup_{t=1}^{\log_2(b/2)} \left(\mathbb{B}^\infty_{\pi(i)}((n/b)2^t) \setminus \mathbb{B}^\infty_{\pi(i)}((n/b)2^{t-1})\right).$$

For simplicity of notation, let $X_0 = \mathbb{B}^\infty_{\pi(i)}(n/b)$ and $X_t = \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^t) \setminus \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^{t-1})$ for $t \ge 1$. For each $t \ge 1$ we have by Lemma 3.1, (2)

$$\max_{\pi(l)\in X_t} |G_{o_i(l)}| \le \max_{\pi(l)\not\in\mathbb{B}^\infty_{\pi(i)}((n/b)2^{t-1})} |G_{o_i(l)}| \le \left(\frac{2}{1+2^{t-1}}\right)^F.$$

Since the rhs is greater than 1 for $t \le 0$, we can use this bound for all $t \le \log_2(b/2)$. Further, by Lemma 3.2 we have for each $j \ne i$ and $t \ge 0$

$$\mathbf{Pr}_{\Sigma,q}[\pi(j)\in X_t] \le \mathbf{Pr}_{\Sigma,q}[\pi(j)\in \mathbb{B}^\infty_{\pi(i)}((n/b)\cdot 2^t)] \le 2(2^{t+1}/b)^d.$$

Putting these bounds together, we get

$$\begin{aligned}\mathbf{E}_{\Sigma,q}[\mu^2_{\Sigma,q}(i)] &= \mathbf{E}_{\Sigma,q}[\sum_{j\in[n]^d\setminus\{i\}} |x_j G_{o_i(j)}|^2] \\ &\le \sum_{j\in[n]^d\setminus\{i\}} |x_j|^2 \cdot \sum_{t=0}^{\log_2(b/2)} \mathbf{Pr}_{\Sigma,q}[\pi(j)\in X_t]\cdot \max_{\pi(l)\in X_t}|G_{o_i(l)}| \\ &\le \sum_{j\in[n]^d\setminus\{i\}} |x_j|^2 \cdot \sum_{t=0}^{\log_2(b/2)} (2^{t+1}/b)^d \cdot \left(\frac{2}{1+2^{t-1}}\right)^F \\ &\le \frac{2^F}{B}\sum_{j\in[n]^d\setminus\{i\}} |x_j|^2 \sum_{t=0}^{+\infty} 2^{(t+1)d-F(t-1)} \\ &\le 2^{O(d)}\frac{\|x\|_2^2}{B}\end{aligned}$$

as long as $F \geq 2d$ and $F = \Theta(d)$, proving **(2)**. $\square$